ABSTRACT

The present invention is directed to an integrated implementation framework and resulting medium for knowledge retrieval, management, delivery and presentation. The system includes a first server component that is responsible for adding and maintaining domain-specific semantic information and a second server component that hosts semantic and other knowledge for use by the first server component that work together to provide context and time-sensitive semantic information retrieval services to clients operating a presentation platform via a communication medium. Within the system, all objects or events in a given hierarchy are active Agents semantically related to each other and representing queries (comprised of underlying action code) that return data objects for presentation to the client according to a predetermined and customizable theme or “Skin.” This system provides various means for the client to customize and “blend” Agents and the underlying related queries to optimize the presentation of the resulting information. The present invention is directed to a framework or medium for knowledge retrieval, management, delivery and/or presentation. The system maintains semantic information and other knowledge to provide retrieval services to clients via a communication medium. Within the system, objects or events in a hierarchy are semantically related to each other, and agents implementing queries return data objects for presentation to the client according to a semantically influenced or determined theme. This system provides various means for the client to customize agents and/or the underlying related queries to optimize the presentation of the resulting information.
**FIGURE 1-1**

<table>
<thead>
<tr>
<th>INFORMATION TOOLS LAYER</th>
<th>PRESENTATION LAYER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypertext (Dumb Links)</td>
<td>Client-static, server-generated HTML, Flash, SVG, etc.</td>
</tr>
</tbody>
</table>

**FIGURE 1-2**

<table>
<thead>
<tr>
<th>INFORMATION TOOLS LAYER</th>
<th>PRESENTATION LAYER</th>
<th>PROTOCOL LAYER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypermedia (Dumb Semantic Links) + Metadata + Semantics/meaning</td>
<td>Client-static, server-generated HTML, Flash, SVG, etc. (generated from RDF and XML)</td>
<td>HTTP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SERVER-SIDE SEMANTIC QUERY LAYER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SQL, XQuery, XPath, etc.</td>
</tr>
</tbody>
</table>

**FIGURE 1-3**

Dumb Link (No semantics, no behavior)
FIGURE 2

Dumb Link + Metadata + Semantics (No behavior)

FIGURE 2

Dumb Link + Metadata + Semantics (No behavior)

Dumb Link + Metadata + Semantics (No behavior)

Dumb Link + Metadata + Semantics (No behavior)

Dumb Link + Metadata + Semantics (No behavior)

Dumb Link + Metadata + Semantics (No behavior)

Dumb Link + Metadata + Semantics (No behavior)

Dumb Link + Metadata + Semantics (No behavior)

Dumb Link + Metadata + Semantics (No behavior)

Dumb Link + Metadata + Semantics (No behavior)

Dumb Link + Metadata + Semantics (No behavior)

Dumb Link + Metadata + Semantics (No behavior)

Dumb Link + Metadata + Semantics (No behavior)
<table>
<thead>
<tr>
<th>Platform Layer</th>
<th>Today's Web</th>
<th>Information Nervous System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured Information Sources</td>
<td>Handled distinctly</td>
<td>Handled uniformly: as sources of metadata and semantics</td>
</tr>
<tr>
<td>Unstructured Information Sources</td>
<td>Handled distinctly</td>
<td>Handled uniformly: as sources of metadata and semantics</td>
</tr>
<tr>
<td>Information Indexing</td>
<td>Keyword-based; search engines do not &quot;understand&quot; the documents they search; no semantics</td>
<td>Concept-based; Extracts semantic metadata from unstructured information</td>
</tr>
<tr>
<td>Knowledge Indexing and Classification</td>
<td>N/A</td>
<td>Semantically encodes and abstracts information from both structured and unstructured sources</td>
</tr>
<tr>
<td>Knowledge Representation</td>
<td>N/A</td>
<td>Creates and maintains a self-healing semantic network of knowledge objects</td>
</tr>
<tr>
<td>Knowledge Ontology and Inference</td>
<td>N/A</td>
<td>Infers new connections and properties in the semantic network</td>
</tr>
<tr>
<td>Logic</td>
<td>Primarily the database. The database allows programmability for searching, rules, views, triggers, etc.</td>
<td>The knowledge-base. The knowledge-base will allow for similar programmability but at a semantic level.</td>
</tr>
<tr>
<td>Application</td>
<td>Server-side scripts that drive e-Business applications based primarily on user input; these applications are limited to presenting information based on user input</td>
<td>Server-side scripts that program the knowledge-base. These scripts can dynamically generate knowledge objects based on user input. Scripts can also include semantic commands for retrieval, notifications, and logic. Abecedarian Agents also reside at this layer for handling semantic user input (like natural-language)</td>
</tr>
<tr>
<td>Presentation</td>
<td>Authored or dynamically generated on the server; semantics are lost on the server</td>
<td>Completely customizable (via XML and XSLT); dynamically generated on the client, semantics are preserved</td>
</tr>
</tbody>
</table>

FIGURE 3
Enterprise Information Portals (EIPs)

<table>
<thead>
<tr>
<th>Enterprise Knowledge Portals (EKPs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge-Integrated E-Business Applications</td>
</tr>
<tr>
<td>Knowledge Web Application Services (Agents, Navigation, Connections, Discovery, Sharing)</td>
</tr>
<tr>
<td>Knowledge Logic</td>
</tr>
<tr>
<td>Knowledge Query Services</td>
</tr>
<tr>
<td>Knowledge Ontology/Inference Services</td>
</tr>
<tr>
<td>Knowledge Representation</td>
</tr>
<tr>
<td>Knowledge Indexing and Classification</td>
</tr>
<tr>
<td>Structured Information Sources</td>
</tr>
<tr>
<td>Unstructured Information Sources</td>
</tr>
</tbody>
</table>

Today's Web

PRIOR ART

FIGURE 7
Cross-Platform Protocol and Security Standards (XML, Interoperable XML Web Services (e.g., WS-I), WS-Security, etc.)

Cross-Platform Presentation Standards (XML, XSLT, SVG, etc.)

Cross-Platform Data Standards (XML)

Knowledge Integration Server

Open Data Storage (e.g., ODBC, JDBC) and Access (e.g., SQL) Standards (where possible)

XML, XML Web Services

Knowledge Base Server

Open Information Access Protocol Standards (where possible)

People DSA (LDAP, etc.)

Events DSA (iCalendar, etc.)

Email DSA (SMTP, IMAP, etc.)

Documents DSA (HTTP, etc.)

Data Repositories

FIGURE 8
Breaking News Agent: Documents on Reuters Related to [My Current User Interface Specification]

Object Title: [Email: Yuying's Thoughts on the Current User Interface]

Link: All Information Related to this Item

Last Item Posted On: May 17, 12:13 PM [Preview]

Next Event Start on: June 28, 09:00 [Preview]

Show only items in the following type:

Show only items posted in the last:

Found a Total of 50 Items [Preview]
### FIGURE 10-1

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectID</td>
<td>Number</td>
</tr>
<tr>
<td>AgentTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>AgentObjectTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>AgentName</td>
<td>Text</td>
</tr>
<tr>
<td>AgentDescription</td>
<td>Text</td>
</tr>
<tr>
<td>AgentDefaultSkinUrl</td>
<td>Text</td>
</tr>
<tr>
<td>AgentQueryText</td>
<td>Text</td>
</tr>
<tr>
<td>AgentQueryTypeID</td>
<td>Number</td>
</tr>
</tbody>
</table>

### FIGURE 10-2

<table>
<thead>
<tr>
<th>Agent Type ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGENTTYPEID_STANDARD</td>
<td>1</td>
</tr>
<tr>
<td>AGENTTYPEID_COMPOUND</td>
<td>2</td>
</tr>
<tr>
<td>AGENTTYPEID_BLENDER</td>
<td>3</td>
</tr>
<tr>
<td>AGENTTYPEID_SMART</td>
<td>4</td>
</tr>
<tr>
<td>AGENTTYPEID_SPECIAL</td>
<td>5</td>
</tr>
</tbody>
</table>

### FIGURE 10-3

<table>
<thead>
<tr>
<th>Agent Query Type ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGENTQUERYTYPEID_SQL</td>
<td>1</td>
</tr>
</tbody>
</table>

### FIGURE 10-4

<table>
<thead>
<tr>
<th>Agent Name</th>
<th>Agent Query Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>News.All</td>
<td>SELECT OBJECTID FROM NEWS</td>
</tr>
<tr>
<td>News.Technology.All</td>
<td>SELECT OBJECTID FROM NEWS WHERE CATEGORYID = &quot;Technology.All&quot;</td>
</tr>
<tr>
<td>Field Name</td>
<td>Type</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
</tr>
<tr>
<td>ObjectID</td>
<td>Number</td>
</tr>
<tr>
<td>ObjectTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>Name</td>
<td>Text</td>
</tr>
<tr>
<td>Title</td>
<td>Text</td>
</tr>
<tr>
<td>Description</td>
<td>Text</td>
</tr>
<tr>
<td>URL</td>
<td>Text</td>
</tr>
<tr>
<td>CreationTime</td>
<td>Date-Time</td>
</tr>
<tr>
<td>LastModifiedTime</td>
<td>Date-Time</td>
</tr>
<tr>
<td>LastAccessTime</td>
<td>Date-Time</td>
</tr>
<tr>
<td>MajorVersion</td>
<td>Number</td>
</tr>
<tr>
<td>MinorVersion</td>
<td>Number</td>
</tr>
<tr>
<td>Attributes</td>
<td>Number</td>
</tr>
<tr>
<td>SourceID</td>
<td>Number</td>
</tr>
<tr>
<td>Custom</td>
<td>Text</td>
</tr>
</tbody>
</table>

**FIGURE 13-1**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubjectID</td>
<td>Number</td>
</tr>
<tr>
<td>SubjectTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>PredicateTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>ObjectID</td>
<td>Number</td>
</tr>
<tr>
<td>ObjectTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>LinkScore</td>
<td>Number</td>
</tr>
<tr>
<td>ReferenceDate</td>
<td>Date-Time</td>
</tr>
</tbody>
</table>

**FIGURE 13-2**
<table>
<thead>
<tr>
<th>Predicate Type ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREDICATETYPEID_OFCATEGORY</td>
<td>1</td>
</tr>
<tr>
<td>PREDICATETYPEID_CREATOR</td>
<td>2</td>
</tr>
<tr>
<td>PREDICATETYPEID_ORGANIZER</td>
<td>3</td>
</tr>
<tr>
<td>PREDICATETYPEID_ATTENDEE</td>
<td>4</td>
</tr>
<tr>
<td>PREDICATETYPEID_CONTRIBUTOR</td>
<td>5</td>
</tr>
<tr>
<td>PREDICATETYPEID_APPROVER</td>
<td>6</td>
</tr>
<tr>
<td>PREDICATETYPEID_ATTACHEDTO</td>
<td>7</td>
</tr>
<tr>
<td>PREDICATETYPEID_SENTTO</td>
<td>8</td>
</tr>
<tr>
<td>PREDICATETYPEID_COPIEDTO</td>
<td>9</td>
</tr>
<tr>
<td>PREDICATETYPEID_BLINDCOPIEDTO</td>
<td>10</td>
</tr>
<tr>
<td>PREDICATETYPEID_REFERENCES</td>
<td>11</td>
</tr>
<tr>
<td>PREDICATETYPEID_SIMILARTO</td>
<td>12</td>
</tr>
<tr>
<td>PREDICATETYPEID_RELATEDTO</td>
<td>13</td>
</tr>
<tr>
<td>PREDICATETYPEID_EXPERTON</td>
<td>14</td>
</tr>
<tr>
<td>PREDICATETYPEID_REPORTSTO</td>
<td>15</td>
</tr>
<tr>
<td>PREDICATETYPEID_MEMBEROF</td>
<td>16</td>
</tr>
<tr>
<td>PREDICATETYPEID_INTERESTEDIN</td>
<td>17</td>
</tr>
<tr>
<td>PREDICATETYPEID_LIKELYTOBEINTERESTEDIN</td>
<td>18</td>
</tr>
<tr>
<td>PREDICATETYPEID_CONTAINSTAG</td>
<td>19</td>
</tr>
<tr>
<td>PREDICATETYPEID_CONTAINS</td>
<td>20</td>
</tr>
<tr>
<td>PREDICATETYPEID_ANNOTATES</td>
<td>21</td>
</tr>
</tbody>
</table>

**FIGURE 14**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>FullName</td>
<td>Text</td>
</tr>
<tr>
<td>JobTitle</td>
<td>Text</td>
</tr>
<tr>
<td>Institution</td>
<td>Text</td>
</tr>
<tr>
<td>BusinessPhone</td>
<td>Text</td>
</tr>
<tr>
<td>HomePhone</td>
<td>Text</td>
</tr>
<tr>
<td>BusinessFax</td>
<td>Text</td>
</tr>
<tr>
<td>MobilePhone</td>
<td>Text</td>
</tr>
<tr>
<td>Email</td>
<td>Text</td>
</tr>
<tr>
<td>DisplayAs</td>
<td>Text</td>
</tr>
<tr>
<td>HomeAddress</td>
<td>Text</td>
</tr>
<tr>
<td>BusinessAddress</td>
<td>Text</td>
</tr>
<tr>
<td>OtherAddress</td>
<td>Text</td>
</tr>
<tr>
<td>MailingAddressTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>WebPageAddress</td>
<td>Text</td>
</tr>
<tr>
<td>IMAddress</td>
<td>Text</td>
</tr>
<tr>
<td>OtherPhone</td>
<td>Date-Time</td>
</tr>
<tr>
<td>CarPhone</td>
<td>Date-Time</td>
</tr>
</tbody>
</table>

**FIGURE 15**
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>OtherFax</td>
<td>Text</td>
</tr>
<tr>
<td>Pager</td>
<td>Number</td>
</tr>
<tr>
<td>BusinessPhone2</td>
<td>Text</td>
</tr>
<tr>
<td>AssistantPhone</td>
<td>Text</td>
</tr>
<tr>
<td>CallbackPhone</td>
<td>Text</td>
</tr>
<tr>
<td>InstitutionPhone</td>
<td>Text</td>
</tr>
<tr>
<td>HomePhone2</td>
<td>Text</td>
</tr>
<tr>
<td>HomeFax</td>
<td>Text</td>
</tr>
<tr>
<td>ISDNPhone</td>
<td>Text</td>
</tr>
<tr>
<td>PrimaryPhone</td>
<td>Text</td>
</tr>
<tr>
<td>RadioPhone</td>
<td>Text</td>
</tr>
<tr>
<td>TelexPhone</td>
<td>Text</td>
</tr>
<tr>
<td>TTYPhone</td>
<td>Text</td>
</tr>
<tr>
<td>Department</td>
<td>Text</td>
</tr>
<tr>
<td>ManagerName</td>
<td>Text</td>
</tr>
<tr>
<td>Office</td>
<td>Text</td>
</tr>
<tr>
<td>AssistantName</td>
<td>Text</td>
</tr>
<tr>
<td>Profession</td>
<td>Text</td>
</tr>
<tr>
<td>Nickname</td>
<td>Text</td>
</tr>
<tr>
<td>SpouseName</td>
<td>Text</td>
</tr>
<tr>
<td>Birthday</td>
<td>Date</td>
</tr>
<tr>
<td>Anniversary</td>
<td>Date</td>
</tr>
<tr>
<td>OnlineConferenceServer</td>
<td>Text</td>
</tr>
<tr>
<td>OnlineConferenceEmailAlias</td>
<td>Text</td>
</tr>
<tr>
<td>InternetFreeBusyAddress</td>
<td>Text</td>
</tr>
<tr>
<td>Comments</td>
<td>Text</td>
</tr>
</tbody>
</table>

FIGURE 15  (continued)
### FIGURE 16-1

<table>
<thead>
<tr>
<th>Mailing Address Type ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAILINGADDRESSTYPEID_HOME</td>
<td>1</td>
</tr>
<tr>
<td>MAILINGADDRESSTYPEID_BUSINESS</td>
<td>2</td>
</tr>
<tr>
<td>MAILINGADDRESSTYPEID_OTHER</td>
<td>3</td>
</tr>
</tbody>
</table>

### FIGURE 16-2

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CategoryUrl</td>
<td>Text</td>
</tr>
<tr>
<td>CategoryName</td>
<td>Text</td>
</tr>
<tr>
<td>CategoryDescription</td>
<td>Text</td>
</tr>
<tr>
<td>KBS Url</td>
<td>Text</td>
</tr>
<tr>
<td>Semantic Domain Name</td>
<td>Text</td>
</tr>
</tbody>
</table>

### FIGURE 16-3

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Text</td>
</tr>
<tr>
<td>Subject</td>
<td>Text</td>
</tr>
<tr>
<td>Author</td>
<td>Text</td>
</tr>
<tr>
<td>DocumentFormatTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>DocumentFormatVersion</td>
<td>Number</td>
</tr>
<tr>
<td>Manager</td>
<td>Text</td>
</tr>
<tr>
<td>InstitutionID</td>
<td>Number</td>
</tr>
<tr>
<td>DocumentCategory</td>
<td>Text</td>
</tr>
<tr>
<td>Keywords</td>
<td>Text</td>
</tr>
<tr>
<td>Summary</td>
<td>Text</td>
</tr>
<tr>
<td>Comments</td>
<td>Text</td>
</tr>
<tr>
<td>NumPages</td>
<td>Number</td>
</tr>
<tr>
<td>NumParagraphs</td>
<td>Number</td>
</tr>
<tr>
<td>NumLines</td>
<td>Number</td>
</tr>
<tr>
<td>NumWords</td>
<td>Number</td>
</tr>
<tr>
<td>CreationTime</td>
<td>Date-Time</td>
</tr>
<tr>
<td>LastModifiedTime</td>
<td>Date-Time</td>
</tr>
<tr>
<td>LastAccessedTime</td>
<td>Date-Time</td>
</tr>
<tr>
<td>LastSavedBy</td>
<td>Text</td>
</tr>
<tr>
<td>LanguageID</td>
<td>Number</td>
</tr>
<tr>
<td>SourceUrl</td>
<td>Text</td>
</tr>
</tbody>
</table>
### FIGURE 17-1

<table>
<thead>
<tr>
<th>Print Media Type ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOCUMENTTYPEID_ANALYSTBRIEF</td>
<td>1</td>
</tr>
<tr>
<td>DOCUMENTTYPEID_ANALYSTREPORT</td>
<td>2</td>
</tr>
<tr>
<td>DOCUMENTTYPEID_CASESTUDY</td>
<td>3</td>
</tr>
<tr>
<td>DOCUMENTTYPEID_WHITEPAPER</td>
<td>4</td>
</tr>
<tr>
<td>DOCUMENTTYPEID_EBOOK</td>
<td>5</td>
</tr>
<tr>
<td>DOCUMENTTYPEID_EMAGAZINE</td>
<td>6</td>
</tr>
<tr>
<td>DOCUMENTTYPEID_NEWSARTICLE</td>
<td>7</td>
</tr>
<tr>
<td>DOCUMENTTYPEID_COMPANYPROFILE</td>
<td>8</td>
</tr>
</tbody>
</table>

### FIGURE 17-2

<table>
<thead>
<tr>
<th>Document Type ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMATTYPEID_TEXT</td>
<td>1</td>
</tr>
<tr>
<td>FORMATTYPEID_HTML</td>
<td>2</td>
</tr>
<tr>
<td>FORMATTYPEID_ADOBEbuquerque</td>
<td>3</td>
</tr>
<tr>
<td>FORMATTYPEID_MSWORD</td>
<td>4</td>
</tr>
<tr>
<td>FORMATTYPEID_MSEXCEL</td>
<td>5</td>
</tr>
<tr>
<td>FORMATTYPEID_MSPowerPoint</td>
<td>6</td>
</tr>
<tr>
<td>FORMATTYPEID_MSACCESS</td>
<td>7</td>
</tr>
<tr>
<td>FORMATTYPEID_MSPROJECT</td>
<td>8</td>
</tr>
<tr>
<td>FORMATTYPEID_MP3</td>
<td>9</td>
</tr>
<tr>
<td>FORMATTYPEID_WMA</td>
<td>10</td>
</tr>
<tr>
<td>FORMATTYPEID_RMA</td>
<td>11</td>
</tr>
</tbody>
</table>
### Email Message Object Schema

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>Text</td>
</tr>
<tr>
<td>To</td>
<td>Text</td>
</tr>
<tr>
<td>Cc</td>
<td>Text</td>
</tr>
<tr>
<td>Bcc</td>
<td>Text</td>
</tr>
<tr>
<td>Subject</td>
<td>Text</td>
</tr>
<tr>
<td>EmailTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>ReceivedTime</td>
<td>Time</td>
</tr>
<tr>
<td>Size</td>
<td>Number</td>
</tr>
<tr>
<td>Priority</td>
<td>Number</td>
</tr>
<tr>
<td>HeaderStatus</td>
<td>Number</td>
</tr>
<tr>
<td>FlagStatus</td>
<td>Bool</td>
</tr>
<tr>
<td>ContainsAttachment</td>
<td>Bool</td>
</tr>
<tr>
<td>Sent</td>
<td>Bool</td>
</tr>
<tr>
<td>Read</td>
<td>Bool</td>
</tr>
<tr>
<td>FollowUpFlag</td>
<td>Bool</td>
</tr>
<tr>
<td>CreatedTime</td>
<td>Date-Time</td>
</tr>
<tr>
<td>SentTime</td>
<td>Date-Time</td>
</tr>
<tr>
<td>Attachments</td>
<td>Text</td>
</tr>
<tr>
<td>SourceUrl</td>
<td>Text</td>
</tr>
</tbody>
</table>

**FIGURE 18-1**

### Email Distribution List Object Schema

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DistributionListName</td>
<td>Text</td>
</tr>
<tr>
<td>DistributionListTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>Owner</td>
<td>Text</td>
</tr>
<tr>
<td>PublicAccess</td>
<td>Bool</td>
</tr>
<tr>
<td>SourceUrl</td>
<td>Text</td>
</tr>
</tbody>
</table>

**FIGURE 18-2**

### Email Public Folder Object Schema

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PublicFolderName</td>
<td>Text</td>
</tr>
<tr>
<td>PublicFolderTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>Owner</td>
<td>Text</td>
</tr>
<tr>
<td>PublicAccess</td>
<td>Bool</td>
</tr>
<tr>
<td>SourceUrl</td>
<td>Text</td>
</tr>
</tbody>
</table>

**FIGURE 18-3**
<table>
<thead>
<tr>
<th>Distribution List Type ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLICFOLDERTYPEID_DISTRIBUTIONLIST</td>
<td>1</td>
</tr>
<tr>
<td>PUBLICFOLDERTYPEID_PUBLICFOLDER</td>
<td>2</td>
</tr>
<tr>
<td>PUBLICFOLDERTYPEID_NEWSGROUP</td>
<td>3</td>
</tr>
</tbody>
</table>

FIGURE 19-1

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventTitle</td>
<td>Text</td>
</tr>
<tr>
<td>EventID</td>
<td>Number</td>
</tr>
<tr>
<td>EventTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>Status</td>
<td>Text</td>
</tr>
<tr>
<td>Location</td>
<td>Text</td>
</tr>
<tr>
<td>StartDate</td>
<td>Date</td>
</tr>
<tr>
<td>StartTime</td>
<td>Time</td>
</tr>
<tr>
<td>EndDate</td>
<td>Date</td>
</tr>
<tr>
<td>EndTime</td>
<td>Time</td>
</tr>
<tr>
<td>RequiredStaffing</td>
<td>Number</td>
</tr>
<tr>
<td>Confirmed</td>
<td>Bool</td>
</tr>
<tr>
<td>AvailableSpaces</td>
<td>Number</td>
</tr>
<tr>
<td>CostPerPerson</td>
<td>Number</td>
</tr>
<tr>
<td>CostCurrencyID</td>
<td>Number</td>
</tr>
<tr>
<td>EventDescription</td>
<td>Text</td>
</tr>
<tr>
<td>Notes</td>
<td>Text</td>
</tr>
<tr>
<td>Organization</td>
<td>Text</td>
</tr>
<tr>
<td>EventOrganizer</td>
<td>Text</td>
</tr>
<tr>
<td>SourceUrl</td>
<td>Text</td>
</tr>
</tbody>
</table>

FIGURE 19-2

<table>
<thead>
<tr>
<th>Event Type ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENTTYPEID_MEETING</td>
<td>1</td>
</tr>
<tr>
<td>EVENTTYPEID_CORPORATEEVENT</td>
<td>2</td>
</tr>
<tr>
<td>EVENTTYPEID_INDUSTRYEVENT</td>
<td>3</td>
</tr>
<tr>
<td>EVENTTYPEID_TVPROGRAM</td>
<td>4</td>
</tr>
<tr>
<td>EVENTTYPEID_RADIOEVENT</td>
<td>5</td>
</tr>
<tr>
<td>EVENTTYPEID_PRINTMEDIAPROGRAM</td>
<td>6</td>
</tr>
<tr>
<td>EVENTTYPEID_ONLIMEVENT</td>
<td>7</td>
</tr>
<tr>
<td>EVENTTYPEID_ARTSANDENTERTAINMENT</td>
<td>8</td>
</tr>
</tbody>
</table>

FIGURE 19-3
### Media Object Schema

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Text</td>
</tr>
<tr>
<td>PublishedDate</td>
<td>Date</td>
</tr>
<tr>
<td>PublisherInstitutionID</td>
<td>Number</td>
</tr>
<tr>
<td>Language</td>
<td>Number</td>
</tr>
<tr>
<td>FormatTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>FormatVersion</td>
<td>Number</td>
</tr>
<tr>
<td>ReportTypeID</td>
<td>Number</td>
</tr>
<tr>
<td>NumPages</td>
<td>Number</td>
</tr>
<tr>
<td>Abstract</td>
<td>Text</td>
</tr>
<tr>
<td>Author</td>
<td>Text</td>
</tr>
<tr>
<td>ProvidedBy</td>
<td>Text</td>
</tr>
<tr>
<td>SourceUrl</td>
<td>Text</td>
</tr>
<tr>
<td>Edition</td>
<td>Number</td>
</tr>
<tr>
<td>ImageUrl</td>
<td>Text</td>
</tr>
<tr>
<td>PurchaseUrl</td>
<td>Text</td>
</tr>
<tr>
<td>RightsProtectedFlag</td>
<td>Bool</td>
</tr>
</tbody>
</table>

**FIGURE 20-1**

<table>
<thead>
<tr>
<th>Print Media Type ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEDIATYPEID_BOOK</td>
<td>1</td>
</tr>
<tr>
<td>MEDIATYPEID_MAGAZINE</td>
<td>2</td>
</tr>
</tbody>
</table>

**FIGURE 20-2**
<table>
<thead>
<tr>
<th>Container Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Information</td>
<td>Root object type—all object types inherit from this type</td>
</tr>
<tr>
<td>Blenders</td>
<td>Complex object type—this is a composite type that includes objects of any object type. Blenders can also include other blenders.</td>
</tr>
<tr>
<td>Documents</td>
<td>Documents</td>
</tr>
<tr>
<td>Email</td>
<td>Email Messages</td>
</tr>
<tr>
<td>Email Distribution Lists</td>
<td>Email Distribution Lists</td>
</tr>
<tr>
<td>Email Public Folders</td>
<td>Email Public Folders</td>
</tr>
<tr>
<td>Events</td>
<td>Events</td>
</tr>
<tr>
<td>News Articles</td>
<td>News Articles (e.g., Reuters publications)</td>
</tr>
<tr>
<td>Internet News Postings</td>
<td>Internet News (NNTP) Postings—from newsgroups</td>
</tr>
<tr>
<td>Online Courses</td>
<td>E-Learning courses (from LMS systems)—e.g., Saba, Docent, etc.</td>
</tr>
<tr>
<td>Media</td>
<td>Media—e.g., books and magazines</td>
</tr>
<tr>
<td>Multimedia</td>
<td>Multimedia</td>
</tr>
<tr>
<td>People</td>
<td>People</td>
</tr>
</tbody>
</table>

FIGURE 21
<table>
<thead>
<tr>
<th>Container Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Information</td>
</tr>
<tr>
<td>Blenders</td>
</tr>
<tr>
<td>Documents\Analyst Brief</td>
</tr>
<tr>
<td>Documents\Analyst Reports</td>
</tr>
<tr>
<td>Documents\Case Studies</td>
</tr>
<tr>
<td>Documents\Company Profiles</td>
</tr>
<tr>
<td>Documents\Documents</td>
</tr>
<tr>
<td>Documents\E-Books</td>
</tr>
<tr>
<td>Documents\E-Magazines</td>
</tr>
<tr>
<td>Documents\White Papers</td>
</tr>
<tr>
<td>Email\Customer Email</td>
</tr>
<tr>
<td>Email\Email Annotations</td>
</tr>
<tr>
<td>Email\Email Messages</td>
</tr>
<tr>
<td>Email\Email Postings</td>
</tr>
<tr>
<td>Email\Email Distribution Lists</td>
</tr>
<tr>
<td>Email Public Folders</td>
</tr>
<tr>
<td>Events\Arts and Entertainment Events</td>
</tr>
<tr>
<td>Events\Corporate Events</td>
</tr>
<tr>
<td>Events\General Events</td>
</tr>
<tr>
<td>Events\Industry Events</td>
</tr>
<tr>
<td>Events\Meetings</td>
</tr>
<tr>
<td>Events\Online Meetings</td>
</tr>
<tr>
<td>Events\Print Media Events</td>
</tr>
<tr>
<td>Events\Radio Events</td>
</tr>
<tr>
<td>Events\TV Events</td>
</tr>
<tr>
<td>News Articles</td>
</tr>
<tr>
<td>Internet News Postings</td>
</tr>
<tr>
<td>Online Courses</td>
</tr>
<tr>
<td>Media\Books</td>
</tr>
<tr>
<td>Media\Magazines</td>
</tr>
<tr>
<td>Multimedia</td>
</tr>
<tr>
<td>People\Users</td>
</tr>
<tr>
<td>People\Customers</td>
</tr>
</tbody>
</table>

**FIGURE 22**
<table>
<thead>
<tr>
<th>Container Type</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Information</td>
<td>Related To</td>
</tr>
<tr>
<td>All Information</td>
<td>Possibly Related To</td>
</tr>
<tr>
<td>All Information</td>
<td>Authored By a Person with the Name</td>
</tr>
<tr>
<td>All Information</td>
<td>Authored By a Person with a Name Containing the Text</td>
</tr>
<tr>
<td>All Information</td>
<td>Authored By a Person with the Email Address</td>
</tr>
<tr>
<td>All Information</td>
<td>Likely Authored By a Person with the Name</td>
</tr>
<tr>
<td>All Information</td>
<td>Likely Authored By a Person with the Email Address</td>
</tr>
<tr>
<td>All Information</td>
<td>Annotated By a Person with the Name</td>
</tr>
<tr>
<td>All Information</td>
<td>Annotated By a Person with a Name Containing the Text</td>
</tr>
<tr>
<td>All Information</td>
<td>Annotated By a Person with the Email Address</td>
</tr>
<tr>
<td>All Information</td>
<td>With Annotations Belonging to the Category</td>
</tr>
<tr>
<td>All Information</td>
<td>Belonging to the Category</td>
</tr>
<tr>
<td>All Information</td>
<td>Belonging to Categories in which a Person with the following Name is an Expert</td>
</tr>
<tr>
<td>All Information</td>
<td>Belonging to Categories in which a Person with the Name Containing the Following Text is an Expert</td>
</tr>
<tr>
<td>All Information</td>
<td>Belonging to Categories in which a Person with the following Email Address is an Expert</td>
</tr>
<tr>
<td>All Information</td>
<td>With Annotations Belonging to the Category</td>
</tr>
<tr>
<td>All Information</td>
<td>Annotations Belonging to Categories in which a Person with the following Name is an Expert</td>
</tr>
<tr>
<td>All Information</td>
<td>Annotations Belonging to Categories in which a Person with a Name Containing the following Text is an Expert</td>
</tr>
<tr>
<td>All Information</td>
<td>With Annotations Belonging to Categories in which a Person with the following Email Address is an Expert</td>
</tr>
<tr>
<td>All Information</td>
<td>Created On</td>
</tr>
<tr>
<td>All Information</td>
<td>Created Before</td>
</tr>
<tr>
<td>All Information</td>
<td>Created After</td>
</tr>
<tr>
<td>All Information</td>
<td>Posted On</td>
</tr>
<tr>
<td>All Information</td>
<td>Posted Before</td>
</tr>
<tr>
<td>All Information</td>
<td>Posted After</td>
</tr>
<tr>
<td>All Information</td>
<td>Posted in the Last</td>
</tr>
<tr>
<td>All Information</td>
<td>With a Body or Title Containing the Text</td>
</tr>
<tr>
<td>All Information</td>
<td>With a Body Containing the Text</td>
</tr>
<tr>
<td>All Information</td>
<td>With a Title Containing the Text</td>
</tr>
<tr>
<td>All Information</td>
<td>With a Body or Title with the Exact Text</td>
</tr>
<tr>
<td>All Information</td>
<td>With a Body with the Exact Text</td>
</tr>
<tr>
<td>All Information</td>
<td>With a Title with the Exact Text</td>
</tr>
<tr>
<td>Email</td>
<td>Copied to a Person with the Name</td>
</tr>
<tr>
<td>Email</td>
<td>Copied to a Person with a Name Containing the Text</td>
</tr>
<tr>
<td>Email</td>
<td>Copied to a Person with the Email Address</td>
</tr>
</tbody>
</table>

FIGURE 23
<table>
<thead>
<tr>
<th>Container Type</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>Blind Copied to a Person with the Name</td>
</tr>
<tr>
<td>Email</td>
<td>Blind Copied To a Person with a Name Containing the Text</td>
</tr>
<tr>
<td>Email</td>
<td>Blind Copied to a Person with the Email Address</td>
</tr>
<tr>
<td>Email</td>
<td>Of the following Priority</td>
</tr>
<tr>
<td>Email</td>
<td>Containing Attachments Belonging to the Category</td>
</tr>
<tr>
<td>Email Distribution List</td>
<td>Containing the Person(s) with the Name</td>
</tr>
<tr>
<td>Email Distribution List</td>
<td>Containing the Person(s) with a Name Containing the Text</td>
</tr>
<tr>
<td>Email Distribution List</td>
<td>Containing the Person with the Email Address</td>
</tr>
<tr>
<td>Event</td>
<td>Starting On</td>
</tr>
<tr>
<td>Event</td>
<td>That Started in the Last</td>
</tr>
<tr>
<td>Event</td>
<td>Starting within the Next</td>
</tr>
<tr>
<td>Event</td>
<td>Ending On</td>
</tr>
<tr>
<td>Event</td>
<td>That Ended in the Last</td>
</tr>
<tr>
<td>Event</td>
<td>Ending within the Next</td>
</tr>
<tr>
<td>Event</td>
<td>Recurring Every</td>
</tr>
<tr>
<td>Event</td>
<td>Located At a Place with the Name</td>
</tr>
<tr>
<td>Event</td>
<td>Located At a Place with a Name Containing the Text</td>
</tr>
<tr>
<td>Event</td>
<td>Organized by a Person with the Name</td>
</tr>
<tr>
<td>Event</td>
<td>Organized by a Person with a Name Containing the Text</td>
</tr>
<tr>
<td>Event</td>
<td>Organized By a Person with the Email Address</td>
</tr>
<tr>
<td>Event</td>
<td>Attended By a Person with the Name</td>
</tr>
<tr>
<td>Event</td>
<td>Attended By a Person with a Name Containing the Text</td>
</tr>
<tr>
<td>Event</td>
<td>Attended By a Person with the Email Address</td>
</tr>
<tr>
<td>People</td>
<td>With the Name</td>
</tr>
<tr>
<td>People</td>
<td>With a Name Containing the Text</td>
</tr>
<tr>
<td>People</td>
<td>With the Email Address</td>
</tr>
<tr>
<td>Customers</td>
<td>With the Name</td>
</tr>
<tr>
<td>Customers</td>
<td>With a Name Containing the Text</td>
</tr>
<tr>
<td>Customers</td>
<td>With the Email Address</td>
</tr>
</tbody>
</table>
EMAIL OBJECT:
<email from="john@foo.com"
    to="nosa@nervana.net"
    cc="steve@nervana.net"
    bcc="patrick@nervana.net"
    subject="meeting this Friday"
    body="let us meet on Friday at 2pm"
    attachments="c:\foo.doc; c:\bar.htm">
</email>

USER OBJECT:
<html>
  <head></head>
  <body>
  </body>
</html>

DOCUMENT OBJECT:
<html>
  <head></head>
  <body>
  </body>
</html>
FIG. 25

FIG. 26
<table>
<thead>
<tr>
<th>OPEN</th>
<th>COPY</th>
<th>ANNOTATE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TYPE-SPECIFIC VERBS (e.g., &quot;ADD TO CALENDAR&quot;)</td>
</tr>
</tbody>
</table>

**FIG. 27**

**INFO NUGGET TEMPLATE: [TEMPLATE]**

- STEVE JUDKINS IS MOST LIKELY TO HAVE AUTHORED THIS DOCUMENT [PREVIEW]
- THIS DOCUMENT IS OF CATEGORY: [CATEGORY]
- THIS DOCUMENT HAS 3 LIKELY EXPERTS: [PREVIEW]
- STEVE JUDKINS HAS 6 DIRECT REPORT [PREVIEW]
- STEVE JUDKINS HAS POSTED A TOTAL OF 500 ITEMS ON THIS AGENCY [PREVIEW]
- STEVE JUDKINS HAS POSTED A TOTAL OF 45 RELATED ITEMS ON THIS AGENCY [PREVIEW]
- STEVE JUDKINS IS AN EXPERT ON 78 DOCUMENTS ON THIS AGENCY [PREVIEW]
- STEVE JUDKINS IS HOSTING 12 UPCOMING EVENTS [PREVIEW]

**FIG. 28**
<table>
<thead>
<tr>
<th>Agent Lens: Documents on Reuters Related to [My Nervana UI Specification]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Title: [Email: Yuying's Thoughts on the Nervana UI]</td>
</tr>
<tr>
<td>Link: All Information Related to this Item</td>
</tr>
<tr>
<td>Last Item Posted On: May 17, 12:14PM [Preview]</td>
</tr>
<tr>
<td>Next Event Starting On: June 28, 09:00AM [Preview]</td>
</tr>
<tr>
<td>Show only items of the following type:</td>
</tr>
<tr>
<td>Show only items posted in the last:</td>
</tr>
<tr>
<td>Found a Total of 50 Items [Preview]</td>
</tr>
</tbody>
</table>

FIGURE 29
<table>
<thead>
<tr>
<th>Object Lens Title: [My Nervana UI Specification]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent Title: Documents on Reuters Related to [My Nervana UI Specification]</td>
</tr>
<tr>
<td>Link: All Information Related to this Item</td>
</tr>
<tr>
<td>Last item Posted On: May 17, 12:14PM [Preview]</td>
</tr>
<tr>
<td>Next Event Starting On: June 28, 09:00AM [Preview]</td>
</tr>
<tr>
<td>Show only items of the following type:</td>
</tr>
<tr>
<td>Show only items posted in the last:</td>
</tr>
</tbody>
</table>

**Found a Total of 50 Items** [Preview]

**FIGURE 30**
FIG. 31

OBJECT LENS TITLE: [MY NERVANA UI SPECIFICATION]

AGENT TITLE: DOCUMENTS ON REUTERS RELATED TO [MY NERVANA UI

LINK: ALL INFORMATION RELATED TO THIS ITEM

LAST ITEM POSTED ON: MAY 17, 12:14PM [PREVIEW]

NEXT EVENT STARTING ON: JUNE 28, 09:00AM [PREVIEW]

SHOW ONLY ITEMS OF THE FOLLOWING TYPE:

SHOW ONLY ITEMS POSTED IN THE LAST:

FOUND A TOTAL OF 50 ITEMS [PREVIEW]

FIG. 32

OBJECT LENS TITLE: [MY NERVANA UI SPECIFICATION]

OBJECT TITLE: [EMAIL: YUYING'S THOUGHTS ON THE NERVANA UI]

LINK: RELATEDNESS MEASURE ACCORDING TO AGENCY [AGENCY NAME]

RELATEDNESS MEASURE: 93%

0% 100%
<table>
<thead>
<tr>
<th>Agent Lens Type</th>
<th>Object Type</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Information</td>
<td>Document</td>
<td>Related Objects on Agent, Possibly Related Objects on Agent</td>
</tr>
<tr>
<td>All Information</td>
<td>Email</td>
<td>Related Objects on Agent, Possibly Related Objects on Agent</td>
</tr>
<tr>
<td>All Information</td>
<td>Email Distribution List</td>
<td>None</td>
</tr>
<tr>
<td>All Information</td>
<td>Email Public Folder</td>
<td>None</td>
</tr>
<tr>
<td>All Information</td>
<td>Event</td>
<td>Related Objects on Agent, Possibly Related Objects on Agent</td>
</tr>
<tr>
<td>All Information</td>
<td>News Article</td>
<td>Related Objects on Agent, Possibly Related Objects on Agent</td>
</tr>
<tr>
<td>All Information</td>
<td>Internet News Message</td>
<td>Related Objects on Agent, Possibly Related Objects on Agent</td>
</tr>
<tr>
<td>All Information</td>
<td>Online Course</td>
<td>Related Objects on Agent, Possibly Related Objects on Agent</td>
</tr>
<tr>
<td>All Information</td>
<td>Media</td>
<td>Related Objects on Agent, Possibly Related Objects on Agent</td>
</tr>
<tr>
<td>All Information</td>
<td>Multimedia</td>
<td>Related Objects on Agent, Possibly Related Objects on Agent</td>
</tr>
<tr>
<td>All Information</td>
<td>Person</td>
<td>Objects on Agent authored by the person, objects on agent possibly authored by the person, objects on agent annotated by the person, objects on agent on which the person is an expert</td>
</tr>
<tr>
<td>All Information</td>
<td>Customer</td>
<td>Objects on Agent authored by the person, objects on agent possibly authored by the person</td>
</tr>
</tbody>
</table>

FIGURE 33
<table>
<thead>
<tr>
<th>Object Lens Type</th>
<th>Object Type</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document</td>
<td>Document</td>
<td>Similarity Percentage, Similarity Bar Graph</td>
</tr>
<tr>
<td>Document</td>
<td>Email</td>
<td>Similarity Percentage, Similarity Bar Graph</td>
</tr>
<tr>
<td>Document</td>
<td>Email Distribution List</td>
<td>Members that authored the document, Members that might have authored the document, Members that are experts on the document, Whether the document has ever been posted to the distribution list</td>
</tr>
<tr>
<td>Document</td>
<td>Email Public Folder</td>
<td>Email messages in the public folder that relate to this document, Email messages in the public folder that might relate to this document</td>
</tr>
<tr>
<td>Document</td>
<td>Event</td>
<td>Similarity Percentage, Similarity Bar</td>
</tr>
<tr>
<td>Document</td>
<td>News Article</td>
<td>Similarity Percentage, Similarity Bar</td>
</tr>
<tr>
<td>Document</td>
<td>Internet News Message</td>
<td>Similarity Percentage, Similarity Bar</td>
</tr>
<tr>
<td>Document</td>
<td>Online Course</td>
<td>Similarity Percentage, Similarity Bar</td>
</tr>
<tr>
<td>Document</td>
<td>Media</td>
<td>Similarity Percentage, Similarity Bar</td>
</tr>
<tr>
<td>Document</td>
<td>Multimedia</td>
<td>Similarity Percentage, Similarity Bar</td>
</tr>
<tr>
<td>Document</td>
<td>Person</td>
<td>Possible Author? (True/False/%), Expert? (True/False/%), Possible Interest In (True/False/%), Related objects authored by the author</td>
</tr>
<tr>
<td>Document</td>
<td>Customer</td>
<td>Related email messages sent by the customer, Possible Interest In (True/False/%)</td>
</tr>
</tbody>
</table>

FIGURE 34
<table>
<thead>
<tr>
<th>Lens Type</th>
<th>Object Type</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>Email</td>
<td>Similarity Percentage, Similarity Bar</td>
</tr>
<tr>
<td>Email</td>
<td>Email Distribution List</td>
<td>Members that authored the message, members that might have authored the message, Members that are experts on the content of the message</td>
</tr>
<tr>
<td>Email</td>
<td>Email Public Folder</td>
<td>Is it in the Public Folder? (True/False), Email messages in the public folder that relate to this message, Email messages in the public folder that might relate to this message</td>
</tr>
<tr>
<td>Email</td>
<td>Event</td>
<td>TODO</td>
</tr>
<tr>
<td>Email</td>
<td>News Article</td>
<td>TODO</td>
</tr>
<tr>
<td>Email</td>
<td>Internet News Message</td>
<td>TODO</td>
</tr>
<tr>
<td>Email</td>
<td>Online Course</td>
<td>TODO</td>
</tr>
<tr>
<td>Email</td>
<td>Media</td>
<td>TODO</td>
</tr>
<tr>
<td>Email</td>
<td>Multimedia</td>
<td>TODO</td>
</tr>
<tr>
<td>Email</td>
<td>Person</td>
<td>TODO</td>
</tr>
<tr>
<td>Email</td>
<td>Customer</td>
<td>TODO</td>
</tr>
</tbody>
</table>

**FIGURE 35**
FIGURE 36
Agent 1

Agent 2

Agent 3

Agent 4

Agent 5

Agent 6

Agent 7

Agent 8

Agent Mini Player Control

Master Blender Player Control

FIGURE 37
<table>
<thead>
<tr>
<th>Tool</th>
<th>Regular Web Browser</th>
<th>Information Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back</td>
<td>Navigates to the previous web page</td>
<td>Navigates to the previous agent</td>
</tr>
<tr>
<td>Forward</td>
<td>Navigates to the next web page</td>
<td>Navigates to the next agent</td>
</tr>
<tr>
<td>Home</td>
<td>Navigates to the &quot;home page&quot;</td>
<td>Navigates to the &quot;home agent&quot;</td>
</tr>
<tr>
<td>Stop</td>
<td>Stops the current operation</td>
<td>Stops the current operation (or semantic query request)</td>
</tr>
<tr>
<td>Refresh</td>
<td>Refreshes the current Web page</td>
<td>Refreshes the current semantic query and also refreshes the semantic environment (and UI and the tree view reflecting this)</td>
</tr>
<tr>
<td>Print</td>
<td>Prints the current Web page</td>
<td>Prints the currently displayed information objects (semantic query results)</td>
</tr>
</tbody>
</table>

FIGURE 38
<table>
<thead>
<tr>
<th>Tool</th>
<th>File System Explorer or Document Viewer</th>
<th>Semantic Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>Creates a new file (via its application) or folder</td>
<td>Creates a new smart agent (via the &quot;Create Smart Agent&quot; wizard or a new blender (via the &quot;Create Blender&quot; wizard)</td>
</tr>
<tr>
<td>Open</td>
<td>Opens the selected file (via its application) or folder</td>
<td>Opens an agent (via the &quot;Open Agent&quot; dialog), file-system documents (to import them into the semantic environment and make them smart and context/time-aware), searches for local file-system documents (ditto), opens a URL (like an Internet HTTP or FTP address) (ditto), or opens what object that is currently on the clipboard (ditto)</td>
</tr>
<tr>
<td>Open</td>
<td>Opens a dialog box that allows the user to browse his/her file-system environment in order to open a desired document or just to browse the environment</td>
<td>Opens a dialog box that allows the user to browse his/her semantic environment in order to open a desired agent or just to browse the environment</td>
</tr>
<tr>
<td>Save</td>
<td>Saves the current document</td>
<td>Saves the current agent into the &quot;favorites&quot; list (if not already a favorite) or copies the current agent into a blender</td>
</tr>
</tbody>
</table>

FIGURE 39
<table>
<thead>
<tr>
<th>Tool</th>
<th>File System Explorer or Document Viewer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>Copies the currently selected file, folder, or textual/graphical/link information to the clipboard so it can be pasted to a different folder (in the case of files and/or folders) or pasted to the same or a new document (in the case of textual/graphical/link information)</td>
</tr>
<tr>
<td></td>
<td>From the toolbar of the semantic browser, if a Smart Agent is currently selected or opened, the SQML for the Agent is copied to the clipboard using the proprietary SGML clipboard format. This enables the smart lens functionality – the user can then select any information object in the Presenter window and select the “paste as smart lens” icon. This then opens an in-place smart lens pane (see the sample lens pane in the section showing sample UIs) that displays the results of the relational query that is based on merging the SQML of the underlying object with the SQML on the clipboard (by adding the SQML of the underlying object as an argument to the SQML on the clipboard with an appropriate predicate (likely the default predicate) and a link to the object’s resource identifier or URL. Essentially, this creates a new semantic SQML by relationally merging the SQML for the “lens” and the “object” to create an “image.”</td>
</tr>
<tr>
<td></td>
<td>Preferably only Smart Agents can be copied, pasted, and used as Smart Lenses. In addition, only objects returned by queries on Smart Agents can contain deep info. However, all objects can be dragged, dropped, copied, pasted, and can automatically generate context palettes (including dumb objects – e.g., from dumb agents that refer to the local file-system). This is how “dumb” objects can be made “smart” once they are in the semantic sandbox or environment.</td>
</tr>
<tr>
<td></td>
<td>If the “Copy” operation is invoked on an object in the Presenter results window (e.g., via the “Copy” verb on a popup menu), this also copies the SQML of the object to the clipboard. The Presenter will derive the SQML of the object from the object’s XML (or SRML) – primarily by adding a reference to its identifier or URL/path.</td>
</tr>
</tbody>
</table>

FIGURE 39 (continued)
<table>
<thead>
<tr>
<th>Tool</th>
<th>File System Explorer or Document Viewer</th>
<th>Semantic Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paste</td>
<td>Paste the contents of the clipboard to the selected destination. If the destination is a folder, this pastes the contents of the selected file or folder (if appropriate). If the destination is a document, this pastes the textual, graphical or link content of what is on the clipboard.</td>
<td></td>
</tr>
</tbody>
</table>

From the toolbar of the semantic browser, if a smart agent is currently selected or opened, the SCML for what is on the clipboard is inserted into the SCML for the smart agent to create a new SCML buffer or file that includes the relational semantic query with the clipboard object as the filter argument. The tool on the toolbar will only be enabled if there is something on the clipboard and if what is on the clipboard can be pasted onto the agent. The semantic browser will check this by examining the format of what is on the clipboard. If it is the SQML format or is a format that can be converted to SQML, the tool is enabled. If necessary, implicit conversion to SQML is performed when the Paste operation is invoked. Note that if the SQML on the clipboard is that of a smart agent, the semantic browser can support it if the agents belong to the same agency — in this case, the query will result in a complex sub-query that filters the results of the destination agent with the results of the source agent. If the SQML on the clipboard belongs to the currently selected agent (e.g., if the user wants to hit "Copy" and then immediately hit "Paste" on the same smart agent), the tool should be disabled since this will be a redundant operation (akin to the user copying a piece of text in a document and pasting over the same piece of text).

If the "Paste" operation is invoked on an object in the Presenter results window (via, say, the "Paste as Smart Lens" icon on an object), this opens an in-place smart lens pane (see the sample lens pane in the section showing sample UIs) that displays the results of the relational query that is based on merging the SQML of the underlying object with the SQML on the clipboard (by adding the SQML of the underlying object as an argument to the SQML on the clipboard with an appropriate predicate (likely the default predicate) and a link to the object's resource identifier or URL. Essentially, this creates a new semantic SQML by relationally merging the SQML for the "lens" and the "object" to create an "image."

FIGURE 39 (continued)
<table>
<thead>
<tr>
<th><strong>Tool</strong></th>
<th><strong>File System Explorer or Document Viewer</strong></th>
<th><strong>Semantic Browser</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Displays a dialog box showing the properties of the selected file or folder, or the current document.</td>
<td>If an agent (smart or dumb) is currently selected or opened, this displays a dialog box showing the properties of the agent (e.g., the name, description, the creation date/time, the last used date/time, etc.).</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the currently selected file, folder, graphic object, link, or text.</td>
<td>Deletes the currently selected or opened agent and refreshes the environment to reflect the new state.</td>
</tr>
<tr>
<td>Drag and Drop</td>
<td>Copies or moves the selected file, folder, graphics, or link from its source to the destination. If the object is a file or folder, the contents are copied to the destination folder. If the object is a graphic or text, the contents are copied to the “paste” section in the same document or a new document.</td>
<td>Links the file(s), folder(s), URL, text, or other local contextual link that are dragged to the smart agent that they are dragged to, in order to create a new smart agent that contains a semantic query that reflects the relationship between the dragged object (the context) and the smart agent (the destination). The operation results in a new smart agent with a new SQML buffer that has the original SQML of the smart agent (destination) that the source (files, folders, URLs, etc.) were dragged on, filtered with the references of the source. The new SQML is then “compiled” by the client, individual SQMLs are then sent to the server via its XML Web service (the reference of which is copied from the SQML of the destination smart agent), qualified with the XML metadata of the source references. For instance, if the source is a file document, the document metadata is extracted and the metadata sent to the XML Web Services as an SQML argument. If the source has multiple files or is a folder, the metadata for the files in the folder are sent (qualified by whether the source arguments are to be used as a UNION or INTERSECTION filter). If the source is an URL (HTTP, FTP, etc.), the URL is first downloaded and then the metadata for the document the URL refers to is sent as an SQML argument to the XML Web Services in the new SQML buffer or file. The browser will first validate that the source refers to an object type that the server can understand (e.g., a document, an email object, a person, etc.). If it is not a valid object, the browser should ignore the request and/or notify the user via a dialog alert message.</td>
</tr>
</tbody>
</table>

**Hierarchical Object View** Displays a hierarchical view of the folders in the file-system environment Displays a hierarchical view of the semantic environment, including agent views, agency views, agents viewed by “all,” “context,” “category,” “time,” “agency,” “information type,” “blender,” etc., and agencies viewed by “all,” “category,” “time,” etc. (see screenshots) |

FIGURE 39 (continued)
FIGURE 40
| **Last Item** Posted On: May 17, 12:14PM [Preview] |  |
| **Next Event** Starting On: June 28, 09:00AM [Preview] |  |
| **Show only items of the following type:** |  |
| **Show only items posted in the last:** |  |
| **Found a Total of 50 Items [Preview]** |  |

**FIGURE 41**
### Information Tools Layer

*Proprietary* — Hypermedia (Smart Semantic Links) + Metadata + Semantics/meaning + Drag and Drop + Smart Copy and Paste + Intrinsic Alerts + Smart Lens + Deep Info + Intrinsic Verbs + Intrinsic Links + Smart Agents + Context Templates (Special Agents) + Blenders + Smart Recommendations + Breaking News Agents + smart, context-sensitive links to breaking news Agents + semantic hints + Preview Windows + Presentation Skins (subtle, moderate, exciting, animations, etc.) + Smart Screensavers, etc.

### Presentation Layer

Dynamic, client-side, configurable, context-sensitive, time-sensitive “skins” — client-side conversion of XML to XHTML+TIME, SVG, etc. using XSLT.

### Client-Side Semantic Query Layer

SQML (*Proprietary*)

### Protocol Layer

XML Web Services

### Server-Side Semantic Query Layer

SQL, XQuery, XPath, etc.

### Semantic Network Data Model

(Proprietary—data stored in database)

### Semantic Data Source Adapters (DSAs) for Semantic and Non-Semantic Data Sources

Existing stores for documents, email, contacts, events, customer data, e-learning data, etc., metadata stores (RDF, XML, databases, etc.), proprietary mapping from data sources to semantic network

FIGURE 42
1. METADATA
2. SEMANTICS (MEANING, CONTEXT, DEEP INFO, ETC.)
3. CONTEXT-SENSITIVITY (DRAG AND DROP, SMART COPY AND PASTE, SMART SKINS, INTRINSIC LINKS, SMART LENS INITIATION AND RESPONSE, VERBS, DEEP INFO, SMART RECOMMENDATIONS, ETC.)
4. TIME-SENSITIVITY (INTRINSIC ALERTS, LINKS TO BREAKING NEWS AGENTS)
5. AUTOMATIC AND INTELLIGENT DISCOVERABILITY (LINKS TO NEW AGENCIES IN THE NAMESPACE)
6. DYNAMIC LINKING (DRAG AND DROP, SMART COPY AND PASTE)
7. USER-CONTROLLED NAVIGATION AND BROWSING (DRAG AND DROP, SMART COPY AND PASTE INTRINSIC ALERT, BREAKING NEWS, ETC.)
8. FLEXIBLE PRESENTATION (SMART SKINS, SMART SCREENSAVERS, ANIMATIONS, SEMANTICS UI HINTS, PREVIEW WINDOWS, ETC.)
9. NON-HTML AND NON-LOCAL DOCUMENTS PARTICIPATION IN THE NETWORK (LINKS TO DUMB AGENTS THAT GET INFORMATION FROM SOURCES WITHOUT AGENCIES (E.G., THE LOCAL FILE SYSTEM), RESPONSES TO SMART LENS FROM LOCAL FILES/CONTENT, DRAG AND DROP FROM LOCAL FILES/CONTENT, ETC.)
10. FLEXIBLE USER-DRIVEN INFORMATION ANALYSIS (SMART LENS, DEEP INFO, PREVIEW WINDOWS, PIVOT VIEWS, ETC.)
11. FLEXIBLE SEMANTICS QUERIES (ABILITY TO CREATE, EDIT AND LINK TO SMART AGENTS, DRAG AND DROP, SMART LENS, AND SMART COPY AND PASTE AS THE BASIS OF SEMANTICS QUERIES, ETC.)
12. INFORMATION PACKAGES/BLENDEERS (ABILITY TO DRAG AND DROP OR COPY/PASTE OBJECT TO BLENDER TO CREATE A NEW BLENDER WITH NEW SUB-QUERIES, SMART AGENT RESULTS CAN FORM BLENDERS, OBJECT CAN RESPOND TO BLENDER WHEN BLENDER IS USED AS A SMART LENS, ETC.)
13. PIVOTS FOR CONTEXT TEMPLATES (ABILITY FOR USER TO CREATE A HEADLINES SMART AGENT, NEWSMAKERS SMART AGENT, ETC. USING THIS OBJECT AS A PIVOT)

FIG. 43
SHARABLE SMART REQUEST DOCUMENT (.REQ) EXTENSION HANDLER

SHARABLE SMART REQUEST DOCUMENT BUFFER STREAM (BINARY FORMAT)

SQML BUFFER STREAM (XML TEXT FORMAT)

SEMANTIC CLIENT RUNTIME

SEMANTIC NAMESPACE

STORAGE (DISK. ETC.)

EMAIL. ETC.

FIG. 45
FIG. 46
FROM: NOSA OMOGUI
TO: JOHN SMITH
CC: JOE SOMEBODY
SUBJECT: WEB SERVICES
SUMMARY: WEB SERVICES ARE SOFTWARE BUILDING BLOCKS USED FOR DISTRIBUTED COMPUTING
BODY: WEB SERVICES...

FIG. 47

FIG. 48
FIG. 50
WELCOME TO NERVANA LIBRARIAN
GET STARTED NOW: OPEN THE TREE VIEW, RIGHT-CLICK "MY PROFILE", THEN SELECT PROPERTIES TO PERSONALIZE YOUR PROFILE.

FIG. 51
FILTERED SMART REQUESTS (e.g., HEADLINES ON WIRELESS)

CURRENT SMART REQUEST TITLE (e.g., "BREAKING NEWS")

FIG. 52
FIG. 53
Title: AWC Homeland Security
Type: Email/Email Posting
Location: nerv.com/feml
Author: G. Schmitz
Company: Nervana
Categories: Homeland Security
Search Phrases: 'Homeland Security' "DoD Directives impacting Homeland Security, as listed by RAND. Law Enforcement"

Found 367 items. Items 1-4 of 367

AWC: Homeland Security
DoD Directives impacting Homeland Security, as listed by RAND. Law Enforcement

Found 5 relevant items
Preview

Breaking News

Found 23 relevant headlines
Preview

Leaders should understand and implement the policy changes necessary to fully realize this

Geographic Systems Leaders should understand and implement the policy changes necessary to fully realize this

Found 4 relevant items
Preview

Breaking News

Using 21st Century Technology
This ongoing homeland security initiative is a key component of government’s overall investment

Found 4 relevant items
Preview

Breaking News

Computer Security Division
To create the Department of Homeland Security will involve advancements in security technology

Found 4 experts
Preview

Experts

Found 17 items
items 1–12 of 17

Sharing is the focus

State CISOs Advise

Washington Technology

Scope of Proposal

Data Integration

Government Executive

Homeland Security CIO

Citizen is the Focus

Planet Views

Information Builders

Security Solutions

Information Exchange

FIG.54
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>INDEXED</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectID</td>
<td>BIGINT (8 bytes)</td>
<td>Yes (covering index - with all columns)</td>
<td>SubjectsID, PredicateTypeID</td>
</tr>
<tr>
<td></td>
<td>UNICODE String</td>
<td>Yes (covering index - with all columns)</td>
<td>SubjectsID, PredicateTypeID</td>
</tr>
</tbody>
</table>

**FIGURE 60**

**FIGURE 61**
<table>
<thead>
<tr>
<th>Qualifier</th>
<th>PredicateTypeID_AuthorContainsConcept</th>
<th>PredicateTypeID_PublisherContainsConcept</th>
<th>PredicateTypeID_LanguageContainsConcept</th>
<th>PredicateTypeID_HostContainsConcept</th>
<th>PredicateTypeID_TypeContainsConcept</th>
<th>PredicateTypeID_BodyContainsConcept</th>
<th>PredicateTypeID_PubDateContainsConcept</th>
<th>PredicateTypeID_PubYearContainsConcept</th>
</tr>
</thead>
<tbody>
<tr>
<td>author:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>publisher:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>language:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>host:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>filetype:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>title:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>body:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pubdate:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pubyear:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 66
<table>
<thead>
<tr>
<th>RESULTS</th>
<th>WARLORDS FACING WELL IN AFGHAN ELECTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PINR-Poland's Rightward Turn and the Significance for Europe</td>
</tr>
<tr>
<td></td>
<td>WARLORDS FACING WELL IN AFGHAN ELECTIONS</td>
</tr>
<tr>
<td></td>
<td>PINR-Poland's Rightward Turn and the Significance for Europe</td>
</tr>
</tbody>
</table>
FIG. 68

FIG. 69

FIG. 70
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Indexed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectId</td>
<td>BIGINT (8 bytes)</td>
<td>Yes (primary key; clustered)</td>
<td></td>
</tr>
<tr>
<td>ObjectTypeID</td>
<td>INT (4 bytes)</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td>UNICODE String</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>UNICODE String</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>SourceUri</td>
<td>UNICODE String</td>
<td>Yes (non-clustered)</td>
<td>UNIQUE constraint</td>
</tr>
<tr>
<td>Language</td>
<td>UNICODE String</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OriginalCreationTime</td>
<td>DATETIME</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OriginalLastModifiedTime</td>
<td>DATETIME</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ObjectCreationTime</td>
<td>DATETIME</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
<tr>
<td>ObjectLastModifiedTime</td>
<td>DATETIME</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>BIGINT</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>BetStrength</td>
<td>BIGINT</td>
<td>No</td>
<td>aggregate semantic strength of a document object</td>
</tr>
<tr>
<td>NumConcepts</td>
<td>BIGINT</td>
<td>No</td>
<td>number of concepts in a document object</td>
</tr>
<tr>
<td>Creators</td>
<td>UNICODE String</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Contributors</td>
<td>UNICODE String</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Publishers</td>
<td>UNICODE String</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 74-1
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Indexed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>BestBetHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td>whether an object is a Best Bet as indicated by the Semantic Inference Engine (SIE)</td>
</tr>
<tr>
<td>RecommendationHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td>whether an object is a Recommendation as indicated by the Semantic Inference Engine</td>
</tr>
<tr>
<td>BreakingNewsHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td>whether an object is Breaking News as indicated by the Time-Sensitivity Inference Engine</td>
</tr>
<tr>
<td>HeadlinesHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td>whether an object is Breaking News as indicated by the Time-Sensitivity Inference Engine</td>
</tr>
<tr>
<td>BetRankHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td>representative score of the semantic strength of an object</td>
</tr>
<tr>
<td>RichMetadataHint</td>
<td>SMALLINT (2 bytes)</td>
<td>No</td>
<td>whether an object came from a rich metadata source (e.g. RSS)</td>
</tr>
<tr>
<td>SemanticHash</td>
<td>UNICODE String</td>
<td>No</td>
<td>a hash of the body of a document object for duplicate detection alphabetical order</td>
</tr>
</tbody>
</table>

FIGURE 74-2
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Indexed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectId</td>
<td>BIGINT (8 bytes)</td>
<td>Yes (primary key; clustered)</td>
<td></td>
</tr>
<tr>
<td>ObjectTypeID</td>
<td>INT (4 bytes)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Uri</td>
<td>UNICODE String</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 75
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Indexed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinkID</td>
<td>BIGINT</td>
<td>Yes</td>
<td>Represents the Best Bet context predicate as supplied by the Semantic Inference Engine</td>
</tr>
<tr>
<td>SubjectID</td>
<td>BIGINT</td>
<td>Yes</td>
<td>Represents the Recommendations context predicate as supplied by the Semantic Inference Engine</td>
</tr>
<tr>
<td>PredicateTypeID</td>
<td>INT</td>
<td>Yes</td>
<td>Represents the Breaking News context predicate as supplied by the Time-Sensitivity Inference Engine</td>
</tr>
<tr>
<td>ObjectId</td>
<td>BIGINT</td>
<td>Yes</td>
<td>Represents the Headlines context predicate as supplied by the Time-Sensitivity Inference Engine</td>
</tr>
<tr>
<td>LinkStrength</td>
<td>BIGINT</td>
<td>Yes</td>
<td>Represents the BetRankHint representative score of an object's semantic strength</td>
</tr>
<tr>
<td>BestBetHint</td>
<td>SMALLINT</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>RecommendationHint</td>
<td>SMALLINT</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>BreakingNewsHint</td>
<td>SMALLINT</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HeadlinesHint</td>
<td>SMALLINT</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>BetRankHint</td>
<td>SMALLINT</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Partitioned</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>SubjectID</td>
<td>BIGINT (8 bytes)</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
<tr>
<td>PredicateTypeID</td>
<td>INT (4 bytes)</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
<tr>
<td>ObjectID</td>
<td>BIGINT (8 bytes)</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
<tr>
<td>BestBetHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
<tr>
<td>RecommendationHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
<tr>
<td>BreakingNewsHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
<tr>
<td>HeadlinesHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
<tr>
<td>BetRankHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 77**
Incoming metadata feed containing individual informational items

Metadata feed skipped in its entirety because it was previously processed

Metadata Processor

Metadata feed parsed and individual information items

Database table containing prior received metadata feeds

FIGURE 78
Objects

Semantic strength assigned to objects for a given context

Top semantic strength values of all objects in a context marked based on a relative threshold value

Adjuster modifies the threshold value based on the highest semantic strength score

Adjuster updates recommendations threshold value

FIGURE 79
NEWS SEMANTIC TAILORED CONTENT DELIVERED TO CLIENT → SEMANTIC SOUND GENERATOR → TAILORED AUDIBLE SOUND PRODUCED

FIG. 84

PERSONAL

DEFAULT FAVORITES LIVE MY DOCUMENTS

COMMUNITY

REFERENCES POPULAR NEWS BEST BETS

FIG. 85
FIG. 86

NON ALPHA TEXT DATA ANNOTATED WITH TEXT

ANNOTATIONS LINKED TO NON-TEXT DATA SOURCE

ANNOTATIONS SEMANTICALLY INDEXED

CLIENT

FIG. 87

CLIENT

ONTOSTY SUPPORT PERSONNEL

ONTOSTY DATA
Concepts Extracted → 3404 Server → Knowledge Community 1 → 3406

Concepts Extracted → 3404 Server → Knowledge Community 2 → 3408

Concepts Extracted → 3404 Server → Knowledge Community 3 → 3410

Map Returned Category Weights to Context Templates → 3412

Create Query Rules → 3414

Associate Query Rules with Context Template → 3416

FIGURE 96
Weighted Index Applied indicating Strength of Relationship between Object and Category

Information Type Selected by Client User

Client

FIGURE 103
<table>
<thead>
<tr>
<th>KNOWLEDGE FILTER</th>
<th>PRIMARY RANK AXIS</th>
<th>SECONDARY RANK AXIS</th>
<th>TERTIARY RANK AXIS</th>
<th>FILTER AXIS</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL BETS</td>
<td>SEMANTIC RELEVANCE</td>
<td>TIME</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>NONE</td>
<td>EMINENCE IS ON BREADTH</td>
</tr>
<tr>
<td>BEST BETS</td>
<td>SEMANTIC RELEVANCE</td>
<td>TIME</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>SEMANTIC RELEVANCE (VERY HIGH)</td>
<td>EMINENCE IS ON DEPTH</td>
</tr>
<tr>
<td>BREAKING NEWS</td>
<td>SEMANTIC RELEVANCE</td>
<td>TIME</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>TIME (VERY HIGH)</td>
<td>EMINENCE IS ON VERY HIGH TIME-SENSITIVITY; ANALOGOUS TO CNN'S BREAKING NEWS OR &quot;AT THIS HOUR&quot; AS BASIS FOR SEMANTIC FILTERING AND INTERPRETATION</td>
</tr>
<tr>
<td>HEADLINES</td>
<td>SEMANTIC RELEVANCE</td>
<td>TIME</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>TIME (HIGH)</td>
<td>EMINENCE ON TIME-SENSITIVITY; ANALOGOUS TO CNN'S &quot;HEADLINE NEWS&quot; PROGRAM</td>
</tr>
<tr>
<td>RECOMMENDATIONS</td>
<td>TIME (DESCENDING)</td>
<td>SEMANTIC RELEVANCE</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>NONE</td>
<td>EMINENCE IS ON COMPUTER-AIDED GUIDANCE (&quot;I DON'T KNOW WHAT I DON'T KNOW&quot;)</td>
</tr>
<tr>
<td>RANDOM BETS</td>
<td>NONE (RANDOM)</td>
<td>NONE (RANDOM)</td>
<td>NONE (RANDOM)</td>
<td>NONE</td>
<td>EMINENCE IS ON SERENDIPITY AND DISCOVERY</td>
</tr>
<tr>
<td>CLASSIC</td>
<td>TIME (ASCENDING)</td>
<td>SEMANTIC RELEVANCE</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>TIME (VERY HIGH/OLD)</td>
<td>EMINENCE IS ON OLD FAVORITES, HISTORICAL PERSPECTIVE, NOSTALGIA</td>
</tr>
<tr>
<td>CONVERSATIONS</td>
<td>SEMANTIC RELEVANCE</td>
<td>TIME (DESCENDING)</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>THREADS INVOLVING MULTIPLE PEOPLE</td>
<td>EMINENCE IS ON THE EXISTENCE OF HUMAN DISCOURSE AND Debates; ANALOGOUS TO CNN'S &quot;CROSSFIRE&quot; PROGRAM</td>
</tr>
</tbody>
</table>

FIG. 104A
<table>
<thead>
<tr>
<th>COMMENTARY ITEMS</th>
<th>SEMANTIC RELEVANCE</th>
<th>TIME (DESCENDING)</th>
<th>NON-SEMANTIC RELEVANCE</th>
<th>COMMUNITY PUBLICATIONS AND ANNOTATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANNOTATED ITEMS</td>
<td>SEMANTIC RELEVANCE</td>
<td>TIME (DESCENDING)</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>ITEMS THAT HAVE BEEN ANNOTATED</td>
</tr>
<tr>
<td>POPULAR ITEMS</td>
<td>POPULARITY</td>
<td>SEMANTIC RELEVANCE</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>POPULAR ITEMS</td>
</tr>
<tr>
<td>EXPERTS</td>
<td>SEMANTIC RELEVANCE</td>
<td>TIME (DESCENDING)</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>PEOPLE THAT HAVE RELEVANT EXPERTISE</td>
</tr>
<tr>
<td>INTEREST GROUP</td>
<td>SEMANTIC RELEVANCE</td>
<td>TIME (DESCENDING)</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>PEOPLE THAT HAVE AN INTEREST IN ONE OR MORE TOPICS</td>
</tr>
<tr>
<td>NEWSMAKERS</td>
<td>SEMANTIC RELEVANCE</td>
<td>TIME (DESCENDING)</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>PEOPLE THAT HAVE RECENTLY PUBLISHED AN OPINION</td>
</tr>
<tr>
<td>UPCOMING EVENTS</td>
<td>SEMANTIC RELEVANCE</td>
<td>TIME (DESCENDING)</td>
<td>NON-SEMANTIC RELEVANCE</td>
<td>UPCOMING EVENTS</td>
</tr>
</tbody>
</table>

**FIG.104B**
[+] WORK
  [+] EMPLOYEES
  [+] BUSINESS PARTNERS
  [+] CUSTOMERS
  [+] EVENTS
[+] FAMILY
  [+] SIBLINGS
  [+] WIFE AND KIDS
  [+] EXTENDED FAMILY
[+] VACATION
  [+] MEXICO
  [+] FRANCE
  [+] ROCK CLIMBING
  [+] SKIING

FIG. 105

<table>
<thead>
<tr>
<th>PROPERTY NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNOWLEDGE DOMAIN ID</td>
<td>76331eb3-e494-45b5-8939-a4db68bea4bd</td>
</tr>
<tr>
<td>KNOWLEDGE DOMAIN NAME</td>
<td>LIFE SCIENCES</td>
</tr>
<tr>
<td>KNOWLEDGE DOMAIN PUBLISHER NAME</td>
<td>NERVANA, INC.</td>
</tr>
<tr>
<td>KNOWLEDGE DOMAIN PUBLISHER DOMAIN NAME</td>
<td>NERVANA.COM</td>
</tr>
<tr>
<td>KNOWLEDGE DOMAIN PUBLISHER DOMAIN ZONE</td>
<td>INTERNET</td>
</tr>
</tbody>
</table>

FIG. 106

<table>
<thead>
<tr>
<th>PROPERTY NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>CARDIOVASCULAR DISORDER EPIDEMIOLOGY</td>
</tr>
<tr>
<td>URI</td>
<td>nerv://76331eb3-e494-45b5-8939-a4db68bea4bd?type=category&amp;path=Biology/Ecology/Human Ecology/Human Population Study/Epidemiology/Cardiovascular Disorder Epidemiology</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>0.431</td>
</tr>
</tbody>
</table>

FIG. 107
<table>
<thead>
<tr>
<th>YOUR QUESTION</th>
<th>NERVANA QUERY</th>
</tr>
</thead>
</table>
| * WHAT STUDIES HAVE BEEN THAT IDENTIFY THE RELATIONSHIP BETWEEN CIGARETTE OR SECOND-HAND SMOKE AND ALL FORMS OF HEART DISEASE? | REQUEST TYPE: ALL BETS  
CATEGORY: HEART DISEASES (MeSH)  
AND  
KEYWORDS: (SMOKING OR SECOND-HAND SMOKE OR CIGARETTE OR CIGARETTES) |
| * I NEED HIGHLY RELEVANT STUDIES ABOUT THE IMPACT OF APOPTOSIS OR OTHER TYPES OF CELL DEATH ON LYMPHATIC CANCER. | REQUEST TYPE: RECOMMENDATIONS  
CATEGORY: BLOOD AND LYMPHATIC CANCER (CRISP)  
AND  
CATEGORY: CELL DEATH (CO-BIOLOGICAL PROCESS) |
| * HAS ANY RESEARCH BEEN DONE IN THE PAST THREE YEARS BY AMGEN, GENENTECH, OR AVENTIS ON THE IMPACT OF CYTOKINE RECEPTOR GENES ON CANCER? | REQUEST TYPE: DDOSSIER  
CATEGORY: NEOPLASM AND CANCER (CRISP)  
AND  
CATEGORY: CYTOKINE RECEPTOR GENE (CANCER, NCI)  
AND  
KEYWORDS: (2003 OR 2004 OR 2005)  
AND  
KEYWORDS: (AMGEN OR GENENTECH OR AVENTIS) |
| * ARE THERE ANY STUDIES MOST RELEVANT TO LIGAND BINDING PROTEIN GENES? | REQUEST TYPE: BEST BETS  
CATEGORY: LIGAND BINDING PROTEIN GENES (CANCER, NCI) |
| * HAS ANY RECENT RESEARCH BEEN DONE ON LIGAND BINDING PROTEIN GENES BY THE UNIVERSITY OF WASHINGTON? | REQUEST TYPE: HEADLINES  
CATEGORY: LIGAND BINDING PROTEIN GENES (CANCER, NCI)  
AND  
KEYWORDS: "UNIVERSITY OF WASHINGTON" |
| * I NEED THE LATEST AVAILABLE RESEARCH ON CELL COMMUNICATION BY DR. LEE HARTWELL. | REQUEST TYPE: BREAKING NEWS  
CATEGORY: CELL COMMUNICATION (CO-BIOLOGICAL PROCESS)  
AND  
KEYWORDS: "HARTWELL LH" |
| * I'M NEW TO THE FIELD OF PROTEIN FOLDING AND WISH TO EXPLORE ALL ITS RESEARCH AVENUES. I AM LOOKING FOR OBSCURE RESEARCH PAPERS THAT MIGHT HAVE "HIDDEN JEWELS." WHERE DO I START? | REQUEST TYPE: RANDOM BETS  
CATEGORY: PROTEIN FOLDING (CO-BIOLOGICAL PROCESS) |

FIG.111
* Is there any research done by Bristol-Myers Squibb about how monoclonal antibodies relate to any type of cancer?

* I'm looking for studies on cancer authored by a researcher named Smith at the University of Washington and Dr. Cole who is affiliated with Harvard University.

* What is the most recent research available about diagnostic tests being developed for HIV?

* Is there any research about how people can reduce exposure to pollution?

* What is the cumulative effect of multiple pollutants on human health?

* What types of zinc therapies are being used or developed for curing the common cold?

Learn more about Nervana...

*Sales: 425.637.1105
  sales@nervona.com

*SUPPORT: 1.866.NERVANA
  support@nervona.com

FIG.112
FIGURE 117

Live Results by Publisher

- Pfizer: 21%
- CNN: 12%
- Gene News: 8%
- Merck: 6%
- BBC: 6%
- Novartis: 5%
- FDA: 5%
- MSNBC: 8%
- AP: 10%
- Aventis: 9%
- Other: 15%
Live Results by Publisher

- Pfizer: 30%
- CNN: 25%
- Gene News: 15%
- Merck: 10%
- BBC: 5%
- Novartis: 5%
- FDA: 5%
- MSNBC: 5%
- AP: 5%
- Aventis: 5%

FIGURE 118
Display Options

New Window Display Settings:
When launching a new Librarian item:
- Use the current browser instance

Highlighting Settings:
- Enable semantic highlighting with: Standard Highlighting
FIGURE 120
FIGURE 121
FIGURE 122
Browse Knowledge Directories

Show Directory Entries

Published to: [ ] All Knowledge Directories
Relevant to: [ ] My Areas of Interest
Type: [ ] All Types

Directory Entries

- All Patent Authorities (Life Sciences)
- European Granted Patents (Life Sciences)
- European Published Applications (Life Sciences)
- General News
- Life Sciences Business Discovery
- Life Sciences Drug Discovery
- Life Sciences IP Discovery
- Life Sciences News
- Life Sciences Premium Discovery
- Life Sciences Web
- MEDLINE
- US Granted Patents (Life Sciences)

nervana

Knowledge Directories

The following Knowledge Application groups and Knowledge Community groups are available for subscription.

All Patent Authorities

The All Patent Authorities Knowledge Community group includes U.S. Granted Patents, U.S. Published Applications, European Granted Patents, and European Published Applications. This service is updated weekly.

FIGURE 124
Long-Term Decline in R&D Productivity

FIGURE 125
- Semi-automated ontology mining
- Make it easy for humans to participate
- Virtuous cycle
FIGURE 136

FIGURE 137
PRODUCT ILLUSTRATIONS FOR NERVANA SOCIAL DISCOVERY™

The following are illustrations of what a typical Nervana user will see when logged in to Nervana Social Discovery™:

MY DISCOVERY SPACE

- General
  - My Nervana Networks
    - Pfizer’s Global Innovation Network
    - American Cancer Institute’s Oncology Network
  - My Collaborators
    - John Smith
    - Philip Davis, Ph.D.
  - My Groups
    - Pfizer - Autoimmune Diseases
    - American Cancer Society
  - My Queries
    - Drugs used to treat infectious diseases

- [-] Nervana’s Recommendations
  - [-] All
  - [-] Relevant Industry News
  - [-] Relevant Industry Blogs
  - [-] Relevant Web Pages
  - [-] Relevant Patents
  - [-] Relevant Patent Applications
  - [-] Relevant Scientific Literature
  - [-] Relevant People within the Social Network
  - [-] Relevant Information within the Social Network
    - [-] Relevant Projects
    - [-] Relevant Ideas
    - [-] Relevant Meetings & Events
  - [-] Relevant Premium Scientific Content
  - [-] Relevant Scientific Lecture Videos
  - [-] Relevant Podcasts
  - [-] Relevant Theses & Dissertations
  - [-] Relevant Industry Events

1 A Nervana user’s identity automatically roams.

2 Subscribing to a person adds that person to the “My Collaborators” list. Each user has the ability to impose access control rules so that only these users he/she trusts will be able to see sensitive sections of his/her space.

3 These semantic connections are much more powerful than today’s social networks’ (like LinkedIn’s or Facebook’s), which typically have notifications without context and which tend to become meaningless after several degrees of separation.

FIGURE 138
Projects can contain multiple documents

These are documents that are processed as natural-language semantic queries

1 Users will be able to semantically share their favorite documents and have those documents intelligently connected to people and information in the social network. Companies will also be able to publish generic corporate documents for discovery and collaboration.
Nervana Discovery Spaces Daily Digest for Philip Rivers

Your Discovery Space has a total of 236 new items today. Here is the breakdown:

My Collaborators

- John Smith: 12 new relevant items
- Philip Watson, Ph.D.: 23 new relevant items

My Groups

- Pfizer – Autoimmune Diseases: 211 new relevant items
- American Cancer Society: 123 new relevant items

My Queries

- Drugs used to treat infectious diseases: 14 new relevant items
- Diagnostic techniques for cancer detection: 7 new relevant items including 2 new relevant people in the social network based on their backgrounds, and 3 new relevant people in the network based on their projects, documents, and ideas.

My Projects

- Technical Report on Inhibition of Cell Migration to Joint Tissues: 27 new relevant items including 5 new relevant people in the social network based on their backgrounds, and 6 new relevant people in the network based on their projects, documents, and ideas.

My Ideas

- Idea on a new Technique for Cell Signaling: 13 new relevant items including 5 new relevant people in the social network based on their backgrounds, and 7 new relevant people in the network based on their projects, documents, and ideas.

---

1 The Daily Digest illustrates what the user will see via an email and/or RSS alert, indicating new updates to his/her Discovery Space.

FIGURE 140
Community Content → TF-IDF → Stemming-based Semantic Variants → Exclusions based on Ontology Rings → Lookup for Predicates and Relationships

FIGURE 141
Next-Generation Clinical Trial Services Proposal

Nervana Discovery Publisher™ - OEM
- Smart RSS Feeds
- Smart Portals

Nervana Social Discovery™
- Knowledge-Driven Collaboration
- Social Networking

Nervana Smart Documents™

Clinical Trial Semantics Knowledge Services

CTS Knowledge Base
- Clinical Trials Data
- Research
- Nervana-Provided Content:
  - Patents
  - Drug Safety Information
  - Market Intelligence

FIGURE 142
Next-Generation Clinical Trial Workflow Proposal

FIGURE 143
Next-Generation Research Collaboration Platform

Nervana Social Discovery

FIGURE 144
# Technology Uniqueness

## Nervana Competitive Matrix - Knowledge Discovery

<table>
<thead>
<tr>
<th>Feature</th>
<th>Nervana</th>
<th>Convex</th>
<th>Autonomy</th>
<th>Vidy</th>
<th>Endeca</th>
<th>Insight</th>
<th>FAST</th>
<th>Vivisimo</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Semantic Search</td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
</tr>
<tr>
<td>Natural Language Based Queries</td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
</tr>
<tr>
<td>Federation and Cross Fertilization of Ontologies</td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
</tr>
<tr>
<td>Hypothesis Support and Qualification</td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
</tr>
<tr>
<td>Using Documents/Text as a Query</td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
</tr>
<tr>
<td>Contextual Awareness/Alerts</td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
</tr>
<tr>
<td>End User Personalization &amp; Collaboration Tools</td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
</tr>
<tr>
<td>Software-As-A-Service</td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
</tr>
</tbody>
</table>

*FIGURE 145*
Nervana Solution

Examples:
• Information
• Drug Targets
• Clinical Data
• Disease Phenotype
• Time
• Environmental Factors

Web pages
tech reports
documents

1. Semantic indexing
2. Knowledge retrieval
3. Rank / group

Examples:
• Discovery Libraries
• Discovery
• Early warning systems

FIGURE 146
Business Process with Nervana AdEngine™

FIGURE 147
AdEngine™ Workflow

- Ad Descriptions
- Product Descriptions
- Press Releases
- Brochures

Ad Campaign

Nervana Semantic Profiling

Publishers

Nervana Semantic Matching

FIGURE 148
Nervana AdEngine™
Site Ranking Attributes

FIGURE 149
Content Sources

FIGURE 150
Nervana TalentEngine™
AI Model Components

Overall Candidate Relevance

Job Industry Relevance
Job Role Relevance
Job Experience Relevance
Job Skills Relevance
General Relevance
Custom Relevance (Ultimate Edition)
Red Flags

FIGURE 151
SYSTEM AND METHOD FOR KNOWLEDGE RETRIEVAL, MANAGEMENT, DELIVERY AND PRESENTATION

PRIORITY CLAIM


[0002] This application is also a Continuation-In-Part Application of and claims priority to U.S. Provisional Patent Application No. 60/970,498 filed Sep. 6, 2007 and U.S. Provisional Patent Application No. 60/820,606 filed Jul. 27, 2006. This application also claims priority to U.S. patent application Ser. No. 11/383,736 (NERV-1-1013), filed May 16, 2006 which application claims priority to U.S. Provisional Patent Application No. 60/681,892 filed May 16, 2005. This application also claim priority to U.S. patent application Ser. No. 11/127,021 filed May 10, 2005 which application claims priority to U.S. Provisional Application Ser. Nos. 60/569,663 (Attorney Docket No. NERV-1-1007) and/or U.S. Provisional Application Ser. No. 60/569,665 (Attorney Docket No. NERV-1-1008).


[0005] All of the foregoing applications are hereby incorporated by reference in their entirety as if fully set forth herein.

COPYRIGHT NOTICE

[0006] This disclosure is protected under United States and International Copyright Laws.© 2002-2009 Nosa Omoigui. All Rights Reserved. A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

BACKGROUND OF THE INVENTION

[0007] Knowledge is now widely recognized as a core asset for organizations around the world, and as a tool for competitive advantage. In today's connected, information-based world, knowledge-workers must have access to the knowledge and the tools they need to make better, faster, and more informed decisions to improve their productivity, enhance customer relationships, and to make their businesses more competitive. In addition, industry observers have touted "agility" and the "real-time enterprise" as important business goals to have in the information economy.

[0008] Many organizations have begun to realize the value of disseminating knowledge within their organizations in order to improve products and customer service, and the value of having a well-trained workforce. The investments businesses are making in e-learning and corporate training provides some evidence of this. Companies have also invested in tools for content management, search, collaboration, and business intelligence. Companies are also spending significant resources on digitizing their business processes, particularly with respect to acquiring and retaining customers.

[0009] However, many knowledge/learning and customer-relationship assets are still stored in a diverse set of repositories that do not understand each other's language, and as a result are managed and interrelated with as independent islands of information. As such, what many organizations call "knowledge" is merely data and information. The information economy in large part is a struggle to find a way to provide context, meaning and efficient access to this ever increasing body of data and information. Or, stated differently, to turn the mass of available data and information into usable knowledge.

[0010] Information has been long accessible in a variety of forms, such as in newspapers, books, radio and television media, and in electronic form, with varying degrees of proliferation. Information management and access changed dramatically with the use of computers and computer networks. Networked computer systems provide access throughout the system to information maintained at any point along the system. Users need only establish the requisite connection to the network, provide proper authorization and identify the desired information to obtain access.

[0011] Information access further improved with the advent of the Internet, which connects a large number of computers across diverse geography to provide access to a
vast body of information. The most wide spread method of providing information over the Internet is via the World Wide Web. The Web consists of a subset of the computers or Web servers connected to the Internet that typically run Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), Gopher or other servers. Web servers host Web pages at Web sites. Web pages are encoded using one or more languages, such as the original Hypertext Markup Language (HTML) or the more current eXtensible Markup Language (XML) or the Standard Generic Markup Language (SGML). The published specifications for these languages are incorporated by reference herein. Web pages in these formats may be accessed by Internet users via web browsing software such as Microsoft’s Internet Explorer or Netscape’s Navigator.

Both the current Web and the Semantic Web fail to provide proper context, meaning and efficient access to data and information to allow users to acquire actionable knowledge. This is partially a problem related to the ways in which Today’s Web and the contemplated Semantic Web are structured or, in other words, related to their technology layers. As shown in FIG. 1-1, Today’s Web, for example, which is a hypertext medium, provides the three technology layers, which include “dumb” links, or links having no context-sensitivity, time-sensitivity, etc. Present conceptualizations of the Semantic Web, also referred to as a “semantic hypermedia,” provide for five technology layers, as shown in FIG. 1-2. As explained in greater detail below, there are serious limitations associated with each of the technology layer structures.

In addition, various properties must be present in a comprehensive information management system to provide an integrated and seamless implementation framework and resulting medium for knowledge retrieval, management and delivery. A non-exhaustive list of these properties include: Semantics/Seamless; Context-Sensitivity; Time-Sensitivity; Automatic and Intelligent Discoverability; Dynamic Linking; User-Controlled Navigation and Browsing; Non-HTML and Local Document Participation in the Network; Flexible Presentation that Smartly Conveys the Semantics of the Information being Displayed; Logic, Inference, and Reasoning; Flexible User-Driven Information Analysis; Flexible Semantic Queries; Read/Write Support; Annotations; “Web of Trust” Information Packages (“Blenders”); Context Templates, and User-Oriented Information Aggregation. Each of these properties will be discussed below in the context of their applicability to both Today’s Web and the Semantic Web.

In enterprises around the world today, the Web is the information platform for knowledge-workers. And there lies the problem. The Web as we know it is a platform for data and information while its users operate at the level of “knowledge.” This disconnect is a very fundamental one and cannot be understated. The Web, in large measure, has fulfilled the dream of “information at your fingertips.” However, knowledge-workers demand “knowledge at your fingertips” as opposed to mere “information at your fingertips.” Unfortunately, today’s knowledge-workers use the Web to browse and search for documents—compilations of data and information—rather than actual knowledge relevant to their inquiry. To achieve improved knowledge requires providing proper context, meaning and efficient access to data and information, all of which are missing with the traditional Web.

Efforts have been made to achieve the goal of “knowledge at your fingertips.” One example is a new concept for information organization and distribution referred to as the Semantic Web. The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. While conceptually a significant step forward in supporting improved context, meaning and access of information on the Internet, the Semantic Web has yet to find successful implementation that lives up to its stated potential.

Today’s Web lacks semantics as an intrinsic part of the platform and user experience. Web pages convey only textual and graphical data rather than the semantics of the data they contain. As a result, users cannot issue semantic queries such as those that one might expect with natural language—for example, “find me all books less than hundred pages long, about Latin Jazz, and published in the last five years.” To be able to process such a query, a Web site or search engine must “know” it contains books and must be able to intelligently filter its contents based on the semantics of the query request. Such a query is not possible on the Web today. Instead, users are forced to rely on text-based searches. These searches usually result in information overload or information lost because the user is forced to pick search terms that might not match the text in the information base. In the aforementioned example, a user might pick the search term “Books Latin Jazz” and hope that the search engine can make the connection. The user is usually then left to independently filter the search results. This sort of text-based search also implies that terms that might convey the same meaning. In the above example, results from search terms such as “Books on South or Central American Jazz” or “Publications on Jazz from Latino Lands” might be ignored during the processing of the search query.

The lack of semantics also implies that Today’s Web does not allow users to navigate based on the way humans think. For example, one might want to navigate a corporate intranet using the organizational structure. For example, from people to the documents they create to the experts on that documents to the direct reports of those experts to the distri-
The lack of semantics also has other implications. First, it means that the Web is not programmable. With semantics, the Web can be consumed by Smart Agents that can make sense of the pages and the links and then make inferences, recommendations, etc. With Today’s Web, the only “Agent” that can make inferences is the human brain. As such, the Web does not employ the enormous processing power that computers are capable of—because it is not represented in a way that computers can understand.

The lack of semantics also implies that information is not actionable. A search engine does not “understand” the results it spits out. As such, once a user receives search results, he or she is “on his or her own.” Also, a web browser does not “understand” the information it is displaying and as such cannot do smart things with the information. With semantics in place, a smart display, for example, will “know” that an event is an event and might do interesting things like check if the event is already in the user’s calendar, display free/busy information, or allow the user to automatically insert the event into his/her calendar thereby making the information actionable. Information presented without semantics is not actionable or might require that the semantics be inferred, which might result in an unpleasant user experience.

The Semantic Web seeks to address semantics/meaning limitations with Today’s Web by encoding information with well-defined semantics. Web pages on the Semantic Web include metadata and semantic links to other metadata, thereby allowing search engines to perform more intelligent and accurate searches. In addition, the Semantic Web includes ontologies that will be employed for knowledge representation, thereby allowing a semantic search engine to interpret terms based on meaning and not merely on text. For example, in the previous example, Latin Jazz ontology might be employed on a Semantic Web site and would allow a search engine on the site to “know” that the terms “Books on South or Central American Jazz” or “Publications from Jazz from Latin Lands” have the same meaning as the term “Books on Latin Jazz.” While conceptually overcoming many of the deficiencies with Today’s Web, there has not to date been a successful implementation of a well-defined data model providing context and meaning, including in particular the necessary semantic links, ontologies, etc. to provide additional characteristics such as context-sensitivity and time-sensitivity.

Contact Sensitivity

Today’s Web lacks context-sensitivity. The implication of a lack of context is that Today’s Web is not personal. For example, documents in accessible storage are independently static and therefore stupid. Information relevant to the subject matter of the document has already been published, being newly published, or will soon be published. Because the document in storage is static, however, there is no way to dynamically associate its subject matter with this relevant information in real-time. Stated differently, users have no way to dynamically connect their private context with external information in real-time. Information sources (such as the document) that form context sit in their own islands, totally isolated from other relevant information sources. This results in information and productivity losses.

The primary reason for this is that Today’s Web is a presentation-oriented medium designed to present views of information to a dumb client (e.g., remote computer). The client has virtually no role in the user experience, aside from merely displaying what the server tells it to display. Even in cases where there is client-side code (like Java applets and ActiveX controls), the controls usually do one specific thing and do not have coordinated action with the remote server such that code on the client is being orchestrated with code on the server.

From a productivity standpoint, the implication of this is that knowledge-workers and information consumers are totally at the mercy of information authors. Today, knowledge-workers have portals that are maintained and updated to provide custom views of corporate information, external data, etc. However, this is still very limiting because knowledge-workers are completely helpless if nothing dynamically and intelligently connects relevant information in the context of their task with information that users have access to.

If a knowledge-worker does not see a link to a relevant piece of information on his or her portal, or if a friend or colleague does not email him or her the link, the information gets dropped; information does not connect with or adapt to the user context or the context in which it is displayed. Likewise, it is not enough to just notify a user that new data for an entire portal is available and shove it down to their local hard drive. It lacks a customizable presentation with context-sensitive alert notifications.

The Semantic Web suffers from the same limitations as Today’s Web when it comes to context-sensitivity. On the Semantic Web, users are likewise at the mercy of information authors. The Semantic Web itself will be authored, but the authoring will include semantics. As a result, users are still largely on their own to locate and evaluate the relevance of available information. The Semantic Web, as a standalone entity, will not be able to make these dynamic connections with other information sources.

Time-Sensitivity

Today’s Web lacks time-sensitivity. The Web platform (e.g., browser) is a dumb piece of software that merely presents information, without any regard to the time-sensitivity of the information. The user is left to infer time sensitivity or do without it. This results in a huge loss in productivity because the Web platform cannot make time-sensitive connections in real-time. While some Web sites focus on presenting time-sensitive information, for example, by indexing information past a predetermined date, the Web browser itself has no notion of time-sensitivity. Instead, it is left to individual Web sites to include time-sensitivity in the information they display in their own island. In other words, there is no axis of time on a Web link.

The Semantic Web, like Today’s Web, also does not address time-sensitivity. A Semantic Web can have semantic links that do not internalize time. This is largely because the Semantic Web implicitly has no notion of software Web services that address context and time-sensitivity.

Automatic and Intelligent Discoverability

Today’s Web lacks automatic and intelligent discoverability of newly created information. There is currently no
way to know what Web sites started anew today or yesterday. Unless the user is notified or the user serendipitously discovers a new site when he or she does a search, he or she might not have any clue as to whether there are any new Web sites or pages. The same problem exists in enterprises. On Intranets, knowledge-workers have no way of knowing when new Web sites come up unless informed via some external means. The Web platform itself has no notion of announcements or discovery. In addition, there is no context-sensitive discovery to determine new sites or pages within the context of the user’s task or current information space.

[0030] The Semantic Web, like Today’s Web, does not address the lack of automatic discoverability. Semantic Web sites suffer from the same problem—users either will have to find out about the existence of new information sources from external sources or through personal discovery when they perform a search.

Dynamic Linking

[0031] Today’s Web employs a pure network or graph “data structure” for its information model. Each Web page represents a node in the network and each page can contain links to other nodes in the network. Each link is manually authored into each page. This has several problems. First, it means that the network needs to be maintained for it to have continuous value. If Web pages are not updated or if Web page or site authors do not have the discipline to add links to their pages based on relevance, the network loses value. Today’s Web is essentially prone to having dead links, old links, etc. Another problem with a pure network or graph information model is that the information consumer is at the mercy of—rather than in control of—the presentation of the Web page or site. In other words, if a Web page or site does not contain any links, the user has no recourse to find relevant information. Search engines are of little help because they merely return pages or nodes into the network. The network itself does not have any independent or dynamic linking ability. Thus, a search engine can easily return links to Web pages that themselves have no links or dead, stale or irrelevant links. Once users obtain search results, they are on their own and are completely at the mercy of whether the author of the returned pages inserted relevant, time-sensitive links into the page.

[0032] The Semantic Web suffers from the same problem as Today’s Web because the Semantic Web is merely Today’s Web plus semantics. Even though users will be able to navigate the network semantically (which they cannot currently do with the Web), they will still be at the mercy of how the information has been authored. In other words, the Semantic Web is also dependent on the discipline of the authors and hence suffers from the same aforementioned problems of Today’s Web. If the Semantic Web includes pages with ontologies and metadata, but those pages are not well maintained or do not include links to other relevant sources, the user will still be unable to obtain current links and other information. The Semantic Web, as currently contemplated, will not be a smart, dynamic, self-authoring, self-healing network.

User-Controlled Navigation and Browsing

[0033] With Today’s Web, the user has no control over the navigation and browsing experience, but rather is completely at the mercy of a Web page and how it is authored with links (if any). As shown with reference to prior art FIG. 1-3, Today’s Web consists of “dumb links,” or statically authored generic links that are wholly dependent on continuous maintenance to be navigable.

[0034] The Semantic Web suffers from a similar problem as Today’s Web in that there is no user-controlled browsing. Instead, as shown with reference to prior art FIG. 2, the Semantic Web consists of “dumb links,” further including semantic information and metadata. However, the Semantic Web links remain equally dependent on continuous maintenance to be navigable.

Non-HTML and Local Document Participation in the Network

[0035] Another problem with Today’s Web is the requirement that only documents that are authored as HTML can participate in the Web, in addition to the fact that those documents have to contain links. The implication is that other information objects like non-HTML documents (e.g., PDF, Microsoft Word, PowerPoint, and Excel documents, etc.)—especially those on users’ hard drives—are excluded from the benefits of linking to other objects in the network. This is very limiting, especially since there might be semantic relevance between information objects that are not HTML and which do not contain links.

[0036] Furthermore, search engines do not return results for the entire universe of information since vast amount of content available on the web is inaccessible to standard web crawlers. This includes, for example, content stored in databases, unindexed file repositories, subscription sites, local machines and devices, proprietary file formats (such as Microsoft Office documents and email), and non-text multimedia files. These form a vast constellation of inaccessible matter on the Internet, referred to as “the invisible Intranet” inside corporations. Today’s Web servers do not provide web crawler tools that address this problem.

[0037] The Semantic Web also suffers from this limitation. It does not address the millions of non-HTML documents that are already out there, especially those on users’ hard drives. The implication is that documents that do not have RDF metadata equivalents or proxies cannot be dynamically linked to the network.

Flexible Presentation that Smartly Conveys the Semantics of the Information being Displayed

[0038] Today’s Web does not allow users to customize or “skin” a Web site or page. This is because Today’s Web servers return information that is already formatted for presentation by the browser. The end user has no flexibility in choosing the best means of displaying the information—based on different criteria (e.g., the type of information, the available amount of real estate, etc.).

[0039] The Semantic Web does not address the issue of flexible presentation. While a semantic Web site conceptually employs RDF and ontologies, it still sends HTML to the browser. Essentially, the Semantic Web does not provide for specific user empowerment for presentation. As such, a Semantic Web site, viewed by Today’s Web platform, will still not empower the user with flexible presentation. Moreover, despite industry movement towards XML, only a new platform can dictate that data will be separated from presentation and define guidelines for making the data programmable. Authors building content for the Semantic Web either return XML and avoid issues with presentation entirely, or focus their efforts on a single presentation style (vertical industry
scenario) for rendering. Neither approach allows the Semantic Web to achieve an optimum degree of knowledge distribution.

Logic, Inference and Reasoning

Because today's Web does not have any semantics, metadata, or knowledge representation, computers cannot process Web pages using logic and inference to infer new links, issues notifications, etc. Today's Web was designed and built for human consumption, not for computer consumption. As such, today's Web cannot operate on the information fabric without resorting to brittle, unreliable techniques such as screen scraping to try to extract metadata and apply logic and inference.

While the Semantic Web conceptually uses metadata and meaning to provide Web pages and sites with encoded information that can be processed by computers, there is no current implementation that is able to successfully achieve this computer processing and which illustrates new or improved scenarios that benefit the information consumer or producer.

Flexible User-Driven Information Analysis

Today's Web lacks user-driven information analysis. Today's Web does not allow users to display different "views" of the links, using different filters and conditions. For example, Web search engines do not allow users to test the results of searches under different scenarios. Users cannot view results using different pivots such as information type (e.g., documents, email, etc.), context (e.g., "Headlines," "Best Bets," etc.), category (e.g., "wireless," "technology," etc.).

While providing a greater degree of flexible information analysis, the Semantic Web does not describe how the presentation layer can interact with the Web itself in an interactive fashion to provide flexible analysis.

Flexible Semantic Queries

Today's Web only allows text-based queries or queries that are tied to the schema of a particular Web site. These queries lack flexibility. Today's Web does not allow a user to issue queries that approximate natural language or incorporate semantics and local context. For example, a query such as "Find me all email messages written by my boss or anyone in research and which relate to this specification on my hard disk" is not possible with today's Web.

By employing metadata and ontologies, the conceptual Semantic Web allows a user to issue more flexible queries than today's Web. For example, users will be able to issue a query such as "Find me all email messages written by my boss or anyone in research." However, users will not be able to incorporate local context. In addition, the Semantic Web does not define an easy manner with which users will query the Web without using natural language. Natural language technology is an option but is far from being a reliable technology. As such, a query user interface that approximates natural language yet does not rely on natural language is required. The Semantic Web does not address this.

Read Write Support

Today's Web is a read-only Web. For example, if users encounter a dead link (e.g., via the "404" error), they cannot "fix" the link by pointing it to an updated target that might be known to the user. This can be limiting, especially in cases where users might have important knowledge to be shared with others and where users might want to have input as to how the network should be represented and evolve.

While the Semantic Web conceptually allows for read/write scenarios as provided by independent participating applications, there is no current implementation that provides this ability.

Annotations

Today's Web has no implicit support for annotations. And while some specific Web sites support annotations, they do so in a very restricted and self-contained way. Today's Web medium itself does not address annotations. In other words, it is not possible for users to annotate any link with their comments or additional information that they have access to. This results in potential information loss.

While the Semantic Web conceptually allows for annotations to be built into the system subject to security constraints, there is no current implementation that provides this ability.

"Web of Trust"

Today's Web lacks seamless integration of authentication, access control, and authorization into the Web, or what has been referred to as a "Web of Trust." With a Web of Trust, for example, users are able to make assertions, fix and update links to the Web and have access control restrictions built in for such operations. On today's Web, this lack of trust also means that Web services remain independent islands that must implement a proprietary user subscription authorization, access control or payment system. Grand schemes for centralizing this information on a third party servers meet with consumer and vendor distrust because of privacy concerns. To gain access to rich content, asset users must log in individually and provide identity information at each site.

While the Semantic Web conceptually allows for a Web of Trust, there is no current implementation that provides for this ability.

Information Packages (Blenders)

Neither today's Web nor the Semantic Web allows users to deal with related semantic information as a whole unit by combining characteristics of potentially divergent semantic information to produce overlapping results (for example, like creating a custom, personal newspaper or TV channel).

Context Templates

Neither today's Web nor the Semantic Web allows users to independently create and map to specific and familiar semantic models for information access and retrieval.

User-Oriented Information Aggregation

Today's Web lacks support for user-oriented information aggregation. The user can only access one Web site or one search engine at a time, within the context of one browsing session. As such, even if there is context or time-sensitive information on other information sources that relate to the information that the user is currently viewing, those sources cannot be presented in a holistic fashion in the current context of the user's task.
The Semantic Web also suffers from a lack of user-oriented information aggregation. The medium itself is an extension of Today’s Web. As such, users will still access one site or one search engine at a time and will not be able to aggregate information across information repositories in a context or time-sensitive manner.

Given the growing demand for “knowledge at your fingertips” as well as the deficiencies in Today’s Web and the conceptual Semantic Web, many of which are noted above, there is a need for a new and comprehensive system and method of knowledge retrieval, management and delivery. Preferred embodiments of the present invention are directed in part to a semantically integrated knowledge retrieval, management, delivery and/or presentation system. Preferred embodiments of the present invention and system include several additional improved features, enhancements and/or properties, including, without limitation, semantic advertisements, spider RSS integration, pivot views, watch lists, context extraction methods, context ranking methods, client duplication management methods, a server data and index model, improved metadata indexing methods, adaptive ranking methods, and content transformation methods.

The explosive growth of digital information is increasingly impeding knowledge-worker productivity due to information overload. Online information is virtually doubling every year and/or most of that information is unstructured—usually in the form of text. Traditional search engines have been unable to keep up with the pace of information growth primarily because they lack the intelligence to “understand,” semantically process, mine, infer, connect, and/or contextually interpret information in order to transform it to—and/or expose it as—knowledge. Furthermore, end-users want a simple yet powerful user-interface that allows them to flexibly express their context and/or intent and/or be able to “ask” natural questions on the one hand, but which also has the power to guide them to answers for questions they wouldn’t know to ask in the first place. Today’s search interfaces, while easy-to-use, do not provide such power and/or flexibility.

Now that the Web has reached critical mass, the primary problem in information management has evolved from one of access to one of intelligent retrieval and/or filtering. Computer users are now faced with too much information, in too many formats and/or via multiple applications, with little or no help in transforming that information into useful knowledge.

Search engines such as Google™ provide some help in filtering information by indexing content based on keywords. Google™, in particular, has gone a step further by mining the hypertext links in Web pages in order to draw inferences of relevance based on page popularity. These techniques, while helpful, are far from sufficient and/or still leave end-users with little help in separating wheat from chaff. The primary reason for this is that current search engines do not truly “understand” what they index or what users want. Keywords are very poor approximations of meaning and/or user intent. Furthermore, popularity, while useful, is no guarantee of relevance: Popular garbage is still garbage.

Furthermore, knowledge has multiple axes, and/or search is only one of those axes. Knowledge-workers also wish to discover information they might not know they need ahead of time, share information with others (especially those that have similar interests), annotate information in order to provide commentary, and/or have information presented to them in a way that is contextual, intuitive, and/or dynamic—allowing for further (and/or potentially endless) exploration and/or navigation based on their context. Even within the search axis, there are multiple sub-axes, for instance, based on time-sensitivity, semantic-sensitivity, popularity, quality, brand, trust, etc. The axis of choice depends on the scenario at hand.

Search engines are appropriately named because they focus on search. However, merely improving search quality without reformulating the core goal of search will leave the information overload problem unaddressed.

BRIEF DESCRIPTION OF THE DRAWINGS

The preferred and alternative embodiments of the present invention are described in detail below with reference to the following drawings.

FIG. 1-1 is a table showing the technology layers of Today’s Web;
FIG. 1-2 is a table showing the technology layers of the conceptual Semantic Web;
FIG. 1-3 is a diagram showing user navigation to links in Today’s Web;
FIG. 2 is a diagram showing user navigation to links in the conceptual Semantic Web;
FIG. 3 shows the technology platform stacks of Today’s Web and the Information Nervous System of the present invention;
FIG. 4 is a diagram showing an overview of the system of the present invention;
FIG. 5 is a diagram showing the end-to-end system architecture for the Information Nervous System of the present invention;
FIG. 6 is a diagram showing the system architecture for the Knowledge Integration Server (KIS) of the Information Nervous System of the present invention;
FIG. 7 is a comparison between the high-level descriptive platform layers of Today’s Web and the equivalents (where applicable) in the Information Nervous System of the present invention;
FIG. 8 illustrates the preferred embodiment of the Information Nervous System and illustrates the heterogeneous, cross-platform context for the present invention;
FIG. 9 is an exemplar pane of a Breaking News Agent user interface;
FIG. 10-1 shows the Agent schema of the preferred embodiment of the present invention;
FIG. 10-2 shows the AgentTypeIDs of the preferred embodiment of the present invention;
FIG. 10-3 shows the AgentQueryTypeIDs of the preferred embodiment of the present invention;
FIG. 10-4 illustrates sample semantic queries that correspond to Agent names showing how server-side Agents are preferably configured on the KIS of the present invention;
FIG. 11 is a diagram showing an overview of the KIS of the present invention;
FIG. 12 is a diagram showing a sample Semantic Network directed towards an enterprise situation in accordance with the present invention;
FIG. 13-1 is a table showing the preferred schema of the Object type in accordance with the present invention;
FIG. 13-2 shows the SemanticLinks table of the present invention;
FIG. 14 is a table showing predicate type IDs of the preferred embodiment of the present invention;
FIG. 15 is a table showing the preferred user object schema made in accordance with the present invention;

FIG. 16-1 is a table showing MailingAddressTypeIDs preferably associated with the User (person) object schema;

FIG. 16-2 is a table of the preferred category object schema made in accordance with the present invention;

FIG. 16-3 is a table of the preferred document object schema made in accordance with the present invention;

FIG. 17-1 shows the Print Media Type IDs of the preferred embodiment;

FIG. 17-2 shows the preferred FORMATTYPEID;

FIG. 18 shows the preferred email message list object schema made in accordance with the present invention;

FIGS. 18-2 and 18-3 are exemplar tables showing the email distribution list and email public folder object schemas, respectively, of a preferred embodiment of the present invention;

FIG. 19-1 shows the preferred PublicFolderTypeID of the present invention;

FIG. 19-2 shows the preferred event object schema message list object schema made in accordance with the present invention;

FIG. 19-3 shows the events types of a preferred embodiment of the present invention;

FIG. 20-1 shows the preferred media object schema message list object schema made in accordance with the present invention;

FIG. 20-2 shows the media types of a preferred embodiment of the present invention;

FIGS. 21-23 illustrate additional samples showing how objects are categorized and utilized in the preferred embodiment of the present invention;

FIG. 24 is an object graph showing mapping of raw email XML metadata to the Semantic Network according to the present invention;

FIG. 25 shows a sample user interface illustrating an information object displayed in the Information Agent Results pane;

FIG. 26 shows an example of a balloon popup associated with an Intrinsic Semantic Link showing an email sample according to the present invention;

FIG. 27 shows an example of a balloon popup associated with a Verb user interface according to the present invention;

FIG. 28 shows an example of a balloon popup associated with a Deep Information Mode user interface according to the present invention;

FIGS. 29-31 provide exemplar balloon popup menus associated with the Smart Lens feature of an Information Agent according to the present invention;

FIG. 32 shows a sample of a variant of the balloon popup menu of FIG. 71 showing the relatedness measure of the two objects;

FIGS. 33-35 show sample tables illustrating the behaviors and relational contains objects types predicates when using Smart Lenses;

FIG. 36 is a user interface sample illustrating semantic results Player/Preview Control according to the present invention;

FIG. 37 is a user interface sample showing the semantic results of a Blender;

FIGS. 38 and 39 illustrate exemplar functionality mappings of the present invention;

FIG. 40 illustrates a user interface showing Agent results and corresponding Context Palettes according to the present invention;

FIG. 41 shows a sample Smart Recommendations popup context Results Pane according to the present invention;

FIG. 42 is a table showing the technology layers of the Information Nervous System of the present invention;

FIG. 43 illustrates dynamic linking and user-controlled navigation and browsing according to a preferred embodiment of the present invention;

FIG. 44 is a partial screenshot overview of a Patent Examiner using the preferred embodiment in a prior art search, a screenshot of where “Magnetic Resonance Imaging” occurs in a Pharmaceuticals taxonomy;

FIG. 45 shows the Shareable Smart Request System Interaction, which is the binary document format that encapsulates the SQML buffer with the smart request and also illustrates how the extension handler opens a document;

FIG. 46 is a Diagram Illustrating the Text-to-Speech Object Skin and shows an illustration of an email message being rendered via a text-to-speech object skin;

FIG. 47 is a Diagram Illustrating a Text-to-Speech Request Skin;

FIG. 48 is a Diagram Illustrating Knowledge Modeling for a Pharmaceuticals Company Example;

FIG. 49 is a Diagram Illustrating Client Component Integration and Interaction Workflow;

FIG. 50 shows how the server-side semantic query processor processes incoming semantic queries (represented as SQML);

FIG. 51 illustrates how a user would configure a profile (to create a profile, the user will use the “Create Profile Wizard” and the profile can then be modified via a property sheet as shown in other Figures);

FIG. 52 illustrates a Watch Window displaying Filtered Smart Requests (e.g., Headlines on Wireless). FIG. 20 is an Illustration of the Watch Window with a Current Smart Request Title (e.g., “Breaking News”);

FIG. 53 illustrates a semantic thread object and its semantic links;

FIG. 54 is an additional screen shot illustrating the interactions, options and operations as described in the Detailed Description;

FIG. 55 is a sample semantic image for Pharmaceuticals/Biotech industry (DNA helix);

FIG. 55-2 is an illustration of a semantically appropriate image visualization for the Breaking News context template;

FIG. 55-3 is a Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Headlines);

FIG. 55-4 is a Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Two people working at a desk);

FIG. 55-5 illustrates a semantic “Newsmaker” Visualization or Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc.;

FIG. 55-6 illustrates a semantic “Upcoming Events” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc.;

FIG. 55-7 is a Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Petri Dish);
FIG. 55-8 illustrates a semantic “History” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc.;

FIG. 56-1 illustrates a semantic Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Spacecraft);

FIG. 56-2 illustrates a “Best Buys” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc.;

FIG. 56-3 illustrates a semantic Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Coffee);

FIG. 56-4 illustrates a semantically appropriate Sample Image for “Classics” for smart hourglass, interstitial page, transition effects, background chrome, etc. (Car);

FIG. 56-5 illustrates a semantically appropriate “Recommendation” Visualization—Sample Image for the context/application elements of smart hourglass, interstitial page, transition effects, background chrome, etc. (Thumbs up);

FIG. 56-6 illustrates a semantic “Today” Visualization—Sample Image for the elements smart hourglass, interstitial page, transition effects, background chrome, etc.;

FIG. 56-7 illustrates a semantic “Annotated Items” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc.;

FIG. 56-8 illustrates a semantic “Annotations” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc.;

FIG. 57-1 illustrates a semantic “Experts” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc.;

FIG. 57-2 illustrates a semantic “Places” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc.;

FIG. 57-3 illustrates a semantic “Blenders” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc.;

FIGS. 57-4 through 57-8 illustrate semantic Visualizations for the following Information Object Types, respectively: Documents, Books, Magazines, Presentations, and Resumes;

FIGS. 58-1 through 58-8 illustrate semantic Visualizations for the following Information Object Types, respectively: Spreadsheet, Text, Web pages, White Papers, Email, Email Annotations, Email Distribution Lists, Events, Meetings, Multimedia, Online Courses, People, Customers, and Users;

FIGS. 59-1 through 59-7 illustrate a semantic “Timeline” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc.;

FIG. 60 is an Ontology Objects Table Data and Index Model according to an embodiment of the invention;

FIG. 61 is an Ontology Semantic Links Table Data and Index Model according to an embodiment of the invention;

FIGS. 62 through 65 are screenshots illustrating principles of at least one embodiment of the invention;

FIG. 66 is a Table Showing Semantic Search Qualifiers and Corresponding Predicates according to an embodiment of the invention;

FIG. 67 is a screenshot illustrating principles of at least one embodiment of the invention;

FIG. 68 is a block diagram of a method for implementing semantic advertisements in an internet browser;

FIG. 69 is a block diagram of a method for integrating HTTP metadata and RSS metadata in an information server;

FIG. 70 is a block diagram of a method for dynamically making input suggestions based upon prior user input;

FIG. 71 is a block diagram of a method for presenting time sensitive information to a user;

FIG. 72 is a block diagram for a method of presenting knowledge community statistics at a client user interface, in accordance with an embodiment of the invention;

FIG. 73 is a block diagram of a method for allowing users to remove duplicative presented information;

FIGS. 74-1 through 74-2 illustrate a documents table data and index model, in accordance with an embodiment of the invention;

FIG. 75 is an objects table data and index model, in accordance with an embodiment of the invention;

FIG. 76 is a semantic links table data and index model, in accordance with an embodiment of the invention;

FIG. 77 is a composite index table model, in accordance with an embodiment of the invention;

FIG. 78 is a block diagram for a method of quickly indexing data contained in a metadata feed, in accordance with an embodiment of the invention;

FIG. 79 is a block diagram for a method of adjusting threshold values that are used to determine the most relevant objects in a given context, in accordance with an embodiment of the invention;

FIG. 80 is a method for indexing and retrieving semantically relevant documents, in accordance with an embodiment of the invention;

FIG. 81 is a method for highlighting semantically relevant keywords in displayed documents resulting from semantic searches, in accordance with an embodiment of the invention;

FIG. 82 is a block diagram showing methods for creating and managing multiple types of knowledge communities, in accordance with an embodiment of the invention;

FIG. 83 is a block diagram of a method for providing user feedback on the available knowledge communities, in accordance with an embodiment of the invention;

FIG. 84 illustrates a method of using semantic sounds to notify a user regarding the arrival of news in accordance with an embodiment of the invention;

FIG. 85 is a method of tracking and presenting multiple lists of categories to a client user as the categories evolve over time, in accordance with an embodiment of the invention;

FIG. 86 is a block diagram of a method of semantically indexing and retrieving non-text data, in accordance with an embodiment of the invention;

FIG. 87 is a block diagram of a method for providing ontology feedback in accordance with an embodiment of the invention;

FIG. 88 is a block diagram of a method for advanced semantic searching in accordance with an embodiment of the invention;

FIG. 89 is a block diagram of a method for handling floating text in an RSS feed;

FIG. 90 is a block diagram of a method for extracting a semantic query from an image, in accordance with an embodiment of the invention;
FIG. 91 is a block diagram for a method for improving ontology development in accordance with an embodiment of the invention;

FIG. 92 is a block diagram of a method for developing and maintaining ontologies, in accordance with an embodiment of the invention;

FIG. 93 is a block diagram for a method for semantic question answering in accordance with an embodiment of the invention;

FIG. 94 is a block diagram of a method of coupling natural language with semantic language queries in accordance with an embodiment of the invention;

FIG. 95 is a block diagram of a method for categorizing extracted concepts from a URI, in accordance with an embodiment of the invention;

FIG. 96 is a block diagram of a method for establishing context queries, in accordance with an embodiment of the invention;

FIG. 97 is a block diagram of a method for extracting concepts from disparate sources, in accordance with an embodiment of the invention;

FIG. 98 is a block diagram of a method for reorganizing independent website data according to semantic strength, in accordance with an embodiment of the invention;

FIG. 99 is a block diagram of a method for semantic analysis on the client, in accordance with an embodiment of the invention;

FIG. 100 is a block diagram for a method of generating information on experts, interest groups, or newsmakers, in accordance with an embodiment of the invention;

FIG. 101 is a method for adding new ontologies to a client semantic browser, in accordance with an embodiment of the invention;

FIG. 102 illustrates a method for using field and category specific searches to supplement keyword searches, in accordance with an embodiment of the invention;

FIG. 103 is a method for creating weighted indices and searching thereon, in accordance with an embodiment of the invention;

FIG. 104 illustrates defined knowledge filters/types, in accordance with an embodiment of the invention;

FIG. 105 is a sample illustration of a user-defined hierarchy for storing personal digital photos;

FIG. 106 illustrates sample fields of the Knowledge Domain Entry data structure returned by the KDS Web in accordance with an embodiment of the invention;

FIG. 107 illustrates the schema and/or sample fields of a KDS result, in accordance with an embodiment of the invention;

FIG. 108 illustrates the representation of a semantic network in the KIS, in accordance with an embodiment of the invention;

FIG. 109 illustrates the end-to-end architecture of one embodiment of the invention;

FIG. 110 illustrates the representation of a semantic network in accordance with an embodiment of the invention;

FIGS. 111 and/or 112 illustrate sample queries of one embodiment of the invention;

FIG. 113 is an illustrative example of a pagination pipeline architecture diagram in accordance with an embodiment of the invention;

FIG. 114 is a block diagram illustrating General Content Transformation Pipeline Architecture in accordance with an embodiment of the invention;

FIG. 115 shows an illustration of the implementation of the feature, the well-known knowledge stack, and/or how this applies to this model in accordance with an embodiment of the invention;

FIG. 116 illustrates a system in accordance with an embodiment of the invention;

FIG. 117 illustrates a pie chart of live search results by publisher;

FIG. 118 is a sample illustration of a bar chart of search results by publisher;

FIG. 119 is a screen shot of the Display Options window in accordance with an exemplary embodiment of the invention;

FIG. 120 is a screen shot of the Live Mode Options window in accordance with an exemplary embodiment of the invention;

FIG. 121 is a screen shot of the Security Settings window in accordance with an exemplary embodiment of the invention;

FIG. 122 is a screen shot of the Proxy Settings window in accordance with an exemplary embodiment of the invention;

FIG. 123 is a screen shot of the Knowledge Directories window in accordance with an exemplary embodiment of the invention;

FIG. 124 is a screen shot of the Knowledge Directories browser window in accordance with an exemplary embodiment of the invention;

FIG. 125 is a line chart illustrating long term decline in R & D Productivity v. NME Output per R & D dollar spent;

FIGS. 126 and 127 illustrate Data Growth Curves in Intractable v. Tractable conditions in accordance with an exemplary embodiment of the invention;

FIG. 128 illustrates the schema and/or sample fields categories and inputs that get added to the semantic network, in accordance with an embodiment of the invention;

FIG. 129 is a representation of a semantic network in accordance with an embodiment of the invention;

FIG. 130 illustrates the end-to-end architecture of one embodiment of the invention;

FIG. 131 illustrates the representation of a semantic network in accordance with an embodiment of the invention;

FIG. 132 is a representation of below illustrates the Nervana Content Framework in accordance with an embodiment of the invention;

FIG. 133 illustrates the end-to-end process cycle of Nervana Discovery Spaces;

FIG. 134 illustrates Nervana’s technology licensing partners;

FIG. 135 illustrates Nervana’s Evidence-Based Medicine—a new business process for matching patient health records to diagnostic information;

FIGS. 136-137 illustrate Nervana’s Semantic Discovery and Collaboration Platform in accordance with an embodiment of the invention;

FIGS. 138-139 illustrate product illustrations for Nervana Social Discovery in accordance with an embodiment of the invention;

FIG. 140 is an exemplary embodiment for Nervana’s Social Discovery in accordance with an embodiment of the invention;

FIG. 141 is a schematic diagram of Nervana’s Ontology Automation Model Social Discovery in accordance with an embodiment of the invention;
[0220] FIGS. 142-143 are schematic diagrams of Next-Generation Clinical Trial Services Proposal in accordance with an embodiment of the invention;

[0221] FIG. 144 is a schematic diagram of Next-Generation Collaboration Platform in accordance with an embodiment of the invention;

[0222] FIG. 145 is a table comparison of technology uniqueness;

[0223] FIG. 146 illustrates the schema and/or sample fields categories and inputs that get added to the semantic network, in accordance with an embodiment of the invention;

[0224] FIG. 147 is a schematic diagram of Business Process with Nervana’s Ad Engine in accordance with an embodiment of the invention;

[0225] FIG. 148 is a schematic diagram of Nervana’s Ad Engine Workflow in accordance with an embodiment of the invention;

[0226] FIG. 149 is a schematic diagram of Nervana’s Site Ranking Attributes in accordance with an embodiment of the invention;

[0227] FIG. 150 is a schematic diagram of Nervana’s Content Sources in accordance with an embodiment of the invention;

[0228] FIG. 151 is a schematic diagram of Nervana’s Talent Engine AI Model Components in accordance with an embodiment of the invention.

[0229] The preferred and alternative embodiments of the present invention are described in detail below with reference to the following drawings.

DOCUMENTS INCORPORATED BY REFERENCE

[0230] The Appendix attached hereto and referenced herein is incorporated by reference. This Appendix includes exemplar code illustrating a preferred embodiment of the present invention.

CONTENTS OF DETAILED DESCRIPTION OF THE INVENTION

A. DEFINITIONS
B. OVERVIEW
C. SYSTEM ARCHITECTURE AND TECHNOLOGY CONSIDERATIONS
D. SYSTEM COMPONENTS AND OPERATION

E. SCENARIOS

5. PROVIDING CONTEXT IN THE PRESENT INVENTION

6. PROPERTY BENEFITS OF THE PRESENT INVENTION

D. SYSTEM COMPONENTS AND OPERATION

[0240] 1. AGENCIES AND AGENTS

[0241] a. Agencies

[0242] b. Agents

[0243] 2. KNOWLEDGE INTEGRATION SERVER

[0244] a. Semantic Network

[0245] b. Semantic Data Gatherer

[0246] c. Semantic Network Consistency Checker

[0247] d. Inference Engine

[0248] e. Semantic Query Processor

[0249] f. Natural Language Parser

[0250] g. Email Knowledge Agent

[0251] h. Knowledge Domain Manager

[0252] i. Other Components

[0253] 3. KNOWLEDGE BASE SERVER

[0254] 4. INFORMATION AGENT (SEMANTIC BROWSER PLATFORM)

[0255] a. Overview

[0256] b. Client Configuration


[0258] d. Client Framework

[0259] e. Semantic Query Document

[0260] f. Semantic Environment

[0261] g. Semantic Environment Manager

[0262] h. Environment Browser (Semantic Browser or Information Agent™)

[0263] i. Additional Application Features

[0264] 1. EXAMPLES OF SEMANTIC QUERIES UTILIZING THE PRESENT INVENTION

[0265] 2. BUSINESS PROBLEMS

[0266] 3. SITUATIONS

[0267] 4. INFORMATION AGENT (SEMANTIC BROWSER PLATFORM)

[0268] 5. PROVIDING CONTEXT IN THE PRESENT INVENTION

[0269] 6. PROPERTY BENEFITS OF THE PRESENT INVENTION

[0270] DETAILED DESCRIPTION OF THE INVENTION

[0271] The present invention is directed in part to an integrated and seamless implementation framework and resulting medium for knowledge retrieval, management, delivery and presentation. The system includes a server comprised of several components that work together to provide context and time-sensitive semantic information retrieval services to clients operating a presentation platform via a communication medium. The server includes a first server component that is responsible for adding and maintaining domain-specific semantic information or intelligence. The first server component preferably includes structure or methodology directed to providing the following: a Semantic Network, a Semantic Data Gatherer, a Semantic Network Consistency Checker, an Inference Engine, a Semantic Query Processor, a Natural Language Parser, an Email Knowledge Agent and a Knowledge Domain Manager. The server includes a second server component that hosts domain-specific information that is used to classify and categorize semantic information. The first and second server components work together and may be physically integrated or separate.

[0272] Within the system, all objects or events in a given hierarchy are active agents semantically related to each other and representing queries (comprised of underlying action code) that return data objects for presentation to the client
according to a predetermined and customizable theme or “Skin.” This system provides various means for the client to customize and “blend” Agents and the underlying related queries to optimize the presentation of the resulting information.

[0280] The end-to-end system architecture of the present invention provides multiple client access means of communication between diverse knowledge information sources via an independent Semantic Web platform or via a traditional Web portal (e.g., Today’s Web access browser) as modified by the present invention providing additional SDK layers that enable programmatic integration with a custom client.

[0281] The methodology of the present invention is directed in part to the operational aspects of the entire system, including the retrieval, management, delivery and presentation of knowledge. This preferably includes securing information from information sources, semantically linking the information from the information sources, maintaining the semantic attributes of the body of semantically linked information, delivering requested semantic information based upon user queries and presenting semantic information according to customizable user preferences. Alternative embodiments of the methodology of the present invention are directed to the operation of Agents representing queries that are used with server-side and client-side applications to enable efficient, inferential-based queries producing semantically relevant information.

[0282] The present invention is directed in part to a semantically integrated knowledge retrieval, management, delivery and presentation system, as is more fully described in my co-pending parent application (U.S. application Ser. No. 10/179,651 filed Jun. 24, 2002). The present invention and system includes several additional improved features, enhancements and/or properties, including, without limitation, Entities, Profiles and Semantic Threads, as are more fully described in the Detailed Description below.

A. Definitions


[0284] Agency. A named instance of a Knowledge Integration Server (KIS) that is the semantic equivalent of a website.

[0285] Agency Directory. A directory that stores metadata information for Agencies and allows clients to add, remove, search, and browse Agencies stored within. Agencies can be published on directories like LDAP or the Microsoft Active Directory. Agencies can also be published on a proprietary directory built specifically for Agencies.

[0286] Agent. A semantic filter query that returns XML information for a particular semantic object type (e.g., documents, email, people, etc.), context (e.g., Headlines, Conversations, etc.) or Blender.

[0287] Blender™ or Compound Agent™. Trademarked name for an Agent that contains other Agents and allows the user (in the case of client-side blenders) or the Agency administrator (in the case of server-side blenders) to create queries that generate results that are the union or intersection of the results of their contained Agents. In the case of client-side blenders, the results can be generated using different views (showing each

Agent in the blender in a different frame, showing all the objects of a particular object type across the contained Agents, etc.)

[0288] Breaking News Agent™. Trademarked name for a Smart Agent that users specially tag as being indicative of time-criticality. Users can tag any Smart Agent as a Breaking News Agent. This attribute is then stored in users’ Semantic Environment. A Breaking News Agent preferably shows an alert if there is breaking news related to any information being displayed.

[0289] Default Agent™. Trademarked name for standardized, non-user modifiable Agents presented to the user.

[0290] Domain Agent™. Trademarked name for an Agent that belongs to a semantic domain. It is initialized with an Agent query that includes reference to the “categories” table.

[0291] Dumb Agent™. Trademarked name for an Agent that does not have an Agency and which refers to local information (on a local hard drive), on a network share or on a Web link or URL. Dumb Agents are used to essentially load information items (e.g., documents) from a non-smart sandbox (e.g., the file system or the Internet) to a smart sandbox (the Information Nervous System via the Information Agent (semantic browser)).

[0292] Email Agent™ (or Email Knowledge Agent™). Trademarked names for a Public Agent used to publish or annotate information and share knowledge on an Agency.

[0293] Favorite Agent™. Trademarked name for Agents that users indicate they like and access often.

[0294] Public Agent™. Trademarked name for Agents that are created and managed by the system administrator.

[0295] Private or Local Agents™. Trademarked names for Agents that are created and managed by users.

[0296] Search Agent™. Trademarked name for a Smart Agent that is created by searching the semantic environment with keywords or by searching an existing Smart Agent, in order to invoke an additional, text-based query filter on the Smart Agent.

[0297] Simple or Standard Agent™. Trademarked names for Standalone Agents that encapsulate structured, non-semantic queries (e.g., from the local file system or data source).

[0298] Smart Agent™. Trademarked name for a standalone Agent that encapsulates structured, semantic queries that refers to an Agency via its XML Web Service.

[0299] Special Agent™. Trademarked name for a Smart Agent that is created based on a Context Template.

[0300] Agent Discovery. The property of the information medium of the present invention that allows users to easily and automatically discover new server-side Agents or client-side Agents created by others (friends or colleagues). Also see “Discoverability.”

[0301] Annotations. Notes, comments, or explanations that are used to add personal context to an information object. In the preferred embodiment, annotations are email messages that are linked to the object they qualify, and which can have attachments (just like regular email messages). In addition, annotations are first class information objects in the system and as such can be annotated themselves, thereby resulting in threaded annotations or a tree of annotations with the initial object as the root.
[0302] Application Programming Interface (API). Defines how software programmers utilize a particular computer feature. APIs exist for windowing systems, file systems, database systems, networking systems, and other systems.

[0303] Calendar Access Protocol (CAP). Internet protocol that permits users to digitally access a calendar store based on the iCalendar standard.

[0304] Compound Agent Manager™. Trademarked name for an Agency component that programmatically allows the user to create and delete Compound Agents and to manage them by adding and deleting Agents.

[0305] Context. Information surrounding a particular item that provides meaning and otherwise assists the information consumer in interpreting the item as well as finding other relevant information related to the item.

[0306] Context Results Pane. A Results Pane that displays results for context-based queries. These include results for Context Palettes, Smart Lenses, Deep Information, etc. See “Results Pane.”

[0307] Context-Sensitivity. The property of an information medium that enables it to intelligently and dynamically perceive the context of all the information it presents and to present additional, relevant information given that context. A context-sensitive system or medium understands the semantics of the information it presents and provide appropriate behaviors (proactive and reactive) based on the user’s actions in order to present information in its proper context (both intrinsically and relationally).

[0308] Context Template™. Trademarked name for scenario-driven information query templates that map to specific and familiar semantic models for information access and retrieval. For example, a “Headlines” template in the preferred embodiment has parameters that are consistent with the delivery of “Headlines” (where freshness and the likelihood of a high interest level are the primary axes for retrieval). An “Upcoming Events” template has parameters that are consistent with the delivery of “Upcoming Events.” And so on. Essentially, Context Templates can be analogized to personal, digital semantic information retrieval “channels” that deliver information to the user by employing a well-known semantic template.

[0309] Deep Information™. Trademarked name for a feature of the present invention that enables the Information Agent to display intrinsic, contextual information relating to an information object. The contextual information that includes information that is mined from the Semantic Network of the Agency from whence the object came.

[0310] Discoverability. The ability of the information medium of the present invention to intelligently and proactively make information known or visible to the user without the user having to explicitly look for the information.

[0311] Domain Agent Wizard™. Trademarked name for a system component and its user interface for allowing the Agency administrator to create and manage Domain Agents.

[0312] DOTNET (NET). Microsoft® .NET is a set of Microsoft software technologies for connecting information, people, systems, and devices. It enables software integration through the use of XML Web Services: small, discrete, building-block applications that connect to each other, as well as to other, larger applications, via the Internet. .NET-connected software facilitates the creation and integration of XML Web Services. See http://www.microsoft.com/net/defined/default.asp.

[0313] Dynamic Linking™. Trademarked name for the ability of the Information Nervous System of the present invention to allow users to link information dynamically, semantically, and at the speed of thought, even if those information items do not contain links themselves. By virtue of employing smart objects that have intrinsic behavior and using recursive intelligence embedded in the Information Agency’s XML Web Service, each node in the Semantic Network is much smarter than a regular link or node on Today’s Web or the conceptual Semantic Web. In other words, each node in the Smart Virtual Network or Web of the present invention can link to other nodes, independent of authoring. Each node has behavior that can dynamically link to Agencies and Smart Agents via drag and drop and smart copy and paste, create links to Agencies in the Semantic Environment, respond to lens requests from Smart Agents to create new links, include intrinsic alerts that will dynamically create links to context and time-sensitive information on its Agency, include presentation hints for breaking news (wherein the node can automatically link to breaking news Agents in the namespace), form the basis for deep info that can allow the user to find new links, etc. A user of the present invention is therefore not at the mercy of the author of the metadata. Once the user reaches a node in the network, the user has many semantic means of navigating dynamically and automatically—using context, time, relatedness to Smart Agencies and Agents, etc.

[0314] Email XML Object. An information object with the “Email” information object type. The XML object has the “Email” SRML schema (which uses XML).


[0316] Favorite Agents Manager™. Trademarked name for a system component and user interface element that allows the Agency administrator to manage server-side Favorite Agents.

[0317] Flash. Macromedia Flash user interface platform that enables developers and content authors to embed sophisticated graphics and animations in their content. See http://www.macromedia.com/flash.

[0318] Flash MX. Macromedia Flash MX is a text, graphics, and animation design and development environment for creating a broad range of high-impact content and rich applications for the Internet. See http://www.macromedia.com/ software/flash/productinfo/product_overview/.


[0320] HTTP. Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol that can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred. See http://www.w3.org/Protocols/ and http://www.w3.org/Protocols/Specs.html.

[0321] Inference Engine™. Trademarked name for the methodology of the present invention that observes patterns and data to arrive at relevant and logically sound conclusions.
by reasoning. Preferably utilizes inference rules (a predetermined set of heuristics) to add semantic links to the Semantic Network of the present invention.

[0322] Information. A quantitative or qualitative measure of the relevance and intelligence of content or data and which conveys knowledge.

[0323] Information Agent™. Trademarked name for the semantic client or browser of the present invention that provides context and time-sensitive delivery and presentation of actionable information (or knowledge) from multiple sources, information types, and templates, and which allows dynamic linking of information across various repositories.

[0324] Information Nervous System™. Trademarked name for the dynamic, self-authoring, context and time-sensitive information system of the present invention that enables users to intelligently and dynamically link information at the speed of thought, and with context and time-sensitivity, in order to maximize the acquisition and use of knowledge for the task at hand.

[0325] Information Object™ (or Item or Packet). Trademarked name for a unit of information of a particular type and which conveys knowledge in a given context.

[0326] Information Object Pivot™. Trademarked name for an information object that users employ as a navigational pivot to find other relevant information in the same context.

[0327] Information Object Type. See Object Type.

[0328] Intelligent Agent. Software Agents that act on behalf of the user to find and filter information, negotiate for services, easily automate complex tasks, or collaborate with other software Agents to solve complex problems. By definition, Intelligent Agents must be autonomous or, in other words, freely able to execute without user intervention. Additionally, Intelligent Agents must be able to communicate with other software or human Agents and must have the ability to perceive and monitor the environment in which they reside. See http://www.findarticles.com/cf_dls/m/nfWE/7_4/64694222/p1/article.html.

[0329] Internet Calendaring and Scheduling (iCalendar). Protocol that enables the deployment of interoperable calendaring and scheduling services for the Internet. The protocol provides the definition of a common format for exchanging calendaring and scheduling information across the Internet.

[0330] Internet Message Access Protocol (IMAP). Communications mechanism for mail clients to interact with mail servers, and manipulate mailboxes thereon. Perhaps the most popular mail access protocol currently is the Post Office Protocol (POP), which also addresses remote mail access needs. IMAP offers a superset of POP features, which allow much more complex interactions and provides for much more efficient access than the POP model. See http://www-smi.stanford.edu/projects/imap/ml3/imap.html.

[0331] Intrinsic Semantic Link™. Trademarked name for semantic links that are intrinsic to the schema of a particular information object. For instance, an email information object has intrinsic links like “from,” “to,” “cc,” “bcc,” and “attachments” that are native to the object itself and are defined in the schema for the email information object type.

[0332] Island. An information repository that is isolated from other repositories which may contain relevant, semantically related, context and time-sensitive information but which are disconnected from other contexts in which such information might be relevant.

[0333] J2EE. The Java™ 2 Platform, Enterprise Edition (J2EE) used for developing multi-tier enterprise applications. J2EE bases enterprise applications on standardized, modular components by providing a set of services to those components and by handling many details of application behavior automatically. See http://java.sun.com/j2ee/overview.html.

[0334] Knowledge. Information presented in a context and time-sensitive manner that enables the information consumer to learn from the information and apply the information in order to make smarter and more timely decisions for relevant tasks.


[0336] Knowledge Base Server™ (KBS). Trademarked name for a server that hosts knowledge for the Knowledge Integration Server (KIS).

[0337] Knowledge Domain Manager™ (KDM). Trademarked name for a component of the Knowledge Integration Server that is responsible for adding and maintaining domain-specific intelligence on the Semantic Network.

[0338] Knowledge Integration Server™ (KIS). Trademarked name for a server that semantically integrates data from multiple diverse sources into a Semantic Network, which can also host server-side Agents that provide access to the network and which hosts XML Web Services that provide context and time-sensitive access to knowledge on the server.


[0340] Liberty Alliance. The vision of the Liberty Alliance is to enable a networked world in which individuals and businesses can more easily conduct transactions while protecting the privacy and security of vital identity information. To accomplish its vision, the Liberty Alliance seeks to establish an open standard for federated network identity through open technical specifications. See http://www.projectliberty.org/index.html.

[0341] Lightweight Directory Access Protocol (LDAP). Technology for accessing common directory information. LDAP has been embraced and implemented in most network-oriented middleware. As an open, vendor-neutral standard, LDAP provides an extendable architecture for centralized storage and management of information that needs to be available for today’s distributed systems and services. LDAP is currently supported in most network operating systems, groupware and even shrink-wrapped network applications. See http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg/244986.html?Open.

[0342] Link Template™. See Context Template.

[0343] Local Context. Local Context refers to client-side information objects and Agents accessible to the users. This includes Agents in the Semantic Environment, local files, folders, email items in users’ email inboxes, users’ favorite and recent Web pages, the current Web page(s), currently opened documents, and other information objects that represent users’ current task, location, time, or condition.

[0344] Meaning. The attributes of behavior of information that allows the consumer of the information to locate and navigate to it based on its relevant information content (as opposed to its text or data) and to act on it in a context and time-sensitive manner, in order to maximize the utility of the information.

[0346] Natural Language Parser. Parsing and interpreting software component that understands natural language queries and can translate them to structured semantic information queries.

[0347] Nervana™. Trademarked name for a proprietary, end-to-end implementation of the Information Nervous System information medium/platform. The name also defines a proprietary namespace for resource type and predicate name qualifiers.

[0348] .NET Passport. Microsoft .NET Passport is a suite of Web-based services directed toward the Internet and online purchasing. .NET Passport provides users with single sign-in (SSI) and fast purchasing capability at a growing number of participating sites, reducing the amount of information users must remember or retype. .NET Passport provides a high-quality online experience for a large user base and uses powerful encryption technologies—such as Secure Sockets Layer (SSL) and the Triple Data Encryption Standard (3DES) algorithm—for data protection. Privacy is a key priority as well, and all participating sites sign a contract in which they agree to post and follow a privacy policy that adheres to industry-accepted guidelines.

[0349] Network Effects. This exists when the number of other users affects the value of a product or service to a particular user. Telephone service provides a clear example. The value of telephone service to users is a function of the number of other subscribers. Few would be interested in telephones that were not connected to anyone, and most would assess higher value to a phone service linked to a national network rather than just a local network. Similarly, many computer users prize a computer system that allows them to exchange information readily with other users.

[0350] Network Effects are thus demand-side externalities that generate a positive feedback effect in which successful products become more successful. In this way, Network Effects are analogous to supply-side economies of scale and scope. As a firm increases output, economies of scale lead to lower average costs, permitting the firm to lower prices and gain additional business from rivals. Continued expansion results in even lower average costs, justifying even lower prices. Similarly, the positive feedback from Network Effects builds upon previous successes. In the computer industry, for example, users pay more for a more popular computer system, all else equal, or opt for a system with a larger installed base if the prices and other features of two competing systems are equivalent. See http://www.et.com/publications/1996/fall1.html.

[0351] Network News Transfer Protocol (NNTP). Protocol for the distribution, inquiry, retrieval, and posting of news articles using a reliable stream-based transmission of news among the ARPA-Internet community. NNTP is designed so that news articles are stored in a central database allowing subscribers to select only those items they wish to read. Indexing, cross-referencing, and expiration of aged messages are also provided.

[0352] Notifications. Notifications are alerts that are sent by the Information Agent or an Agency to indicate to a user that there is new information on an Agent (either a client-side Agent or a server-side Agent). Users can request notifications from Agents in their Semantic Environment. Users can indicate that they have received the notification. The notification source (the client or server) stores information for the user and the Agent indicating the last time the user acknowledged a notification for the Agent. The notification source polls the Agent to check if there is new information since the last acknowledge time. If there is, the notification source alerts the user. Alerts can be sent via email, pager, voice, or a custom alert mechanism such as Microsoft’s .NET Alerts service. Users have the option of indicating their preferred notification mechanism for the entire notification source (client or server)—which applies to all Agents on the notification source—on a per-Agent basis (which overrides the indicated preference on the notification source).


[0354] Object Type. Identification data associated with information that allows the consumer to understand the nature of the information, to interpret its contents, to predict how the information can be acted upon, and to link it to other relevant information items based on how the object types typically relate in the real world. Examples include documents, events, email messages, people, etc.

[0355] Ontology. Hierarchical structuring of knowledge according to essential qualities. Ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where “Ontology” is a systematic account of Existence. For artificial intelligence systems, what “exists” is that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary, with which a knowledge-based program represents knowledge. Thus, in the context of artificial intelligence, the ontology of a program is described by a set of representational terms. In such ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms. Formally, ontology is the statement of a logical theory.

[0356] The subject of ontology is the study of the categories of things that exist or may exist in some domain. The product of such a study, called ontology, is a catalog of the types of things that are assumed to exist in a domain of interest D from the perspective of a person who uses a language L for the purpose of talking about D. The types in the ontology represent the predicates, word senses, or concept and relation types of the language L, when used to discuss topics in the domain D. See, generally, http://www-ksl.stanford.edu/ksl/what-is-an-ontology.html and http://users.bestweb.net/~sowa/ontology/).

[0357] Predicates. A Predicate is an attribute or link whose result represents the truth or falsehood of some condition. For example, the predicate “authored by” links a person with an information object and indicates whether a person authored the object.

[0358] Presenter™. System component in the Information Agent (semantic browser) of the present invention that handles the aggregation and presentation of results from the semantic query processor (that preferably interprets SQML). The Presenter handles layout management, aggregation, navigation, Skin management, the presentation of Context Palettes, interactivity, animations, etc.

[0359] RDF. Resource Description Framework (RDF) is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF emphasizes facilities to enable automated processing of Web resources. RDF defines
a simple model for describing relationships among resources in terms of named properties and values. RDF properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. RDF properties also represent relationships between resources. As such, the RDF data model can therefore resemble an entity-relationship diagram.

RDF can be used in a variety of application areas including, for example: in resource discovery to provide better search engine capabilities, in cataloging for describing the content and content relationships available at a particular Web site, page, or digital library; by intelligent software Agents to facilitate knowledge sharing and exchange, in content rating, in describing collections of pages that represent a single logical "document", for describing intellectual property rights of Web pages, and for expressing the privacy preferences of a user as well as the privacy policies of a Web site. RDF with digital signatures is preferably a component of building the "Web of Trust" for electronic commerce, collaboration, and other applications. See, generally, http://www.w3.org/TR/PR-rdf-syntax/ and http://www.w3.org/TR/rdf-schema/.

RDFS. Acronym for RDF Schema. Resource description communities require the ability to say certain things about certain kinds of resources. For describing bibliographic resources, for example, descriptive attributes including "author", "title", and "subject" are common. For digital certification, attributes such as "checksum" and "authentication" are often required. The declaration of these properties (attributes) and their corresponding semantics are defined in the context of RDF as an RDF schema. A schema defines not only the properties of the resource (e.g., title, author, subject, size, color, etc.) but may also define the kinds of resources being described (books, Web pages, people, companies, etc.). See http://www.w3.org/TR/rdf-schema/.

Results Pane™. Trademarked name for the graphical display area within the Information Agent (semantic browser) that displays results of an SQML query. For example, an optional player control/navigation/filter toolbar, a "Server-Side Agents Dialog" can allow users to browse and open server-side Agents, and display sample results with the "Documents" information object type from the server-side Agent.

Semantics. Connotative meaning.

Semantic Environment™. This refers to all the data stored on users' local machines, in addition to user-specific data on an Agency server (e.g., subscribed server-side Agencies, server-side Favorite Agents, etc.). Client-side state includes favorite and recent Agencies and authentication and authorization information (e.g., user names and passwords for various Agencies), in addition to the SQML files and buffers for each client-side (user-created) Agent. The Information Agent is preferably configured to store Agents for a set amount of time before automatically deleting them, except those that have been added to the "favorites" list. For example, users may configure the Information Agent to store Agents for two weeks. In this case, Agents older than two weeks are automatically purged from the system and the Semantic Environment is adjusted accordingly. The Semantic Environment is employed for Context Palettes (Context Palettes use the Agencies in the "recent" and "favorites" list in order to predict what default Agencies users want to view context from).
etc. In addition, because SQML does not define the rules for presentation, the results of the semantic query can be presented in multiple ways, using a "skin" that takes the results (in SRML) to generate presentation based on the user's preferences, interests, condition, or context. Furthermore, SQML can contain abstract links and predicates such as those that refer to or employ Context Templates. The resource (e.g., the Agency's XML Web Service) then resolves the SQML to an appropriate query format (e.g., SQL or the equivalent in the case of an Agency's XML Web Service) and then invokes the "actual" query in order to generate the results (which will then account for the user's context or Context Template). Also, an SQML buffer or file can refer to multiple resources (and Agencies), thereby empowering the client to view results in an aggregated fashion (e.g., based on context or time-sensitivity), rather than based on the source of the data—this is a powerful feature of the invention that enables user-controlled browsing and information aggregation (see the options on both below). Lastly, every client-side Agent has an SQML definition and file, just as every Web page has an HTML file.

Semantic Query Processor (SQP). Trademarked name for the server-side semantic query processor (XML Web Service in the preferred embodiment) that takes SQML and converts it to SQL (in the preferred embodiment) and then returns the results as XML. On the Knowledge Integration Server (KIS), the SQP is the main entry point to the Semantic Network of the present invention responsible for responding to semantic queries from clients of the KIS. On the server, this is the software component that processes semantic queries presented as SQML from the client. On the client, the client-side SQP takes aggregate SQML and compiles or maps it to individual SQML queries that can be sent to a server (or Agency) XML Web Service.

Semantic Results Markup Language (SRML). A proprietary XML-based data schema and format used by this invention to define, store, interpret and present semantic results. On the client, SRML is returned from the SQP via semantic resource handlers that interpret, format, and issue query requests to semantic data sources. Semantic data sources will include an Agency's XML Web Service, local files, local folders, custom data sources from local or remote applications (e.g., a Microsoft Outlook email application inbox), etc. The XML Web Service will return SRML to a client, in response to the client's semantic query. This way, the XML Web Service will not "care" how the results are being presented at the client. This is in contrast with Today's Web and the Semantic Web where servers return already-formatted HTML for a client to present and where clients merely present presentation data (as opposed to semantic data) and cannot customize the presentation of the data. In this invention, two clients can render the same SRML in completely different ways, based on the current "skin" that has been selected or applied by the user of either client. The "skin" then converts the SRML to a presentation-ready format such as XHTML, DHTML+TIME, SVG, Flash MX, etc.

SRML is a meta-schema, meaning that it is a container format that can include data for different information object types (e.g., documents, email, people, events, etc.). An SRML file or buffer can contain intertwined results for each of these object types. Well-formed SRML will contain well-formed XML document sections that are consistent with the schema of the information object types that are contained in the semantic result the SRML represents. See Sample A of the Appendix hereto.


Facilities to put machine-understandable data on Today's Web are becoming a high priority for many communities. The Web can reach its full potential only if it becomes a place where data can be shared and processed by automated tools as well as by people. For the Web to scale, tomorrow's programs must be able to share and process data even when these programs have been designed totally independently. The Semantic Web is a conceptual vision: the idea of having data on the Web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications. See also http://www.w3.org/2001/sw/.

Session Announcement Protocol (SAP). In order to assist the advertisement of multicast multimedia conferences and other multicast sessions, and to communicate the relevant session setup information to prospective participants, a distributed session directory may be used. An instance of such a session directory periodically multicasts packets containing a description of the session, and these advertisements are received by other session directories such that potential remote participants can use the session description to start the tools required to participate in the session.

In simplest form, this involved periodically multicasting a session announcement packet describing a particular session. To receive SAP, a receiver simply listens on a well-known multicast address and port. Sessions are described using the Session Description Protocol (ftp://ftp.isi.edu/in-notes/rfc2327.txt). If a receiver receives a session announcement packet it simply decodes the SDP message, and then can display the session information for the user. The interval between repeats of the same session description message depends on the number of sessions being announced (each sender at a particular scope can hear the other senders in the same scope) such that the bandwidth being used for session announcements of a particular scope is kept approximately constant. If a receiver has been listening for a set time, and fails to hear a session announcement, then the receiver can conclude that the session has been deleted and no longer exists. The set period is based on receivers' estimate of how often the sender should be sending.


Simple Mail Transfer Protocol (SMTP). Protocol designed to transfer mail reliably and efficiently. SMTP is independent of the particular transmission subsystem and requires only a reliable ordered data stream channel. An important feature of SMTP is its capability to relay mail across transport environments. See http://www.ietf.org/rfc/rfc0821.txt.

Skins. Presentation templates that are used to customize the user experience on a per-Agent basis or which customizes the presentation of the entire layout (independent of the Agent), or object (based on the information object type), context (based on the Context Template), Blender (for Agents that are Blenders), for the semantic domain name/path or ontology, and other considerations. Each Agent will include a Skin which in turn will have an XML: metadata representation of parameters to customize the layout of the
XML-based results that represent information objects (the layout Skin), for example, whether or not those results are animated, the manner in which each result is displayed, including a representation of the object type (the object Skin), styles, colors, graphics, filters, transforms, effects, animations (and so on) that indicate the ontology of the current results (the ontology Skin), styles that indicate the Context Template of the current results (the context Skin) and styles that indicate how to view and navigate results from Blenders (i.e., the Blender Skin).

[0382] Smart Lens™ Trademarked name for a proprietary feature of this invention that allows users to select a Smart Agent or an object as a context with which to view another object or Agent. The lens then displays metadata, links, and result previews that give users an indication of what they should expect if the context is invoked. Essentially, the Smart Lens displays the results of a “potential query.” The Smart Lens allows users to quickly preview context results without actually invoking queries (thereby increasing their productivity). In addition, the Smart Lens can display views that are consistent with the context, using pivots, templates and preview windows, thereby allowing users to analyze the context in different ways before invoking a query.

[0383] Smart Virtual Web™ Trademarked name for the property of the present invention to integrate semantics, context-sensitivity, time-sensitivity, and dynamism in order to empower users to browse a dynamic, virtual “on-the-fly,” user-controlled “Web” that they control and can customize. This is in contrast with Today’s Web and the conceptual Semantic Web, both of which employ a manually authored network wherein users are at the mercy of the authors of the information on the network.

[0384] Structured Query Language (SQL). Pronounced “ess-que-el.” SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system. However, the standard SQL commands such as “Select”, “Insert”, “Update”, “Delete”, “Create”, and “Drop” can be used to accomplish almost everything that one needs to do with a database.

[0385] SQL works with relational databases. A relational database stores data in tables (relations). A database is a collection of tables. A table consists of a list of records, each record in a table preferably includes the same structure, and each has a fixed number of “fields” of a given type.


[0387] Scalable Vector Graphics (SVG). Language for describing two-dimensional graphics in XML. SVG allows for three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images and text. Graphical objects can be grouped, styled, transformed and composited into previously rendered objects. Text can be in any XML namespace suitable to the application, which enhances searchability and accessibility of the SVG graphics. The feature set includes nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility. SVG drawings can be dynamic and interactive. The Document Object Model (DOM) for SVG, which includes the full XML DOM, allows for straightforward and efficient vector graphics animation via scripting. A rich set of event handlers such as onmouseover and onclick can be assigned to any SVG graphical object. Because of its compatibility and leveraging of other Web standards, features like scripting can be done on SVG elements and other XML elements from different namespaces simultaneously within the same Web page. See http://www.w3.org/Graphics/SVG/Overview.html.

[0388] Taxonomy. An organizational structure wherein divisions are ordered into groups or categories.

[0389] Time-Sensitivity. Property of an information medium to deliver and present information based on when the information would be most relevant in time. For instance, freshness is an attribute that denotes time-sensitivity. In addition, the delivery and presentation of upcoming events (which, by definition, are time-sensitive) and the manner in which the time-criticality of the events are displayed are properties of a time-sensitive medium.

[0390] Today’s Web. This refers to the World Wide Web as we know it today. Today’s Web is a universe of hypertext servers (HTTP servers), which are the servers that allow text, graphics, sound files, etc. to be linked together. Hypertext is simply a non-linear way of presenting information. Rather than reading or learning about things in the order that an author, editor, or publisher sets out for us, readers of hypertext may follow their own path, create their own order or meaning out the material. This is accomplished by creating “links” between information. These links are provided so that user may “jump” to further information about a specific topic being discussed (which may have more links, leading each reader off into a different direction). The Hypertext medium can incorporate pictures, sound, and video present a multimedia approach to presenting information, also referred to as hypermedia. See, generally, http://www.w3.org/History.html and http://www.umassdl.edu/Public/People/KAmaril/The-sis/hypertext.html.

[0391] Multicast Time to Live (TTL). Multicast routing protocol uses the field of datagrams to decide how “far” from a sending host a given multicast packet should be forwarded. The default TTL for multicast datagrams is 1, which will result in multicast packets going only to other hosts on the local network. A setsockopt(2) call may be used to change the TTL. As the value for TTL increases, routers will expand the number of hops they will forward a multicast packet. To provide meaningful scope control, multicast routers typically enforce the following “thresholds” on forwarding based on the TTL field:

<table>
<thead>
<tr>
<th>TTL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Restricted to the same host</td>
</tr>
<tr>
<td>1</td>
<td>Restricted to the same subnet</td>
</tr>
<tr>
<td>2</td>
<td>Restricted to the same site</td>
</tr>
<tr>
<td>3</td>
<td>Restricted to the same region</td>
</tr>
<tr>
<td>4</td>
<td>Restricted to the same continent</td>
</tr>
<tr>
<td>255</td>
<td>Unrestricted</td>
</tr>
</tbody>
</table>


[0398] User State. This refers to all state that is either created by a user or which is needed to cache a user’s preferences, favorites, or other personal information on a client or server. Client-side User State includes authentication credential information, users’ Agent lists (and all the metadata including the SQML queries for the Agents), home Agent, configuration options, preferences such as Skins, etc. Essen-
ially, client-side User State is a persisted form of users' Semantic Environment. Server-side User State includes information such as users' Favorite Agents, subscribed Agents, Default Agent, semantic links to information objects on the server (e.g., "favorites" links) etc. Server-side User State is optional for servers but support for it is preferred. Servers preferably support user logon and a "people" object type (even without server-side Agents) because these are needed for features such as favorites, recommendations, and for Context Templates such as "Newsmakers," "Experts," "Recommendations," "Favorites," and "Classics."

Virtual Information Object Type™. Trademarked name for object types that do not map to distinct object types, yet are semantically of interest to users.

Virtual Parameter™. Trademarked name for variables, parameters, arguments, or names that are dynamically interpreted at runtime by the semantic query processor. This allows the Agency administrator to store Agents that refer to virtual names and then have those names be converted to actual relevant terms when the query is invoked.

Web of Trust. Term coined by members of the Semantic Web research community that refers to a chain of authorization that users of the Semantic Web can use to validate assertions and statements. Based on work in mathematics and cryptography, digital signatures provide proof that a certain person wrote (or agrees with) a document or statement. Users can preferably digitally sign all of their RDF statements. That way, users can be sure that they wrote them (or at least vouch for their authenticity). Users simply tell the program whose signatures to trust. Each can set their own levels of trust (or paranoia), and the computer can decide how much of what it reads to believe.

By way of example, with a Web of Trust, a user can tell a computer that he or she trusts his or her best friend, Robert. Robert happens to be a rather popular guy on the Net, and trusts quite a number of people. All the people he trusts in turn trust another set of people. Each of these measures of trust is to a certain degree (Robert can trust Wendy a whole lot, but Sally only a little). In addition to trust, levels of distrust can be factored in. If a user’s computer discovers a document which no one explicitly trusts, but no one has said it has totally false either, it will probably trust that information a little more than one which many people have said is false. The computer takes all these factors into account when deciding the trustworthiness of a piece of information. Preferably, the computer combines all this information into a simple display (thumbs-up/thumbs-down) or a more complex explanation (a description of all the various trust factors involved). See [link to blogspot.com/rdf/SwartzFandelier].

Web Services-Interoperability (WS-I). An open industry organization chartered to promote Web services interoperability across platforms, operating systems, and programming languages. The organization works across the industry and standards organizations to respond to user needs by providing guidance, best practices, and resources for developing Web services solutions. See [link to www.ws-i.org].

Web Services Security (WS-Security). Enhancements to SOAP messaging providing quality of protection through message integrity, message confidentiality, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies. WS-Security also provides a general-purpose mechanism for associating security tokens with messages. No specific type of security token is required by WS-Security. It is designed to be extensible (e.g. support multiple security token formats). For example, a client might provide proof of identity and proof that they have a particular business certification. Additionally, WS-Security describes how to encode binary security tokens. Specifically, the specification describes how to encode X.509 certificates and Kerberos tickets as well as how to include opaque encrypted keys. It also includes extensibility mechanisms that can be used to further describe the characteristics of the credentials that are included with a message. See [link to msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-security.asp].

Extensible Markup Language (XML). Universal format for structured documents and data on the Web. Structured data includes things like spreadsheets, address books, configuration parameters, financial transactions, and technical drawings. XML is a set of rules (you may also think of them as guidelines or conventions) for designing text formats that let you structure your data. XML is not a programming language, and one does not have to be a programmer to use it or learn it. XML makes it easy for a computer to generate data, read data, and ensure that the data structure is unambiguous. XML avoids common pitfalls in language design: it is extensible, platform-independent, and it supports internationalization and localization. XML is fully Unicode-compliant. See [link to www.w3.org/XML/1999/XML-in-10-points].

XML Web Service (also known as “Web Service”). Service providing a standard means of communication among different software applications involved in presenting dynamic context-driven information to the user. More specific definitions include:

1. A software application identified by a URI whose interfaces and binding are capable of being defined, described and discovered by XML artifacts. Supports direct interactions with other software applications using XML-based messages via Internet-based protocols.

2. Application delivered as a service that can be integrated with other Web Services using Internet standards. It is an URL-addressable resource that programmatically returns information to clients that want to use it. The major communication protocol used is the Simple Object Access Protocol (SOAP), which in most cases is XML over HTTP.

3. Programmable application logic accessible using standard Internet protocols. Web

Services combine aspects of component-based development and the Web. Like components, Web Services represent black-box functionality that can be reused without worrying about how the service is implemented. Unlike current component technologies, Web Services are not accessed via object-model-specific protocols, such as DCOM, RMI, or HOP. Instead, Web Services are accessed via ubiquitous Web protocols (ex: HTTP) and data formats (ex: XML). See [link to www.xmlwebservices.com], [link to perfectxml.com/WebSv1.asp] and [link to www.w3.org/2002/ws/archive/2006/wd-wsa-reqs-20020605.html].

XQuery. Query language that uses the structure of XML to intelligently express queries across all these kinds of data, whether physically stored in XML or viewed as XML via middleware. See [link to www.w3.org/TR/xquery/ and link to www-106.ibm.com/developerworks/xml/library/x-xquery.html].

XPath. The result of an effort to provide a common syntax and semantics for functionality shared between XSL
Transformations (http://www.w3.org/TR/XSLT) and XPointer (http://www.w3.org/TR/xpath/XPTR). The primary purpose of XPath is to address parts of an XML [XML] document. In support of this primary purpose, it also provides basic facilities for manipulation of strings, numbers and Booleans. XPath uses a compact, non-XML syntax to facilitate use of XPath within URLs and XML attributes. XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax. XPath gets its name from its use of a path notation as in URLs for navigating through the hierarchical structure of an XML document.

[0413] In addition to its use for addressing, XPath is also designed so that it has a natural subset that can be used for matching (testing whether or not a node matches a pattern); this use of XPath is described in XSLT. XPath models an XML document as a tree of nodes. There are different types of nodes, including element nodes, attribute nodes and text nodes. XPath defines a way to compute a string-value for each type of node. Some types of nodes also have names. XPath fully supports XML Namespaces (http://www.w3.org/TR/xpath/XMNLNAMES). Thus, the name of a node is modeled as a pair consisting of a local part and a possibly null namespace URI; this is called an (http://www.w3.org/TR/xpath/dt-expanded-name). See http://www.w3.org/TR/xpath/XPTR.

[0414] XSLT. A style sheet language for XML that includes an XML vocabulary for specifying formatting. See http://www.w3.org/TR/xslt11/.

[0415] XSLT. Used by XSL to describe how a document is transformed into another XML document that uses the formatting vocabulary. See http://www.w3.org/TR/xslt11/.

B. Overview

[0416] 1. Invention Context

[0417] There is a misconception that the Holy Grail for information access is the provision of natural language searching capability. Prior technologies for information access have focused principally on improving the interface for searching for or accessing information to optimize information retrieval. The presumption has largely been that providing a natural language interface to information will perfectly solve users' information access problems and end the frustration users have with finding information.

[0418] In truth, however, many axes of analysis are involved in how people acquire knowledge in the real world. One example is context. There are many things people know only because of where they were at a certain place and time. If they were not at that place at that time, they would not know what is in fact known or, indeed, might not care to know. Having the ability to search for what is presently known with natural language does not assist in uncovering the knowledge related to that particular time and place. There are simply no natural parameters that form the correct query to retrieve the desired information.

[0419] The conundrum is that a person cannot ask for what he or she might not even know would have value until after the fact. Stated differently, one cannot query for what they do not now they do not know, or for what they do not know that they might want to know. Context-sensitivity, time-sensitivity, discovery, dynamic linking, user-controlled browsing, users’ “Semantic Environment,” flexible presentation, Context Skins, context attributes, Context Palettes (which bring up relevant, context and time-sensitive information based on Context Templates) and other aspects of this invention recognize and correct this fundamental deficiency with existing information systems.

[0420] For example, people may have many CDs in their library (thereby adding to the “knowledge” of music) because they attended certain parties and spoke with certain people. Those people at those parties mentioned the CDs to the person, thereby increasing the person’s knowledge of music. As another example, a person may purchase a book (if read, increasing the person’s knowledge on the particular topic of the book) based on a recommendation from a hitherto unknown stranger the person happened to sit beside on an airplane flight. In the real world, people acquire knowledge based not just on what they read and search for, but also based on the friends they keep, the people with whom they interact and the people whose judgment they trust. The “knowledge environment” is arguably as critical if not more critical for knowledge dissemination and acquisition as the model for retrieval (whether digital or analog).

[0421] The present invention mirrors virtually every real-world knowledge-acquisition scenario in the digital world. The resulting Information Nervous System™ is the medium doing most of the work but the scenarios map very cleanly to the analog (real) world. The inability of efforts such as natural-language search techniques of Today’s Web as well as the Semantic Web to recognize the many ways in which knowledge is disseminated and acquired render them ultimately ineffective. The present invention accounts for the variety of ways in which humans have always acquired knowledge—indeed, independent of the actual technology used for information delivery.

[0422] By way of example, there has always been context and there has always been time. Likewise there has always been the notion of discovery and the need to link information dynamically and with user control. There have always been certain Context Templates, albeit in different mediums that presented herein, including “classics,” “history,” “timelines,” “upcoming events,” “headlines.” These templates existed before the creation of the Internet, Today’s Web, Email, e-Learning, etc. Nevertheless, prior to the present invention, there was no ability in the electronic medium to focus on the mode, protocol and presentation of knowledge delivery which maps to real-world scenarios (for example, via Context Templates, context-sensitivity, time-sensitivity, dynamic linking, flexible presentation, Context Skins, context attributes, etc.) as opposed to actual information types, semantic links, metadata, etc. There will always be new information types. But the dissemination and acquisition axes of knowledge (e.g., Context Templates) have always and will always remain the same. The present invention captures this reality.

[0423] In addition, the present invention provides the ability to disseminate knowledge via serendipity. Serendipity plays a large part in knowledge acquisition in the real world and it is a first-class mode of knowledge delivery. The present invention enables a user to acquire information serendipitously (albeit intelligently) by its support for context, time, Context Templates, etc.

[0424] Information models or mediums that employ a strict, static structure like a “Web” break down because they assume the presence of an authored “network” or “Web” and fail to account for the various axes of knowledge formation. Such information models are not user-focused, do not incorporate context, time, dynamism and templates, and do not
map to real-world knowledge acquisition and dissemination scenarios. The present invention minimizes information loss and maximizes information retained, even without the presence of a “Web” per se, and even if no natural language is employed to find information. This is possible because, unlike existing mediums for information access, a preferred embodiment of the present invention focuses on the knowledge dissemination models that incorporate context, time, dynamism, and templates (for the benefit of both the end-user and the content producer) and not on the specifics of the access interface, or the linking (semantic or non-semantic) of information resources based on static data models or human-based authoring. In many scenarios, a “Web” (semantic or non-semantic) is necessary as a means of navigation, but is far from being sufficient as a means of knowledge dissemination and acquisition. The Information Nervous System of the present invention incorporates “knowledge axes” described in the invention (including but not limited to link-based navigation) and intelligently and seamlessly integrates them to facilitate the dissemination and acquisition of knowledge and to benefit all parties involved in the transfer of knowledge.

[0425] 2. Value Propositions

[0426] Today, knowledge must be “manually hard-coded” into the digital fabric of an information structure, whether it be for an enterprise, a consumer or the general inquiring population. If it is not authored and distributed properly, no one knows of its existence. Knows how it relates to other sources of intelligence, or knows how to act on it in real-time and in the proper fashion. This is largely because Today’s Web was not designed to be a platform for knowledge. It was designed to be a platform for presentation and is intentionally dumb, static, and reactive. Today, knowledge-workers—those who seek to use information by adding context—and meaning are at the mercy of knowledge-authors.

[0427] A significant aspect of knowledge interaction is to have knowledge-workers be able to navigate their way through a knowledge space in a very intuitive manner, and at the speed at which they wish to make decisions and act on the knowledge. In other words, knowledge-workers do not have to “think” about an e-Learning island as being separate from documents in their organizations, e-mail that contains customer feedback, media files, upcoming video-conferences, a meeting they had recently, information stored in newsgroups, or related books. The preferred situation is to relegate the information “type” and “source” to create a seamless knowledge experience that cuts across all those islands in a semantic way.

[0428] In creating a knowledge experience, it is also preferred to be able to integrate knowledge assets across content-provider, partner, supplier, customer and people boundaries. In the enterprise scenario, for example, no single organization has all the knowledge it needs to remain competitive. Knowledge is stored in industry reports, research documents from consulting firms and investment banks, media companies like Reuters™ and Bloomberg™, etc. All this constitutes “knowledge.” It is not enough to deploy an e-Learning repository to train users on a one-time or periodic basis. Users should have always-on access to knowledge from a variety of sources, in-place, and in an intelligent context that is relevant to their current task.

[0429] All this requires a layer of intelligence and proactivity that is not available today. Today, for example, enterprises use information portals, such as intranets and the Internet, as a way of disseminating information to their employees. However, this is far from being enough, as it provides only presentation-level integration. This is akin to subscribing to newsletters to keep updated with information, as opposed to having an Agent that manages your information for you, helps you discover new information on-the-fly, helps you capture and share information with colleagues, etc.

[0430] To accomplish the desired level of knowledge interaction requires Agents working in the background, reasoning, learning, inferring, matching users together based on their profiles, capturing new knowledge and automatically deducing new knowledge, and federating knowledge from external sources so that they become a seamless part of the knowledge experience. This in turn requires the semantic integration of knowledge assets so that they all make sense in a holistic fashion, rather than merely providing the basis for presentation-level integration and document searching. The implementation framework and resulting medium must provide real-time, agile discovery and recommendation services so that context and time-sensitive information is “honored” and such that knowledge-workers can be more productive and get more done faster and with less. And lastly, the system must work with existing information sources in a plug-n-play manner, must seamlessly and automatically classify and integrate known knowledge assets, and must embed the knowledge tools in the knowledge themselves, thereby adding another “dimension” into knowledge assets.

[0431] The present invention is designed to be an intelligent, proactive, real-time knowledge platform that co-exists with Today’s Web (or any other layer of presentation). Incorporation and use of the present invention will allow knowledge-workers to be in control of their knowledge experiences because authoring (via “connections”) will be done intelligently, dynamically, automatically, and at the speed of thought.

[0432] 3. Today’s “Information” Web vs. The Information Nervous System of the Present Invention

[0433] With Today’s Web environment, the semantics of information presented are lost upon conversation of the structured data to HTML at the server, meaning that the “knowledge” is stripped from the objects before the user has an opportunity to interact with them. In addition, Today’s Web is authored and “hard-coded” on the server based on how the author “believes” the information will be navigated and consumed. Users consume only information as it is presented to them.

[0434] The present invention adds a layer of intelligence and layers of customization that Today’s HTML-based Web environment cannot support. The present invention provides an XML-based dynamic Web of smart knowledge objects rather than dumb Web pages wherein the semantics of the objects are preserved between the server and the client, thereby giving users much more power and control over their knowledge experience. In addition, with the Web of the present invention, knowledge-workers are able to consume and act on information on their own terms because they will interactively author their own knowledge experiences via “dynamic linking” and “user-controlled browsing.”

[0435] The Information Agent (semantic browser) of the present invention is designed to co-exist with Today’s Web and to integrate with and augment all facets of private and public intranets as well as the Internet. The technology platform stacks of Today’s Web and the...

[0436] Information Nervous System of the present invention are summarized in FIG. 3. With reference to FIG. 3, the
stock for the Today’s Web has at the bottommost layers Structured Information Sources, including such information as the data stored in databases, and Unstructured Information Sources, including such information as documents, email messages, etc. Information in both of these layers is handled distinctly. No semantics are used at the Information Indexing Layer; rather, search engines based on keywords are used. The Logic Layer consists primarily of a database that allows programmability for searching, rules, view, triggers, etc. The Application Layer consists of server-side scripts that drive e-Business applications based on user input. At the topmost or Presentation Layer, Today’s Web has presentation information (in the form of Web pages) that is exposed via portals with a Web platform (e.g., browser). [0437] Apart from overlapping layers of processing, the present invention uniquely handles information from the bottommost level of operation in a manner that preserves the semantics of the underlying information sources. At both the Structured and Unstructured Information Sources Layers, the system 10 handles information uniformly, taking into account metadata and semantics associated with the information. At the Information Indexing Layer, information metadata and semantics are extracted from unstructured. The system 10 adds three additional platform layers not present in Today’s Web: Knowledge Indexing and Classification Layer, wherein information from both structured and unstructured sources are semantically encoded; Knowledge Representation Layer, wherein associations are created that allows maintenance of a self-correcting or healing Semantic Network of knowledge objects; and Knowledge Ontology and Inference Layer, wherein new connections and properties are inferred in the Semantic Network. At the Logic Layer a knowledge-base is created that allows for programmability at a semantic level. At the Application Layer, server-side scripts are used in association with the knowledge-base. These scripts dynamically generate knowledge objects based on user input, and may include semantic commands for retrieval, notifications and logic. This Layer may also include Smart Agents to optimize the handling of semantic user input. The Presentation Layer of the system 10 preserves the semantics that are tracked from the bottommost layers. Presentation at this Layer is dynamically generated on the client computer system and completely customizable. [0438] By the maintenance, integration and use of semantics in all technology layers, the present invention creates a virtual Web of actionable “objects” that directly correspond to “things” that humans interact with physically or virtually or, in other words, as familiar “Context Templates.” As opposed to Today’s Web, which is a dumb Web of documents, the present invention provides for a smart virtual Web of actionable objects that have properties and relationships, and in which events can dynamically cause changes in other parts of the virtual Web. [0439] The present invention provides a programmable Web. Unlike Today’s Web which is a dumb Web of documents, the Web of the present invention is programmable akin to a database—it is able to process logic and rules, and will be able to initiate events. [0440] While Today’s Web is encoded for human, and thus is focused primarily on presentation of static information, the virtual Web of the present invention is encoded primarily for machines, albeit ultimately presented to humans as the end of the knowledge delivery chain. The present invention provides an intelligent, learning Web. This means that the virtual Web of the present invention will be able to learn new connections and become smarter over time. The Web is dynamic, virtual and self-authoring, thereby providing much more power to knowledge-workers by intelligently and proactively making semantic connections that Today’s Web is unable to provide, thereby leading to a reduction in and eventual elimination of information loss. [0441] The Web of the present invention is a self-healing Web. Unlike Today’s web which has to be manually maintained by document authors, the present invention provides a Web that is self-maintained by machines. This feature rectifies broken links because the Web will fix disconnections in the network automatically. [0442] Finally, as will be set forth in greater detail below, the various embodiments of the present invention incorporate some or all of the axes of knowledge acquisition described above to provide substantial advantages over existing systems directed to Today’s Web or the conceptual Semantic Web. C. System Architecture and Technology Considerations [0443] 1. System Overview [0444] The present invention is directed to a system and method for knowledge retrieval, management and delivery. This system and method is referred to herein by the trademarked term Information Nervous System™. With reference to FIG. 4, at its highest level the system 10 includes a server 20 comprised of several components that work together to provide context and time-sensitive semantic information retrieval services to clients 30 operating a presentation platform (e.g., a browser) via a communication medium 40, such as the Internet or an intranet. The server components preferably include a Knowledge Integration Server (KIS) 50 and a Knowledge Base Server (KBS) 80, which may be physically integrated or separate. Within the system, all objects or events in a given hierarchy are active Agents 90 semantically related to each other and representing queries (comprised of underlying action code) that return data objects for presentation to the client according to a predetermined and customizable theme or “Skin.” This system contemplates wide variety of applications, as well as various means for the client to customize and “blend” Agents and the underlying related queries to optimize the presentation of the resulting information. Each of the preferred components of the system 10 of the present invention, as well as the interaction among the components, is described in greater detail below. [0445] 2. System Architecture [0446] The end-to-end system architecture for the Information Nervous System of the present invention is shown with reference to FIG. 5. FIG. 5 illustrates how the present invention provides multiple client access means of communication between the Information Nervous System XML, Web Service (KIS) and Smart Agents. In the preferred embodiment, this occurs via the Information Agent. In an alternative embodiment, the communication may occur programmatically via an Enterprise Knowledge Portal (e.g., Today’s Web access browser) or via an SDK layer that enables programmatic integration with a custom client. [0447] The system architecture for the KIS of the Information Nervous System, including components thereof, are shown with reference to FIG. 6. These components are described in greater detail below. [0448] 3. Technology Stacks [0449] The significant differences between Today’s Web and the conceptual Semantic Web are further highlighted by
reference to the technology stacks of each as shown with reference to FIG. 7. FIG. 7 is a side-by-side comparison of the high-level descriptive platform layers of Today's Web and the equivalents (where applicable) in the Information Nervous System of the present invention. FIG. 7 illustrates how scenarios in Today's Web map to scenarios in the Information Nervous System in certain instances, thus providing users with a logical migration path, but also highlights aspects of the Information Nervous System that do not exist in Today's Web.

[0450] 4. System Herogeneity

[0451] Heterogeneity is an advantage of the present invention. In the preferred embodiment, the KIS Agency XML Web Service is portable. This means that it supports open standards such as XML, XML Web Services that are interoperable (e.g., that employ the WS-I standard for interoperability), standards for data storage and access (e.g., SQL and ODBC/JDBC) and standard protocols for the information repositories from which the DSAs gather data (e.g., LDAP, SMTP, HTTP, etc.), etc.

[0452] For example, in a preferred embodiment, a KIS (on which an Agency is running) is able to:

[0453] Gather its “people” metadata from an LDAP store (using an LDAP DSA). This allows it to support Microsoft's Windows 2000 Active Directory, Sun's Directory Server, and other Directory products that support LDAP. This is preferable to having a platform-specific Active Directory DSA that uses platform-specific APIs to gather “people” metadata.

[0454] Gather its email metadata from an SMTP store (for email from any source for the system inbox). This allows it to support Microsoft Exchange, Lotus Notes, and other email servers (which support SMTP). This is preferable to having a platform-specific Microsoft Exchange Email DSA or a Lotus Notes Email DSA.

[0455] Gather its “event” metadata from a calendar store supporting an open standard like iCalendar and use a protocol such as Calendar Access Protocol (CAP). This allows it to support any event repository that supports the iCalendar or CAL protocol standard. This is preferable to having a platform-specific Microsoft Exchange Calendar or Event DSA or a Lotus Notes Calendar DSA, etc.

[0456] In an alternative embodiment, the KIS Agency may be configured to extract metadata stored in a proprietary repository (via an appropriate DSA).

[0457] To achieve heterogeneity, in the preferred embodiment, for client-server communications, the system 10 uses XML Web Service standards that work in an interoperable manner (across platforms). These include appropriate open and interoperable standards for SOAP, XML, Web Services Security (WS-Security), Web Services Caching (WS-Caching), etc.

[0458] In the preferred embodiment of the present invention, the semantic browser (also referred to by the trademarked term Information Agent™) is able to operate cross-platform and in different environments, such as Windows, .NET, J2EE, Unix, etc. This ability is consistent with the notion of a semantic user experience in that users do not and should not care about what “platform” the browser is running on or what platform the Agency (server) is running on. The semantic browser of the present invention provides users with a consistent experience regardless whether they are “talking” to a Windows (or .NET) server or a J2EE server. Users are not required to take any extra steps while installing or using the client based on the platform on which any of the Agencies they are interacting with is running.

[0459] The Information Agent preferably uses open standards for its Skins and other presentation effects. These include standards such as XSLT, SVG, and proprietary presentation formats that work across platforms (e.g., appropriate versions of Flash MX/ActionScript).

[0460] A sample, heterogeneous, end-to-end implementation of a preferred embodiment of the Information Nervous System of the present invention is shown with reference to FIG. 8. FIG. 8 illustrates the preferred embodiment of the Information Nervous System and illustrates the heterogeneous, cross-platform context for the present invention. The components shown in FIG. 8 are described in greater detail below.

[0461] 5. Security

[0462] The preferred embodiment of the Information Nervous System provides support for all aspects of security: authentication, authorization, auditing, data privacy, data integrity, availability, and non-repudiation. This is accomplished by employing standards such as WS-Security, which provides a platform for security with XML Web Service applications. Security is preferably handled at the protocol layer via security standards in the XML Web Service protocol stack. This includes encrypting method calls from clients (semantic browsers) to servers (Agencies), support for digital signatures, authenticating the calling user before granting access to an Agency’s Semantic Network and XML Web Service methods, etc.

[0463] The preferred embodiment that the present invention supports local (client-side) credential management. This is preferably implemented by requiring users to enter a list of their usernames and passwords that they use on multiple Agencies (within an Intranet) or over the Internet. The semantic browser aggregates information from multiple Agencies that may have different authentication credentials for the user. Supported authentication credentials optionally include common schemes such as basic authentication using a username and password, basic authentication over SSL, Microsoft’s .NET Passport authentication service, the new Liberty Alliance authentication service, client certificates over SSL, digest authentication, and integrated Windows authentication (for use in Windows environments).

[0464] In the preferred embodiment, with the users’ credentials cached at the client, the semantic browser uses the appropriate credentials for a given Agency by checking the supported authentication level and scheme for the Agency (which is part of the Agency’s schema). For example, if an Agency supports integrated Windows authentication, the semantic browser invokes the XML Web Service method with the logon handle or other identifier for the current user. If the Agency supports only basic authentication over SSL, the semantic browser passes either the username and password or a cached copy of the logon handle (if the client was previously logged on and the logon handle has not expired) in order to logon. The preferred embodiment employs techniques such as logon handle caching, aging and expiration on the KIS in order to speed up the authentication process (and logon handle lookups) and in order to provide more security by guarding against hijacked logon handles.

[0465] The Agency XML Web Service preferably supports different authentication schemes either implicitly (if the feature is natively supported by the server operating system or application server) or at the application-level by the XML
Web Service implementation itself. Alternative embodiments of the KIS Agency’s XML Web Service preferably employ a variety of authentication schemes such as basic authentication, basic over SSL, digest, integrated Windows authentication, and client certificates over SSL, and integrated .NET passport authentication.

[0466] 6. Efficiency Considerations

[0467] Client-Side and Server-Side Query and Object Caches. The present invention provides for query caches, which are responsible for caching queries for quick access. On the client, the client-side query cache caches the results of SQML queries with specified arguments. The cache is preferably configured to purge its contents after a predetermined amount of time (e.g., a few minutes). The amount of time is preferably set by modeling system usage and arriving at an optimal value for the cache time limit. Other parameters may also be considered, such as the data arrival rate on the Agency (in the case of per-Agency caches, which is another implementation option), the usage model (e.g., navigation rate) of the user, etc.

[0468] Caching improves performance because the client does not have to needlessly access recently used servers as the user navigates the semantic environment. In the preferred embodiment, the client employs standard XML Web Service Caching technologies (e.g., WS-Caching). In addition, on the client, there is preferably an object cache. This cache caches the results of each SQML query and is tagged with the resource reference (e.g., the file path, the URL, etc.). This optimizes SQML processing because the client can get the XML metadata for an SQML resource directly from the object cache, without having to access the resource itself. The resource may be the local file system, a local application (e.g., Microsoft Outlook), or an Agency’s XML Web Service. Like the query cache, the object cache may be configured to purge its contents after a set amount of time (e.g., a few minutes).

[0469] In an alternative embodiment, on the server, the server-side query cache caches the category results for XML arguments. This speeds up the query response time because the server does not have to ask the KDM to categorize XML arguments (via the one or more instances of the KBS that the KIS is configured to get its domain knowledge from) on each query request. In addition, the server can cache the SQI equivalents of the SQML arguments it receives from clients. This speeds up the query response time because the server would not have to convert SQML arguments to SQI each time it receives a request from a client. In the preferred embodiment, aggressive client-side caching is employed and server-side caching is avoided unless it clearly improves performance. This is because client-side caching scales better than server-side caching since the client caches requests based on its local context.

[0470] Virtual, Distributed Queries. The present invention employs virtual, distributed queries. This is consistent with its “dynamic linking” and “user-controlled browsing” functionality. The system does not require static networks that link—or massive individual databases that house—all the metadata for the system. This precludes the need for manual authoring and maintenance on a local or global scope. In addition, this precludes the need for integrated (or universal) storage, wherein all the metadata is required to be stored on a single metadata store and accessible through one database query interface (e.g., SQL). Rather, the present invention employs the principle of “Dynamic Access” via its use of XML Web Services to dynamically distribute queries across various agencies (in a context and time-sensitive manner), and to aggregate the results of those queries in a consistent and user-friendly manner on the client.

D. System Components and Operation

[0471] 1. Agencies and Agents

[0472] The present invention introduces a unique approach to using Agencies and Agents to retrieve, manage and deliver knowledge.

[0473] a. Agencies

[0474] In a preferred embodiment of the present invention, the Agency is an instance of the Knowledge Integration Server (KIS) 50 and is the invention’s equivalent of a Web site. An Agency is preferably installed as a Web application (on a Web server) so as to expose XML Web Services. An Agency will preferably include an Agency administrator. In a preferred embodiment of the present invention, an Agency has the following primary components:

[0475] A flag indicating whether the Agency supports or requires authentication (or both). If the Agency requires authentication, the Agency will require basic user information and a password and will store information on the type of authentication it supports. For Agencies that store user information, the Agency will also require user subscription information (for subscription to Agents of a specific Agency).

[0476] Structured stores of semantic objects (documents, email messages, etc.).

[0477] Corresponding to schemas for the respective classes.

[0478] Runtime components that respond to semantic queries—Components return XML to the calling application and provide system services for all the information retrieval features of the semantic browser.

[0479] Server-Side User State. In the preferred embodiment of the present invention, Agencies support server-side User State, which associates related concepts including “people” metadata and user authentication. Server-side User State facilitates many of the implementation details of the present invention, including the storage of user favorites (by semantic links between people objects and information objects), the inference of favorites in order to generate new links (e.g., recommendations), Annotations (that map users’ comments to information objects), and the inference of “experts” based on semantic links that map users to information (e.g., posted emails, annotations, etc.). Server-side User State is preferably used with some Context Templates like “Experts,” “Favorites,” “Recommendations,” and “Newsmakers.”

[0480] Client-Side User State. The Information Agent (semantic browser) preferably supports roaming of local client-side User State. This includes users’ Semantic Environment and users’ credentials (securely transferred). In the preferred embodiment, users are able to easily export their client-side User State to another machine in order to replicate their Semantic Environment onto another machine. This is preferably achieved by transferring users’ Agent list (recent and favorites), the metadata for the Agents (including the SQML buffers), users’ local security credentials, etc. to an XML format that serializes this state and enables the state to be easily transferred. Alternatively, an XML schema may be developed for all the local client-side User State. Caching the User State on a server and synchronizing the User State using common synchronization techniques can also facilitate roam-
ing. The semantic browser preferably downloads and uploads all client-side User State onto the server, rather than storing the state locally (in an XML file or a proprietary store like the Windows registry).

An Agent is the main entry point into the Semantic Network of the present invention. An Agent preferably consists of a semantic filter query that returns XML information for a particular semantic object type (e.g., documents, email, people, etc.). In other words, an Agent is preferably configured with a specific object type (described below). Agents can also be configured with a Context Template (described below). In this case, the query will return an object type, but it will incorporate the semantics of the Context Template. For example, Agents configured with a “Headlines” Context Template will be sorted by time and relevance, etc. Agents are also used to filter notifications, alerts and announcements. Agents can be given any name. However, in the preferred embodiment of the present invention, the naming format for most Agents is:

```
<agentobjecttype>.<semanticqualifier>.<semanticqualifier>
```

Agents can be named arbitrarily. However, examples of Agent names include:

- **All\_All**
- **Email\_All**
- **Documents\_Technology\_Wireless\.80211B\.All**
- **Events\_Upcoming\_Next\_Thirty\_Days\_All**

There will also be Domain Agents (see below) that may follow a different naming convention (see below). At the semantic browser of the present invention, a fully qualified Domain Agent name will have the format:

```
<agentobjecttype>.<semanticdomainname>.<categoriname> [Agency=<Agency url>, kb=<kb url>]
```

For example, the Email Domain Agent on the Agency http://research.Agency.asp configured with the category wireless.all from the knowledge-base ABC.com/kb.asp with the semantic domain name industries.informationtechnology will be fully named as:

```
Email\_Industries\_InformationTechnology\_Wireless\_All
```

```
```

The semantic browser of the present invention is preferably configurable to use only the Agent name or to include the “Agency” and “kb” qualifiers.

**Agent Types.** There are three primary types of Agents created on server 20: Standard Agents, Compound Agents, and Domain Agents. A Standard Agent is a standalone Agent that encapsulates structured, non-semantic queries, i.e., without domain knowledge (or an ontology/taxonomy mapping). For example, on the server, the Agent All\_Posted\_Today\_All is a simple Agent that is resolved by filtering all objects based on the CreationTime property. Standard Agents can also be more complex. For example, the Agent All\_Posted\_By\_Any\_Member\_Of\_My\_Team all may resolve into a complicated query that involves joins and sub-queries from the Objects table and the Users table (see below).

A Compound Agent contains other Agents and allows the Agency administrator to create queries that generate results that are the UNION or the INTERSECTION of the results of their contained Agents (depending on the configuration). Compound Agents can also contain other Compound Agents. In the presently preferred embodiment, Compound Agents contain Agents from the same Agency. However, the present invention anticipates the integration of Agents from different Agencies. By way of example, a Compound Agent All\_Technology\_Wireless\_All might be created by compounding the following Agents:

- **Documents\_Technology\_Wireless\_All**
- **Email\_Technology\_Wireless\_All**
- **People\_Experts\_Technology\_Wireless\_All**

As described above, a Domain Agent is an Agent that belongs to a semantic domain. A Domain Agent is initialized with an Agent query, just like any other Agent. However, this query includes the CATEGORIES table, which is populated by the Knowledge Domain Manager (see below). While the preferred embodiment of the present invention utilizes a KBS 80 having proprietary ontologies corresponding to a private Semantic Environment, the present invention contemplates integrated support of ontology interchange standards that will enable an Agency to connect to one or more custom private KBS, for example within an organization where the Agency was previously initialized with a proprietary ontology for that organization.

An example of a Domain Agent is Email\_Technology\_Wireless\_All. This Agent is preferably created with a knowledge source URL such as:

```
[category=technology\_wireless\_all@ABC.com/marketing\_knowledge\_asp]
```

This knowledge source URL corresponds to the Technology\_Wireless\_All category for the default domain on the knowledge base installed on the ABC.com/marketing\_knowledge\_asp Web service. This is resolved to the following HTTP URL: http://ABC.com/marketing\_knowledge\_asp\?category=“technology\_wireless\_all.” In this example, a fully qualified version of the category URL may be:

```
[category=technology\_wireless\_all@abccorp\_com/marketing\_knowledge\_asp?domain=information\_technology]
```

In this case, the category URL is qualified with the domain names.

**Domain Agents.** Domain Agents are preferably created via a Domain Agent Wizard, and the Agency administrator is able to add Domain Agents from the KBS 80 to the Semantic Network of the present invention. The Domain Agent Wizard allows users to create Domain Agents for specific categories (using a category URL) or for an entire semantic domain name. In the latter case, the Agency is preferably configured to automatically create Domain Agents as new categories are added to the semantic domain on the KBS. This feature allows domains and categories to remain dynamic and therefore easily adaptable to the user’s needs over time. When Domain Agents are managed in this fashion, the Agency is configurable so as to remove Agents that are no longer in the semantic domain. Essentially, in this mode, the Domain Agents are synchronized with the CATEGORIES table (which in turn is synchronized with the CATEGORIES list at the relevant KBS by the Knowledge Domain Manager, described below).

**Domain Agent.** A Domain Agent is initialized with a structured query that filters the data the Agent manages based on a category name or URL. In this situation, the structured query is identical to the queries for Standard Agents. An example of a resultant query for a category Agent is:

```
SELECT OBJECT FROM OBJECTS WHERE OBJECTID IN (SELECT OBJECTID FROM SEMANTICLINKS WHERE PREDICATETYPEID=50 AND
```
functions in an individual way because each Breaking News Agent is personal (“breaking” is subjective and depends on the user). For example, a user in Seattle perhaps would want to be notified on events in Seattle in the next 24 hours, events on the West Coast in the next week (during which time he or she can find a cheap flight), events in the United States in the next 14 days (the advance notice for most U.S. air carriers to get a modestly priced cross-continental flight), events in Europe in the next month (likely because he or she needs that amount of time to get a hotel reservation), and events anywhere in the world in the next six months.

[0516] In a preferred embodiment, the present invention automatically checks the Semantic Environment for breaking news by querying each Breaking News Agent or by querying the “Breaking News” Context Template. It will do this for all objects displayed in the semantic browser window. If a Breaking News Agent indicates that there is breaking news, the Information Agent object Skin so indicates by flashing the window or by showing a user interface that clearly indicates that there is an alert that relates to the object. When the user clicks on the breaking news icon, a breaking news pane or a Context Palette for the “Breaking News” Context Template is displayed allowing the user to see the breaking news, select the Breaking News Agent (if there are multiple with breaking news), select predicates, and select other options. An exemplar pane of a Breaking News Agent user interface is shown in FIG. 9. This sample user interface illustrates the popup menu in the context Results Pane. The sample shows a similar context pane as a Smart Lens (Agent-Object) popup context Results Pane (discussed below) except that the Agent is a Breaking News Agent.

[0517] Default Agents. In an alternative embodiment, each Agent exposes a list of default Agents. Default Agents are similar to the default page on a Web site; authors of the Agency determine which Agents they want users to always see. Alternatively, on the client, Default Agents may be invoked when users click on the root of the Information Agent’s Environment (which preferably corresponds to a “Home Agent,” for example, the equivalent of the “Home Page” on Today’s Web browser). Combined Default Agents may also be configured by users.

[0518] Default Special (or Context) Agents. In the preferred embodiment, the client or the Agency support a Default Special or Context Agent that maps to each Context Template (discussed below). These Agents preferably use the appropriate Context Template without any filter. For example, a Default Special Agent called “Today” returns all items on all Agencies in the “recent” and “favorites” lists (or on a configured list of Agencies) that were posted today. In yet another example, the Default Special Agent called “Variety” shows random sets of results for every Agency in the Semantic Environment corresponding to the “variety” Context Template.

[0519] Default Special Agents preferably function as a starting point for most users to familiarize themselves with the Information Nervous System of the present invention. In addition, Default Special Agents retain the same functionality as Smart Agents, such as use of drag and drop, copy and paste, Smart Lens, Deep Information, etc.

[0520] Horizontal Decision Agents. In the preferred embodiment, Agents utilized by the client to assist with user interaction, including:
[0521] Schedule Agent: The Schedule Agent intelligently ranks events based on the probability that particular users would want to attend the event.

[0522] Meeting Follow-up Agent: The Meeting Follow-up Agent intelligently notifies users when the time has come to have a follow-up meeting to one that occurred in the past. The Inference Engine (see below) monitors relevant semantic activity to determine whether enough change has occurred to warrant a follow-up meeting. Users preferentially use the previous meeting object as an Information Object Pivot to find the relevant knowledge changes (such as new documents, new people that might want to attend, etc.)

[0523] Task Follow-up Agent. The Task Follow-up Agent sends recommendations to users in response to tasks users perform (such as reading a document, adding an event to their calendar, etc.). The Agent ensures that users have constant follow-up. The recommendations are based on users’ profile, and the Agent prefers to use collaborative filtering to determine recommendations.

[0524] Customer Follow-up Agent. The Customer Follow-up Agent sends notifications to users based on customer activity. The Agent intelligently determines when the user needs attention (based on email received from the user, new documents that might aid user service, etc.)

[0525] Public versus Local Agents. Agents that are created by the Agency administrator are “Public Agents.” Agents created and managed by users are “Local Agents.” Local Agents can refer to remote Agencies via SQML that includes references to Agency XML Web Service URLs, or can refer to local Agencies that run a local instance of the KIS with a local metadata store.

[0526] Saved Agents—Users’ My Agents List. In the preferred embodiment, users are able to save a copy of an invoked Agent or a query result as a local Agent. For example, users may drag and drop a document on their hard drive to an Agent folder to generate a semantic relational query. Users could save that result as an Agent named “Documents.Technology.Wireless.RelatedToMyDocument.” This will then allow the user to navigate to that Agent to see a personalized semantic query. Users would then be able to use that Agent to create new personal Agents, and so on. Personal Agents can also be “published” to the Agency. Other users are preferably able to discover the Agent and to subscribe to it.

[0527] In the preferred embodiment, a local Agent is created by a “Save as Agent” button that appears on the client anytime a semantic relational query result is displayed. This is analogous to users saving a new document. Once the Agent is saved, it is added to the users’ My Agents list. An Agent responds to a semantic query based on the semantic domain of the Agency on which it is hosted. Essentially, a semantic query to an Agent is analogous to asking whether the Agent “understands the query.” The Agent responds to a query to the best of its “understanding.” As a further illustration, an Agent that manages “People” responds to a semantic query asking for experts for a document based on its own internal mapping of of people in its semantic domain to the categories in that domain.

[0528] Alternatively, the system client may be configured to use non-semantic queries. In this case, the Agency will use extracted keywords for the query. All Agents support non-semantic queries. Preferably only Agents on Agencys that belong to a semantic domain will support semantic queries. In other words, semantic searches degrade to searches.

[0529] Each Agent has an attribute that indicates whether it is “smart” or not. A Smart Agent is preferably created on an Agency if that Agency belongs to a semantic domain. In addition, a Smart Agent only returns objects it fully “understands.” In the preferred embodiment, when an Agency is installed, there are several default Smart Agents that the Agency administrator may optionally choose to install, including: All.Understood.All; Documents.Understood.All; and Email.Understood.All.

[0530] For example, Email.Understood.All only returns email objects that the Agency can semantically understand based on its semantic domain (or ontology).

[0531] The present invention preferably includes the capability for users to display all objects and only those the Agency understands.

[0532] Search Agents. A Search Agent is an Agent that is initialized with a search string. In the preferred embodiment, on invocation, the client issues the search request. A Search Agent is configurable so as to search any part of the Semantic Environment, including: Frequently Used Agents; Recently Used Agents; Recently Created Agents; Favorite: All [Saved] Agents; Deleted Agents; Agents on the local area network; Agents on the Global Agency Directory; Agents on any user-customized Agency directories; and All Agents in the entire Semantic Environment. The client issues the search request based on the scope of the Search Agent. If users indicate that they want the search to cover the entire Semantic Environment, the client issues the request to all Agents in the Semantic Environment Manager (see below) and all Agents on the local area network, the Global Agency Directory and user-customized Agency Directories.

[0533] Server-Side Favorite Agents. In yet another alternative embodiment, the Agency supports User States support Favorite Agents. In the analogous context of Today’s Web, a Web site allows users to customize their favorite links, stocks, etc. When initially queried, an Agency displays both its Default Agents and the Favorite Agents of the calling user (if there is a User State).

[0534] Smart Agents. A Smart Agent is a standalone Agent that encapsulates structured, semantic queries that refer to an Agency via its XML Web Service. In the preferred embodiment, user on the client are able to create and edit Smart Agents via a “Create Smart Agent” wizard that allows them to browse the Semantic Environment via the Open Agent dialog, and add links from specified Agencies. Essentially, this corresponds to users creating the SQML query from the user interface. In the preferred embodiment, the user interface only allows users to add links from the same Agency resource. However, users can create Agents of the same categories across Agencies, in addition to Special Agents and Blenders (which are also preferably cross-Agency). The user interface allows the user to add links using existing Smart Agents as Information Object Pivots provided that the Smart Agent refers to the same Agency for the current query. In a preferred embodiment the Open Agent dialog with the user interface controls allow a user to select link (predicate) templates, the links themselves, and the objects. In one embodiment, the Open Agent dialog allows users to browse the Semantic Environment and open an Agent. Agencies can be navigated in the Semantic Environment using the “Open Agent” dialog with the “Small Preview” view. An “Open” tool on the toolbar can show a user new options to open Agents form the Semantic
Environment or to import regular information (e.g., from the file system) to the Semantic Environment by creating Dumb Agents.

[0535] The link templates essentially allow the user to navigate predicate for the current object type using predefined filters, thus allowing the user to avoid going through all the predicates for the object type. Examples of link templates include: All; Breaking News (links that refer to time-sensitivity, e.g., “posted in the last”); Categorization; Definite (non-probabilistic links); Probable (probabilistic links); and Annotations.

[0536] In the preferred embodiment, the Open Agent dialog allows the user to select the object to “link to” and, depending on the type of the object, allows the user to browse the object (e.g., from a calendar control if it is a date/time, from a text box if it is text, from the file system if it is a file or folder path, etc.) The wizard user interface also allows the user to preview the results of the query. A temporary SQML entry is created with the current predicate list and that is loaded in a mini-browser window within the wizard dialog box. The user is able to add and remove predicates, and will also have the option of indicating whether he or she wants a union (an “OR”) or an intersection (an “AND”) of the predicates. The user interface will also check for duplicate predicates.

[0537] Once the user finishes the wizard to create the Smart Agent, the Smart Agent is added to the Semantic Environment and the SQML is also saved with the associated object entry. In the preferred embodiment, the user can later browse the Smart Agent using the Agent property inspector property sheet. This allows the user to view the simple Semantic Environment properties (e.g., name, description, creation time, etc.) and also to view the resource URL (the WSDL URL to the XML Web Service of the Agency being queried) and the predicate list. The user can edit the list from the property sheet.

[0538] Default Smart Agent. A Default Smart Agent is similar to a Default Special Agent except that it is based on information object types and not on Context Templates. By way of example, “Documents” would return all documents on all Agencies in the users’ Semantic Environment; “Email” would return all email messages in user’s Semantic Environment, etc.

[0539] Special Agent. A Special Agent is a Smart Agent created by users based on a Context Template (see below). A Special Agent is preferably initialized with an Agent name, albeit without a specific Agent reference. For example, a Special Agent “Email.Technology.Wireless.All1” may be created even if there are no Agents of that name in the Semantic Environment. Like a Search Agent, a Special Agent is scoped to search for any Agent with its name on any part of the Semantic Environment. In the preferred embodiment, when a Special Agent is invoked by users, the client searches for any Agents that bear its name. If or when it finds any Agents with the name, the client invoke the Agent.

[0540] In the preferred embodiment, users enter parameters consistent with a Context Template, indicating the category fillers (if required) and what Agency(ies) to query. These can be manually entered using the Open Agent dialog, or users can indicate that they want to query the “recent” Agencies, “favorite” Agencies, or both. In an alternative embodiment, users have the choice of selecting categories (if required) that are in the union or intersection of the selected Agency, or all categories known to the Global Agency Directory. In yet an alternative embodiment, users are able to select the information type (as opposed to a Context Template) and keywords to search (as opposed to predicates or categories).

[0541] Default Special Agents. In the preferred embodiment, the system client installs Default Special Agents that map to all supported Context Templates. By way of example, in the preferred embodiment, Default Special Agents including the following: Headlines, Breaking News, Conversations, Newsmakers, Upcoming Events, Discovery, History, All Bets, Best Bets, Experts, Favorites, Classics, Recommendations, Today, Variety, Timeline, Upcoming Events and Guide.

[0542] Custom Special Agents. In contrast to user-created Special Agents, Custom Special Agents are Special Agents specially developed and signed in order to guarantee that the Special Agents are safe, secure, and of high-performance. The present invention provides for a plug-in layer to allow organizations and developers to create their own custom blenders. An example of a custom blender is “All.CriticalPriority. All that relates to my most recent documents or email.” This Custom Blender may be implemented by an SQML file with a resource entry as follows:

```
<resource
type="nervana.uri"
agent://all.criticalpriority.all@localhost>
<-link
predicate="nervana.relevantto"
type="nervana.localsemanticizer"
recontituciones>

<-link
operator="or"
type="nervana.localsemanticizer"
recentemail>

</link>
</resource>
```

[0543] In the preferred embodiment, the Presenter (see below) resolves the “link” entry locally and initiates XML Web Service requests to the target resource with XML arguments corresponding to the newest documents or email messages. This allows the target Agent to focus on responding to semantic queries purely with XML filters without knowing the semantics related to filter origination. In an alternative embodiment, a Custom Blender such as the above example is a Default Agent.

[0544] Vertical Decision Agents. Vertical Decision Agents are Agents that provide decision-support for vertical industry scenarios.

[0545] Agent Schema. Agents operate within specified parameters and exhibit predetermined characteristics that comprise the Agent schema. Agent schemas may vary widely with being equally applicable within the technology of the present invention. By way of example only, the Agent schema of the preferred embodiment of the present invention is shown in FIG. 10-1. The present invention specifically contemplates the addition of further fields. For example, fields for category URL (or path) and Context Template name can be added to the Agent schema to provide the client and server quick access to the category and Context Template the Agent represents (if applicable). This is helpful for the Semantic Environment Manager to provide different views of Agents (by category, by context, etc.). This complements the existence of these fields in the SQML for the Agent (expressed via attributes and/or predicates). The AgentTypeIDIs included in the preferred embodiment are shown in FIG. 10-2. The AgentQueryTypeIDIs included in the preferred embodiment are shown in FIG. 10-3.
[0546] In the preferred embodiment, SQL query formats are used. However, multiple query formats, for example XML, XQuery, etc., are contemplated within the scope of the present invention.

[0547] The KIS 50 preferably hosts an Agents table (for server-side Agents) in its data store corresponding to this schema. FIG. 10-4 illustrates sample semantic queries that correspond to Agent names showing how server-side Agents are preferably configured on the KIS of the present invention.

[0548] As explained in greater detail below, Agents may optionally include their own Skins. An Agent Skin is represented as an XML file or equivalent Flash MX or ActionScript. If the Agent’s Skin URL is not specified, a default Skin for the Agent’s object type is assumed.

[0549] Agent Query Rules. Each server-side Agent query must be specified to return the OBJECTID column. Each table has this column for it is what links the Objects table with the tables for the derived object type. Objects and other tables are described in greater detail below.

[0550] Because each Agent query can form the basis of a sub-query, cascaded query or a join, it is preferable that each query follow this format. By way of example, the query for News.All will be may appear as “SELECT OBJECTID FROM NEWS” (where “NEWS” is the name of the table hosting metadata for news articles, with the “news” schema). As a result, the server 10 can then use this query as part of a complex query. For example, if the user drags and drops a document onto an Agent, the server can execute this query as:

```
[0551] SELECT OBJECTID FROM NEWS WHERE OBJECTID IN (SELECT OBJECTID FROM SEMANTIC.LINKS WHERE SUBJECTID IN (50, 67, 89) AND LINKSCORE>90)
```

[0552] This example assumes that the document is classified to belong to categories in the CATEGORIES table with object identifiers 50, 67, and 89 and that a link probability of 0.9 is the threshold to establish that a document belongs to a category. In this example, the document is used as a filter for the News.All query and the query text is used as part of the complex query.

[0553] Having a consistent standard for queries allows the semantic query processor to merge queries until they finally have to be presented. For example, each call to the semantic query processor must indicate what object type in which to return the results. The query processor then returns XML information consistent with the schema for the requested object type. In other words, the query processor preferably returns schema-specific results for presentation. Each query is stored at the semantic layer (to return an OBJECTID). To use the last example, when the user invokes the News.All Agent, the browser calls the query processor on the Agency XML Web Service. The query processor will then invoke the query and filter it with the ‘News Article’ object type, as such:

```
[0554] SELECT*FROM NEWS WHERE OBJECTID IN (SELECT OBJECTID FROM NEWS)
```

This returns all the fields for the News schema. The browser (via the Presenter) displays the information using the XSLT (or a presentation tool such as Flash MX or ActionScript) for either the Agent Skin or for a user-specified Skin (which will override the Agent Skin).

[0555] Query Virtual Parameters. Agent queries preferably contain special Virtual Parameter. A typical example may include: “%USERNAME%”. In this example, the Semantic Query Processor (SQP) resolves the Virtual Parameter to a real argument before invoking the query. An Agent People. MyTeam.All is configured with the SQL query:

```
[0556] SELECT*FROM USERS WHERE Division IN (SELECT Division FROM USERS WHERE Name LIKE "%USERNAME%")
```

[0557] In this example, the Agent name includes “MyTeam” even though the Agent can apply to any user. The %USERNAME% variable is resolved to the actual calling user’s name by the SQP. The SQL call is resolved to as follows:

```
[0558] SELECT*FROM USERS WHERE Division IN (SELECT Division FROM USERS WHERE Name LIKE JohnDoe)
```

In this example, JohnDoe is assumed to be the user name of the caller.

[0559] Simple Agent Search. Each Agent will support simple search functionality. In the preferred embodiment, a user is able to right-click on a Smart Agent in the Information Agent and hit “Search.” This will bring up a dialog box where the user enters search text. This creates the appropriate SQL with the associated predicate, e.g., “nervana.contains”, The present invention provides a simple, fast way for users to search Agents (and create Smart Agents from there) without going through the “Create Smart Agent” wizard and selecting the “contains text” predicate (which alternatively achieves the same result).

[0560] Agency Agent Views. An alternative embodiment of the present invention includes Agency Agent Views. An Agency Agent View is a query that filters Agents based on predefined criteria. For example, the Agent view “Documents” returns only Agents that manage objects of the document semantic class. The Agent view “Reuters News” returns a list of Agents that manage news objects with “Reuters” as the publisher. Agency Agent Views are important in order to give users an easy way to navigate through Agents. The Agency administrator is able to create and delete Agent views.

[0561] Agent Publishing and Sharing. The preferred embodiment makes it easy for Agents to be published and shared. This is preferably implemented by serializing the Semantic Environment into an XML document containing the recent and Favorite Agents, their schema, their SQL query, etc. and publishing the document to a publishing point. This XML document may also be emailed to colleagues, friends, etc. in order to facilitate the propagation and sharing of local (user-created) Agents. This is analogous to how Web pages are published today and how web URLs and links are shared by sending links and attachments via email.

[0562] 2. Knowledge Integration Server

[0563] The Knowledge Integration Server (KIS) 50 is the heart of the server-side of the system 10. The KIS semantically integrates data from multiple diverse sources into a Semantic Network and hosts Agents that provide access to the network. The KIS also hosts semantic XML Web Services to provide clients with access to the Semantic Network via Agents. To users, a KIS installation may be viewed as an Agency. The KIS is preferably initialized with the following properties:

```
[0564] Agency Name. Name of the Agency (e.g., “ABC”)
```

[0565] Agency Friendly Name. Full name of the Agency (e.g., “ABC Corporation”)

[0566] Agency Description. Description of the Agency
Agency System User Name. User name of the Agency. Each Agency is represented by a user on the directory of the enterprise (or Web site) on which it is installed. The system user name is used to host the system inbox (through which users will publish documents, email and annotations to the Agency). For authentication, the Agency must be installed on a server that has access to the system user account.

Agency Authentication Support Level. Indicates whether the Agency supports or requires user authentication. An Agency can be configured to not support authentication (in which case it is open to all users and does not have any User State), to support but not require authentication, and to require authentication, in which case it preferably indicates the authentication encryption type.

Agency User Directory Type. This indicates the type of user directory the Agency authenticates users against and where the Agency gets its user information from. For example, this could be an LDAP directory, a Microsoft Exchange 2000 User Directory, or a Lotus Notes User Directory on the Windows 2000 Active Directory, etc.

Agency User Directory Name. This indicates the server name of the Agency user directory (e.g., a Microsoft Exchange 2000 server name).

Agency User Domain Name. This indicates the name of the user domain for authentication purposes. This field is optional and included only if the Agency supports authentication.

Agency User Group Name. This indicates the name of the user group for authentication purposes. For example, an Agency might be initialized with the domain name “US Employees” and the group name “Marketing.” In such a case, the Agency will first check the user name to ensure that the user is a member of the user group, and then forward authentication requests to the user directory authenticator indicated by the user directory type. If the calling user is not a member of the user group, the authentication request is denied. This field is only valid if the Agency supports authentication.

Data Store Connection Name. This indicates the name of the connection to a database store. This could be represented by a JDBC connection name on Windows (or a JDBC name, etc.). The KIS will use the database referred to by the connection name to store, update, and maintain its tables (see below).

Dynamic Properties Evaluation. The Agency XML Web Service preferably exposes methods to return dynamic properties such as the list of semantic domain paths the server currently supports or “understands.” This allows users to browse Agencies on the client using their supported semantic domain paths or ontologies/taxonomies.

As illustrated with reference to FIG. 11, the KIS preferably includes the following main components: a Semantic Network, a Semantic Data Gatherer, a Semantic Network Consistency Checker, an Inference Engine, a Semantic Query Processor, a Natural Language Parser, an Email Knowledge Agent, and a Knowledge Domain Manager.

The Semantic Network is the core data component of the KIS. The Semantic Network links objects of the defined schemas of the present invention together in a semantic way via database tables. The Semantic Network consists of schemas and the Semantic Metadata Store (SMS). The Semantic Network is preferably comprised of two data schemas: Objects and Semantic.links. Additional data schemas may be included based on system requirements and enterprise needs. The SMS is preferably a standard database (SQL Server, Oracle, DB2, etc.) where all semantic data is stored and updated via database tables. The SMS preferably includes tables for each primary object type (described below).

By way of example, a sample Semantic Network directed towards an enterprise situation is shown with reference to FIG. 12, which illustrates the relationship between business users of the present invention and the various sources of and results of knowledge retrieval, management, delivery and presentation.

Objects. The Objects table contains every object in the Semantic Network. The “Object” can be thought of as the “base class” from which every semantic object type will be derived. The preferred schema of the Object type is shown with reference to FIG. 13-1. The ObjectID is a unique identifier that tags the object in the Semantic Network. Every object in the system will have a schema that is an extension of the Object schema. Alternatively, semantic object types (e.g., document, email, event, etc.) will have only the ObjectID field. When a query is invoked, the query processor can then aggregate information from the Object table and the specific semantic table to form the final results. The former approach (having each schema be an extension of the Object schema) results in better runtime performance since joins are avoided. However, the latter approach, while computationally more expensive, results in less wasted storage. The ObjectTypeID is preferably a number that resolves to a string that describes the hierarchy of the object type, e.g., “documents/documents”; “documents/analyst briefs”; and “events/meetings.”

The SourceID refers to the identifier for the Semantic Data Adapter (SDA) from which the object was gathered. The Semantic Data Gatherer (SDG) uses this information to periodically check whether the object still exists by requesting status information from the SDA from which the object was retrieved.

Semantic.links. The SMS preferably includes a Semantic.links schema (and corresponding database table) that will store semantic links. These links will annotate the objects in the other data tables of the SMS and will preferably constitute the data model for the Semantic Network. Each semantic link will have a semantic link ID. The Semantic.links table preferably includes the field names and types as shown with reference to FIG. 13-2. The SubjectID and SubjectTypeID are the object ID and object type ID of the object being linked from. The ObjectID and ObjectTypeID are the object ID and object type ID of the object being linked to. The LinkScore preferably ranges from 0 to 100, and represents the semantic strength of the link as a probability. These fields are exemplary only; more predicates are contemplated based on the particular object type as well as the user’s desire to semantic links. The preferred embodiment of the present invention provides the predicate type IDs shown in FIG. 14. The present invention contemplates the addition of further predicate type IDs.

By way of example, the semantic link “Steve reports to Patrick” will be represented in the table with a subject ID corresponding to Steve’s ID in the Users table, a predicate type of PREDICATETYPEID_REPORTSTO (see table
below), Patrick's object ID in the Users table, a link score of 100 (indicating that it is a “truth” and that the link is not probabilistic) and a Reference Date that qualifies the link.

The KIS creates, updates, and maintains database tables for each object type (via the SMS). The following illustrates preferred but nonexclusive list of primary and derived object types: Person, User, Customer, Category, Document, Analyst Brief, Analyst Report, Case Study, White Paper, Company Profile, E-Book, E-Magazine, Email Message, Email Annotation, Email News Posting, Email Distribution list, Email Public Folder, Email Public folder News-group, New Article, Event, Meeting, Corporate Event, Industry Event, TV Event, Radio Event, Print Media Event, Online Meeting, Arts and Entertainment Event, Online Course, Media, Book, Magazine, Multimedia, Online Broadcast and/or Online Conference. Object types are preferably expresses as hierarchical paths. The path can be extended, e.g., “events/meetings” can be extended with “qualified Meetings,” e.g., “events/meetings/company meetings.” This schema model is indefinitely extensible and configurable.

Virtual Information Object Types. Virtual Information Object Types are object types that do not map to distinct object types, yet are semantically of interest to users. An example is the “Customer Email” object type, which derives from the “Email” object type. This object type is “virtual” in that it does not have a distinct schema and, as a consequence, does not have a distinct table in the SMS on the KIS. Rather, it uses the “Email” table on the SMS, since it derives from the “Email” object type. Even though it is not a distinct object type, users will be interested in browsing and searching for “Customer Email” as though it were indeed distinct.

In the preferred embodiment, Virtual Object Types are implemented by storing the metadata in the appropriate object table on the SMS (in this case, the “Email” table, since the object type derives from “Email”). However, the resolution of queries for the object type is accomplished differently from regular queries for distinct object types. When the server SQL receives a semantic query request (via the XML Web Service) for a virtual information object type (such as “Customer Email”), it resolves the request by joining the tables that together form the object type. For instance, in the preferred embodiment, in the case of “Customer Email,” the server will resolve in query with the SQL sub-query:

```
[0585] SELECT OBJECTID FROM EMAIL WHERE
OBJECTID IN (SELECT OBJECTID FROM CUSTOMERS WHERE EMAIL.ADDRESS IN (SELECT EMAIL.ADDRESS FROM EMAIL))
```

This query corresponds to “Select all objects from the Email table that have an email address value that is also in the Customers table.” This assumes that “Customer Email” refers to email that is sent by or to a customer. Other definitions of the virtual object type are also possible and the query resolution is preferably consistent with the definition. The SQL preferably applies this sub-query to all queries for “Customer Email.” This sub-query essentially filters the Email table for those email messages that are from customers. This returns the desired result to the user with the illusion that there is a “Customer Email” table where there really is not.

The present invention contemplates a variety of schemas associated with each object type. Other schemas are in development that will have comparable applicability to the present invention. The “Document” schema, for example, may be extended with fields from the Dublin Core schema (http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2413.html) and other industry standard schemas. In yet another example, “News Article” schema may be an extension of the NewsML schema (http://www.newsmil.org). By way of example only, preferred user object schema made in accordance with the present invention are shown with reference to FIG. 15. All schemas preferably have as an identical subset the fields of the Object schema. Mailing Address TypeIDs preferably associated with the User (person) object schema includes those shown with reference to FIG. 16-1.

By way of example only, the preferred category object schema made in accordance with the present invention is shown with reference to FIG. 16-2.

By way of example only, the preferred document object schema made in accordance with the present invention is shown with reference to FIG. 16-3. The “Document Category” field refers to a proprietary category that is tagged with the document (by the document data source) and not to a semantic category managed by the KIS itself. The “Document Format TypeID” field refers to the type of document. The Print Media TypeIDs of the preferred embodiment are shown in FIG. 17-1, and the preferred FORMATTYPEID are shown in FIG. 17-2.

By way of example only, the preferred email message list object schema made in accordance with the present invention is shown with reference to FIG. 18-1. Email Priorities are preferably 0, 1, or 2, corresponding to low, medium, and high priority. The EmailTypeID preferably includes EMAILTYPEID, EMAIL._EMAILTYPEID_NEWSPOSTING and EMAILTYPEID_EMAILANNOTATION (values 1, 2 and 3). Exemplar tables showing the email distribution list and email public folder object schemas of a preferred embodiment of the present invention are shown in FIGS. 18-2 and 18-3, respectively. In the preferred embodiment, the PublicFolderTypeID includes those shown in FIG. 19-1.

By way of example only, the preferred event object schema message list object schema made in accordance with the present invention is shown with reference to FIG. 19-2. FIG. 19-3 shows the events types of a preferred embodiment of the present invention.

By way of example only, the preferred media object schema message list object schema made in accordance with the present invention is shown with reference to FIG. 20-1. FIG. 20-2 shows the media types of a preferred embodiment of the present invention.

By way of example, FIGS. 21-23 illustrate additional samples showing how objects are categorized and utilized in the preferred embodiment of the present invention. FIG. 21 illustrates root object container types. FIG. 22 illustrates a hierarchical schema for qualified object types. FIG. 23 illustrates samples of native container object type predicates. All types except the Person and Customer types preferably inherit all predicates from the root type “All Information.” The present invention provides for native container object type predicate templates, for example including for: All: Breaking News; Categorization; Author; Annotations; Definite Links; Probabilistic Links; and Popular.

b. Semantic Data Gatherer

In the preferred embodiment, the Semantic Data Gatherer (SDG) is responsible for adding, removing, and updating entries in the Semantic Network via the SMS. The SDG consists of a list of XML Web Service references. These form an Information Source Abstraction Layer (ISAL). Each of these references is initialized to gather data from via a Data Source Adapter (DSA). A data source adapter is an XML Web
Service that gathers information from a local or remote semantic data source for a given object type. It then returns the XML corresponding to object entries at the data source. All DSAs preferably support the same interface via which the SDG will gather XML data. This interface includes methods to:

0595 Retrieve the XML metadata for objects for a given start and end index (e.g., objects 0 through 49).

0596 Check whether there any objects have been added or deleted since a particular date/time (on the DSA’s time clock).

0597 Fetch the XML metadata for objects added or deleted since a particular date/time (on the DSA’s time clock).

0598 Check whether an object still exists in the semantic data source—by examining the XML metadata for the object (passed as an argument).

0599 If each call to the DSA XML Web Service will be stateless, the API should include information, preferably via a string with command parameters, which qualifies the request. For example, a DSA for an email inbox includes parameters such as the name of the user whose inbox is to be gathered. A DSA for a Web site or document store will have to include information on the URL or directory path to be crawled.

0600 Each DSA is required to retrieve information in the schema for its object type. Because a DSA must be implemented for a particular object type, the SDG will expect XML for the schema for that object type when it invokes a gather call to the DSA.

0601 The SDG is responsible for maintaining the integrity and consistency of all the database tables in the SMS (the Semantic Network). In this embodiment, the SDG is also referred to as a Semantic Network Manager (SNM). The database tables preferably do not contain redundant or stale entries. Because the SDG retrieves objects with well-known schemas the semantics of each of the object types is understood, and the SDG maintains the consistency of the tables accordingly. For example, the SDG preferably does not add redundant Document XML metadata to the DOCUMENTS table. The SDG uses the semantics of documents to check for redundancy. In the preferred embodiment this is accomplished by comparing the author name, creation date/time, file path, etc. The SDG also performs this check for other tables (e.g., EVENTS, CUSTOMERS, NEWS, etc.). For example, the SDG will perform redundancy checking for events by examining the title, the location, and the date/time. Other tables are maintained accordingly. The SDG will also update objects in the database tables that have been changed.

0602 The SDG is also preferably responsible for cleaning up the database tables. The SDG periodically queries the DSA to determine whether all of the objects in each table managed by the DSA still exists. For example, for a DSA that retrieves documents, the SDG will pass the XML metadata to the DSA Web service and query whether the object still exists. The DSA attempts to open the URL for the document. If the document does not exist anymore, the DSA will indicate this to the SDG. Individual DSAs, and not the SDG, are responsible for object validation to avoid security restrictions that are data source specific. For example, there might be data source restrictions that prevent remote access to local resources. In such a case, only the DSA XML Web Service (which is preferably running locally, relative to the data source) will have access to the data source. Alternatively, some DSAs might run on the Agency server, alongside the SDG and other server components, and retrieve their data remotely.

0603 Having the DSAs handle object validation also provides additional efficiency and security in that the DSA prevents the SDG from knowing the details of how to open each data source to check whether an object still exists. Since the DSA needs to know this (since it retrieves the XML data from the data source and therefore has code specific to the data source), it is more appropriate for the DSA to handle this task.

0604 The SDG preferably maintains a gather list that will point to DSA XML Web Service URLs. The KIS administrator is able to add, delete, and update DSA entries from the SDG gather list. Each gather list entry is preferably configured with:

0605 1. The name and XML Web Service reference of the DSA. This essentially will refer to a combination of the data source, the object type, and a reference to the XML Web Service that implements the DSA (e.g., via a WSDL web service URL). Examples include:

0606 a. Microsoft Exchange 2000 Email DSA. This DSA will gather email XML metadata from a Microsoft Exchange 2000 Inbox or Public Folder.

0607 b. Microsoft Exchange 2000 Calendar DSA. This DSA will gather event XML metadata from a Microsoft Exchange 2000 Calendar.

0608 c. Microsoft Exchange 2000 Users DSA. This DSA will gather users/people XML metadata from a Microsoft Exchange 2000 Directory.

0609 d. Microsoft Exchange 2000 Email Distribution List DSA. This SDA will gather email distribution list metadata from a Microsoft Exchange 2000 Directory.

0610 e. Lotus Notes Inbox. This DSA will gather email XML metadata from a Lotus Notes Inbox or Public Folder.

0611 f. Siebel CRM Database. This DSA will gather customer XML metadata from a Siebel CRM system.

0612 g. Web site. This DSA will gather document XML metadata from a Web site.

0613 h. File Directory or Share. This DSA will gather document XML metadata from a file directory or share.

0614 i. Saba E-Learning LMS Repository. This DSA will gather E-Learning XML metadata from a Saba Learning Management System (LMS) repository.

0615 j. Microsoft SharePoint Document DSA. This DSA will gather document XML metadata from a Microsoft SharePoint server workspace.

0616 k. Reuters News Repository. This DSA will gather News Article XML metadata from a Reuters news article repository.

0617 2. The description of the DSA gather entry.

0618 3. A string indicating initialization information for the DSA.

0619 4. The gather schedule—this indicates how often the SDG should “crawl” the DSA to gather XML metadata.

0620 In a preferred embodiment, the Agency is initialized with a user directory domain and group name. In this case, the SDG preferably automatically enters a gather list entry for the user directory DSA. For example, if the Agency is configured with a Exchange 2000 User Directory with Domain Name “Corp” and Address Book or group name “Everyone,” the SDG creates a gather list entry with the Exchange 2000 Users.
DSA (initialized with these parameters). Alternatively, the
Agency can be configured to obtain its user directory from
any email application server (e.g., Microsoft Exchange or
Lotus Notes). The SDG initializes gather list entries with an
Email Inbox and Calendar DSA for the system user (and
Email Knowledge Agent, described below). These three
gather list entry DSAs (Users, Inbox, and Calendar) are ini-
tialized by default. The Inbox is preferably used to store
Agency email postings and annotation and the Calendar DSA
is used to store events posted to the Agency by users. Other
custom DSAs can be added by the Agency administrator.

[0621] The SDG also keeps track of the last time the SDA
reported to it that objects have been added or deleted to or
from the data source. This date/time information is prefer-
ably based on the SDA’s clock. Each time the SDA reports
that there is new or deleted data, the SDG will update the date/time
information in its entry for the SDA and gather all the new or
deleted information in the SDA. The SDG will then update
the database tables.

[0622] The SDG preferably maps the XML information it
receives from the SDAs to the Semantic Network of the
present invention. The SDG stores all the XML metadata in
the database tables in the SMS. In addition, the SDG parses
the XML it receives from the SDA and, where necessary,
maps semantic links to specific XML fields. The SDG’s objects or
updates semantic links in cases where the XML includes
information that “links” objects together. For example, the
schema for an email object preferably includes fields includ-
ing “From,” “To,” “Cc,” “Bcc,” and “Attachments.” In the
case of the “From,” “To,” “Cc,” and “Bcc” columns, the fields in
the XML refer to email addresses (separated by delimiters
such as “,” “;”, or “)” or a space. In the case of the “Attachments”
column, this field will refer to the file paths of the files that are
attached to the email message (separated by delimiters such as
“,” “;”, or “)” or a space. This raw XML is stored in the EMAIL database table,
along with the other columns. In addition, the SDG parses
the fields of the email object and adds semantic links to other
objects that are identified by the contents of those fields. For
example, if the “to” field contains “john@foo.com” and the
attachments field contains the string “c:foo.doc, c:bar.doc,”
the SDG will process the email as follows:

[0623] 1. Find any object in the USERS table with the email
address “john@foo.com.” Also, search for other
USER objects with email addresses in the FROM, TO,
CC, and BCC fields.

[0624] 2. If any objects are found, add a semantic link
entry to the SEMANTICLINKS table with the email
object id as the subject and the appropriate predicate
object id as the object. In this case, the predicate PREDICATE-
TYPEID_CREATOR refers to the subject of the
email message. The predicate PREDICATETYPEDS-
ENTRY is used to link the email object and the USER
objects referred to by the content of the “to” field in the
email XML metadata. The predicate PREDICATETYPEID_COPIED
and PREDICATETYPEID_BLINDCOPYEDTO
are used to link objects in the “cc” and “bee” fields in similar
fashion.

[0625] In the case of attachments, the SDG extracts the
XML metadata for the attached documents. If an XML object
with the file path already exists in the SMS (or, in other words,
the Semantic Network), the SDG will update the metadata. If
the XML object does not already exist, the SDG creates a new
document object with the XML metadata. The SDG will add
an entry to the SEMANTICLINKS table with the email
object id as the subject, the new document’s object id as the
object, and the predicate PREDICATETYPED_AT-
TACHEDTO. This allows the user to be able to navigate from
an email message to its attachments and then use the attach-
ments as pivots to continue to browse the Semantic Network,
for example using semantic tools like the Smart Lens (discussed
below).

[0626] The SDG does not create any objects in the event for
which it does not find user objects that match the entries in the
XML fields. Preferably, the SDG gathers information from a
Directory SDA when a user is manually added to the Agency.
The Agency administrator preferably adds users to the
Agency via the user group on the Agency properties.

[0627] The following illustrates an example of mapping
raw email XML metadata to the Semantic Network.

```
<email from="johns@foo.com"
to="tina@nervana.net"
cc="steve@nervana.net"
bcc="patrick@nervana.net"
subject="Meeting this Friday"
body="Let us meet on Friday at 2pm"
attachments="c:foo.doc, c:bar.htm"/>
```

is converted to the object graph illustrated in FIG. 24.

[0628] c. Semantic Network Consistency Checker

[0629] The Semantic Network Consistency Checker (CC)
complements the consistency checking that is performed by
the SDG. As described above, the SDG maintains the integ-
ity of the database tables by precluding the addition of redu-
dant entries into the Semantic Network (from various data
sources). The CC also ensures the consistency of the
OBJECTS and SEMANTICLINKS tables. The CC periodic-
ally checks the OBJECTS table to ensure that each object
exists in the native table (preferably by checking the OBJECT-
TID field value). For example, a document object entry in the
OBJECTS table preferably also exists in the DOCUMENTS
table (with the same object ID). The CC removes any object in the
OBJECTS table without a corresponding object in the
native table (DOCUMENTS, EVENTS, EMAIL, etc.) and
vice-versa.

[0630] The CC is also responsible for maintaining the con-
istency of the SEMANTICLINKS table. The semantics of this
table are preferably as follows: A semantic link cannot exist
if either its subject (“linked from”) or its object (“linked to”) do not exist. To illustrate this, if object A links to object B
with predicate P, and either A or B is deleted, the link should
be deleted. The CC periodically checks the SEMANTI-
CLINKS table. If any of the subjects or objects has been
deleted, the CC deletes the semantic link entry.

[0631] Consistency checks may be implemented in code in
the KIS itself or as stored procedures or constraints at the
database level.

[0632] d. Inference Engine

[0633] The Inference Engine is responsible for adding
semantic links to the Semantic Network. The Inference
Engine employs Inference Rules, which consist of a set of
heuristics, to add semantic links based on ongoing semantic
activity. The Inference Engine is preferably allowed to
remove semantic links. Decision Agents (described below)
use the Inference Engine to assist knowledge-workers in
making decisions.
The Inference Engine operates by mining the Semantic Network and adding new semantic links that are based on probabilistic inferences. For example, the Inference Engine preferably monitors the Semantic Network and observes patterns in how email is sent, the type of email sent and by whom. The Inference Engine infers from this information background information, such as the expertise of the user, related to various subject matter categories within the monitoring purview of the Inference Engine. For example, the Inference Engine adds semantics links with the predicate PREDICATEYPEID_EXPERTON to indicate that a user is an expert in a particular category. The subject in this case will be a user object and the object will be a category object. To infer this, the Inference Engine is preferably configured to observe semantic activity for at least a certain period of time (e.g., two weeks), or to only infer links after users have sent at least a certain predetermined number of messages or authored a certain number of documents. The Inference Engine infers the new link by keeping statistics on the PREDICATEYPEID.Creator and PREDICATEYPEID.CONTRIBUTOR links.

By way of example, the Inference Engine may infer that users are experts on a category if:
- All categories of email messages they have written, this category is one of the top N (configurable).
- They have written email messages on the same category an average of M times or more per week (configurable).
- They have written at least N email messages (configurable) in the past P months (configurable).

More sophisticated inference models with which to accurately infer this data are contemplated. For example, probability distributions as well as statistical correlation models may be employed. Preferably these models will be developed on a per-scenario basis over time.

The Inference Engine is also responsible for removing links that it might have added. For example, if an employee changes jobs, he or she might “cease” to be an expert on a specific category (relative to other employees). Once the Inference Engine detects this (e.g., by observing email patterns), it removes semantic links that indicate that the person is an expert on the category.

Inferred semantic links are important for scenarios that involve probabilistic semantic queries. For example, in one embodiment of the present invention, using the Information Agent, users may drag and drop a document from their file-system onto an Agent (say, PeopleResearchAll). In this case, users will want to know the people in the Research department that are experts on the document. The browser will then invoke an SQL query with the Agent as resource (or subject), the predicate hasDocumentKnowledge, and the document path as the object. The Presenter will then retrieve the XML metadata for the document and call the XML Web Service, residing on the Agent that hosts the Agent, with the predicate ID and the document’s XML metadata as arguments. The server-side semantic query processor on the Agent processes this XML Web Service call and translates the call to a SQL query consistent with the data model of the Semantic Network. In this example, the call is preferably resolved as follows:

1. For all semantic domain entries in the KDM, call the corresponding KBS to categorize the document.
2. Map the returned objects to category objects in the Semantic Network (by comparing URLs)

3. Invoke a query using the query of the PeopleResearchAll Agent as a sub-query.

This query assumes that the object type ID for the user object type is 32, the predicate type ID value for PREDICATEYPEID_EXPERTON is 98, the document belonged to categories with the object ID 34, 56, and 78 and that the semantic link score threshold is 90.

e. Server-Side Semantic Query Processor

The server-side Semantic Query Processor (SQP) responds to semantic queries from clients of the KIS. The SQP is preferably the main entry point to the Semantic Network on the KIS (or Agency). The SQP is exposed via the Agency’s XML Web Service. The SQP processes the Agent semantic queries and generic (client-generated) semantic queries with semantic link filters (see below). For queries with server-side Agent filters, the Information Agent passes the Agent name and object index arguments to the SQP to be invoked. For example, the browser may ask for objects 0-24 on Agent Documents:Technology:Wireless:All. In this example, the SQP looks up the Agent query in the Agents table and invokes the query onto the database that hosts the Semantic Metadata Store (SMS). The Agent query is preferably stored as SQL or another well-known query format like XQuery or XML. The SQP can convert the query format to a format that the database (that holds all the tables) understands. Because most commercial databases understand SQL, it will preferably operate as the default Agent query format.

The Agent query preferably follows the query rules described above. Therefore, the query returns the object ID rather than the schema fields for the Agent’s object type. In the above-described example, Documents:Technology:Wireless:All invokes the Agent query “SELECT OBJECTID FROM DOCUMENTS WHERE . . . “. The SQP is responsible for issuing a query that is filtered with the Agent query, but which returns the actual metadata for the object type (in this case, the “document” object type). In this example, the query appears as follows:

SELECT * FROM DOCUMENTS WHERE OBJECTID IN (SELECT OBJECTID FROM DOCUMENTS WHERE . . . )

This query returns the data columns for the “document” schema for all the objects with an object ID that matches those in the original Agent query. The SQP reviews the metadata results of the database query and translates them to well-formed XML using the appropriate schema for the object type of the Agent (in this case, “document”). In the event that the database supports raw XML retrieval, the SQP optimizes the query by asking the database to give it XML results. This results in better performance since the SQP does not have to perform the extra translation step. The SQP passes the XML back to the caller via the Agency’s XML Web Service.

The SQP preferably handles more complex queries that are passed by the semantic browser (or other client of the
XML Web Service). By way of example, such queries may take the form of the following XML Web Service API:

```xml
InvokeSemanticQuery(
    Integer BeginIndex,
    Integer EndIndex,
    String AgentName,
    Integer NumberOfLinks,
    String OperatorNames[ ],
    String LinkPredicateNames[ ],
    String LinkTypeNames[ ]
);
```

This is preferably resolved to a SQL query:

```sql
SELECT TOP 25 * 
   OBJECTID IN (SELECT OBJECTID FROM OBJECTS WHERE CREATIONDATETIME="02/26/02" AND OBJECTID [RELATEDTO] OBJECT WITH ID 4576) AND OBJECTID IN (SELECT OBJECTS FROM EMAIL WHERE CATEGORY [IS] "WIRELESS")
```

This SQL example uses shorthand to illustrate the type of query that will be generated by the SQP. The SQP retrieves the XML and returns it to the caller. This XML is in the form of SRML (or Semantic Results Markup Language), which is the XML meta-schema definition for semantic query results in the preferred embodiment of the invention. Sample A shown in the Appendix hereto is a sample SRML semantisc results buffer or document. This is a sample of the XML that an Agency returns in response to a semantic query. The client Skin takes these results and generates presentation form them (using XSLT and/or script), based on the properties of the Skin and the Agent (object Skin/Context Skin/Blender Skin), the amount of display area available, disability considerations and other Skin attributes.

[0651] In this example, the [[]] symbols refer to arrays. The API takes a zero-based begin index, a zero-based end index, an Optional Agent name, an integer indicating the number of semantic links, an array of operator names, an array of link predicate names, an array of link type names, and an array of strings that refer to the link objects. If the Agent name is NULL ("[]"), the SQP processes the query "as is", without any preconceived Agent filter. This will be the case with queries that are wholly generated form the client. The arrays are variable sized because the "NumberOfLinks" parameter indicates the size of each array. The operator names include valid predetermined operators, including logical operators, which can be used to qualify queries in SQL or other query formats. Examples include terms or and term and. The link predicate names may include one or more predefined predicates (e.g., term:relevanth, term:reportsto, term:senuto, term:annotatedby, term:withcontext, etc.). The link type names indicate the type of link objects. Common examples include term:url and term:object. In the case of term:url, the link object string refers to a well-formed URL comprising objects:// or Agent://. In the case of term:object, the argument will be a well-formed XML metadata instruction referring to an object defined within the present invention. This object is preferably resolved from the client or from another Agency. The API returns a string that contains the XML results (in addition to the return value for the XML Web Service method called itself).

[0652] By way of example, the SQML with the data:

```xml
<resource type="term:url" Agent://all.criticalpriority.all@abc.com/Agency.asp>
  <link predicate="term:relevanth" type="term:object" object://4576 >
</link>
<link> operator="or" predicate="term:intersects" type="term:url" Agent://email.wireless.all@abc.com/Agency.asp>
</link>
</resource>
```

is resolved on the Agency located at the Web service on abc.com/Agency.asp to:

```xml
InvokeSemanticQuery(
  0, 24,
  "all.criticalpriority.all", 2,
  { "term:and", "term:or" }
);
```

[0653] f. Natural Language Parser

[0654] The Natural Language Parser (NLP) preferably converts natural language text to either an API call that the SQP understands or to raw SQL (or a similar query format) that can be processed by the database. The Natural Language Parser is passed text directly from the semantic browser or by email via the Email Knowledge Agent (see below).

[0655] g. Email Knowledge Agent

[0656] The KIS preferably includes one primary publishing component, referred to as the Email Knowledge Agent (or Enterprise Information Agent (EIA)). This Agent functions, in essence, as a digital employee, and preferably includes a unique email address (e.g., a custom name selected by the Agency administrator). The Email Knowledge Agent complements existing publishing tools such as Microsoft Office, SharePoint, etc. by adding a "Fire and Forget" method of publishing information and sharing knowledge. This is especially useful in cases where the person publishing the information does not know who might be interested in it.

[0657] In a preferred embodiment of the present invention, users send email to the Email Knowledge Agent to publish comments, annotations, documents, attachments, etc. The Email Knowledge Agent extracts meaning from the email and properly adds it to the Semantic Network. Other users are able to access published information via Agents of other platform presentation tools such as drag and drop, the Smart Lens, etc. (discussed below).

[0658] The Email Knowledge Agent is a system component that is created by the Agency administrator. The system user name is indicated when the server is first installed. The system user preferably corresponds to an email user in the enterprise email system (e.g., Microsoft Exchange, Lotus Notes, etc.) In this embodiment, the Email Agent has its own mailbox, calendar, address book, etc. These in turn correspond to the
objects on the Email Server for the system user. When the server is installed, the KIS installs the appropriate DSA for the system inbox (depending on the email application). The KIS preferably automatically adds a gatherer list entry in the SDG indicating that the system inbox should be periodically crawled for email.

**[0659]** Because the Email Knowledge Agent is a first-class email address, it also serves as a notification source and a query source (for natural-language and instant messaging). Notifications from an Agency are preferably sent by the Email Knowledge Agent (indicating that there is new and relevant information the user might be interested in, etc.). The Email Knowledge Agent may also receive email from users as natural-language queries. These messages are parsed by the SQP and processed. The XML results are preferably sent to the user as an HTML file (with the appropriate default Skin) generated with XSLT processed over the XML results of the natural-language query.

**[0660]** Because the Email Knowledge Agent is a regular familiar component or "employee," the Agency administrator preferably adds the address to distribution lists. This step allows the SDG to semantically index all the email in these distribution lists, thereby populating the Semantic Network by seamlessly integrating the Email Knowledge Agent into distribution lists useful to users. This is a very seamless way of integrating the digital Information Nervous System of the present invention with the way people already work in an organization.

**[0661]** Annotations. The Email Knowledge Agent is preferably used to publish annotations. In the present invention, annotations are preferably email messages. In the preferred embodiment, the annotation object type is a subclass of the email object type. This allows users to use email, typically the most common publishing tool, to annotate objects in the semantic browser. Users are able to annotate objects and add attachments to the annotations. These attachments are semantically indexed by the SDG on the KIS. This makes possible scenarios where a user is able to navigate from, say, a document, to an annotation, to its document attachment, to an article on Reuters, to an industry event that starts next week.

**[0662]** The process described for semantically indexing email (by mapping the email XML schema to the Semantic Network) is applied to annotations. However, in the case of annotations in a preferred embodiment of the present invention, additionally processing is desirable. Specifically, when the user clicks "Annotate" on an object in the Presenter window in the semantic browser (described below), the browser loads the registered email client on the local machine (e.g., Microsoft Outlook, Microsoft Outlook Express, etc.). The "to" field is populated with the address of the system user for the Agency that hosts the object. The subject field is populated with a special string, for example, "annotation: object=objectid". When the email arrives in the Email Knowledge Agent's inbox, the DSA for the email inbox will pick it up (e.g., via a server event). The SDG retrieves the new email XML metadata from the DSA by receiving an event, or from the DSA the next time it asks the DSA for more data. In a preferred embodiment, this polling process occurs frequently. The DSA returns the XML metadata of the email object, oblivious to the fact that the email object refers to an email object type or an annotation object type. The SDG processes the email XML metadata, and examines the "subject" field. If the SDG "sees" the "annotation:" prefix, it knows that the email is actually an annotation, and proceeds to extract the object ID argument from the subject text. The SDG updates the Semantic Network for remaining email messages (adding each message to the OBJECTS and EMAIL tables, adding semantic links for the "from", "to", "cc", "bcc," and "attachments" fields, where necessary, etc.). In the preferred embodiment, the SDG performs an extra step. Specifically, it adds a semantic link entry that links the email object with the object indicated by the object ID argument in the subject text (with the PREDICATETYPE=ANNOTATES predicate).

**[0663]** With the present invention, an annotation is treated as another semantic link with a special predicate. As a result, all the semantic features apply to annotations, such as semantic navigation via semantic links, semantic queries, etc. For example, a user can query for all annotations written by every member of his or her team in the last six months. This can be accomplished in the semantic browser by dragging, for example, the Annotation All on top of the Agent People.MyTeam. All and then sorting the results, or by creating a Smart Agent, which in turn invokes the "Create Smart Agent" wizard to create the query.

**[0664]** h. Knowledge Domain Manager

**[0665]** The Knowledge Domain Manager is the component on the KIS that is responsible for adding and maintaining domain-specific intelligence on the Semantic Network. The KDM essentially "annotates" the Semantic Network with domain-intelligence. The KDM is initialized with URLs associated with one or more instances of the Knowledge Base Server (KBS), which in turn effectively stores "knowledge" for one or more semantic domains. The KBS has ontology and categories corresponding to taxonomy for each semantic domain that it supports. In addition, an Agent with a semantic domain (connected to a KBS) responds to semantic queries. If an Agent does not belong to a semantic domain, it cannot correspond to semantic queries (that require an ontology or taxonomy). Rather, it only responds to keyword-based queries (albeit it will still provide context and time-sensitive retrieval services, but the available contexts will be limited).

**[0666]** Each entry in the KDM is a semantic domain entry. The semantic domain entry has the URL to the KBS and a semantic domain name. The semantic domain name maps to a specific ontology on the KBS. In the preferred embodiment of the present invention, semantic domain names follow the convention:

**[0667]** \(<\text{Top Level Domain Name}>>.<\text{Secondary Level Domain Name}>. . .\)

Examples of semantic domain names include: Industries, Industries\>Pharmaceuticals\>LifeSciences, Industries\>InformationTechnology, General\>Sports, Basketball\>NBA, General\>Sports\>Basketball\>CBA.

**[0668]** Alternatively, semantic domains names can be referred to as "domain paths" as long as they are fully qualified. Full qualification is achieved by adding an Internet domain name prefix to the beginning of the path. This indicates the "owner" or "source" of the semantic domain. For example, "Nervana.NET\>Industries\>Pharmaceuticals" refers to "industries\>Pharmaceuticals" semantic domain according to the "Nervana.NET" Internet domain name. In another example, "Reuters.com\>Sports\>Basketball" refers to "Sports\>Basketball" on "Reuters.com." Using this approach, domain names and paths are maintained globally unique.

**[0669]** The Knowledge Domain Manager (KDM) periodically requests each KBS in its domain entry list for the categories in the knowledge domain. The KDM is preferably implemented as an XML Web Service on the KIS. The KDM
includes configuration options for each semantic domain entry. One of these options may include the schedule with which the KDM will update the Semantic Network with domain-specific intelligence corresponding to the semantic domain entry. For example, the Agency administrator may configure the KDM (via the KIS) to crawl a semantic domain on a KBS every day at 1 pm. The update schedule should be consistent with how often the administrator believes the ontology or taxonomy on the KBS changes.

[0670] The KIS periodically invokes the KDM periodically and asks it to update the CATEGORIES table. In the preferred embodiment, the KDM calls the KBS (via an XML Web Service API call) to obtain updated categories for the semantic domain name in the semantic domain entry, which corresponds to a particular taxonomy. An example of an API call follows: GetCategoriesForSemanticDomain(String SemanticDomainName). The KBS returns an XML-based list of all the categories in the semantic domain referred to by the semantic domain name. This XML list is consistent with the CATEGORIES schema shown above (category URL, name, description, the KBS URL and the semantic domain name). The KDM updates the CATEGORIES table with this information. For category entries that already exist in the table, the KDM updates the name and description. For new entries, the KDM requests a new object ID from the object manager and assigns that to the category entry. Since, in the preferred embodiment, a category is an “object,” it inherits from the Object type and therefore has an object ID.

[0671] The KDM synchronizes the CATEGORIES table to the CATEGORIES list on the KBS (for a particular semantic domain) by deleting entries in the CATEGORIES table not present in the new list after examining the URL of the category entries and obtaining the relevant KBS URL and semantic domain name. If a semantic domain entry is deleted from the KIS, the KDM deletes all category entries with a corresponding semantic domain name and KBS URL. Essentially, this will be akin to ridding the Agency of existing knowledge.

[0672] The KDM periodically categorizes all “knowledge objects” in the Semantic Network based on its semantic domain entries. When new objects are added to the Semantic Network by the SDG, the SDG requests that the KDM categorize the objects. The KDM enumerates all KBS instances in its semantic domain entries and invokes XML Web Service calls with the XML of the object as the argument. In the preferred embodiment, the KBS returns a result in an XML buffer similar to:

```
<results>
  <result categoryurl="/category://foo" score="91" />
  <result categoryurl="/category://bar" score="91" />
  <result categoryurl="/category://foosbar" score="100" />
</results>
```

[0673] This information indicates the semantic categorization weights of the XML object for the categories in the semantic domain on the KBS. In a preferred embodiment of the present invention, the semantic entry is initialized with a threshold (0-100) indicating the minimum weight that the KDM should request from the KBS. The KBS returns scores that exceed the predetermined threshold. The KDM annotates the Semantic Network based on these categorization results. This is preferably accomplished by adding or updating a semantic link with the predicate type ID of “belongs to category” with the object ID of the category in the result. The KDM will update the SEMANTICLINKS table. Assuming by way of example that the object that is categorized has an object ID value of 56, the update query appears as follows:

```
UPDATE SEMANTICLINKS SET LINKSCORE = 91 WHERE OBJECTID=56 AND PREDICATETypeID = 67 AND SUBJECTID IN (SELECT OBJECTID AS SUBJECTID FROM CATEGORIES WHERE URL LIKE "/CATEGORY://FOO")
```

[0674] The KDM periodically scans and categorizes all the “knowledge objects” (documents, news articles, events, email, etc., preferably not including objects like people). This process preferably occurs even if an object in the Semantic Network has previously been categorized as the KBS might have become “smarter” and therefore provides superior categorization. In such a case, the results could change even if the same categorization request is repeated. This will occur, for example, if the ontology on the KBS has been updated. Thus, in the preferred embodiment, categorization will be performed both when an object is added to the Semantic Network by the Semantic Data Gatherer and periodically to ensure that the Semantic Network has the most up-to-date domain knowledge.

[0675] i. Other Components

[0676] The Favorite Agents Manager. On Agencies that support User States, a Favorite Agents Manager manages a list of per-user favorite Agents. In the preferred embodiment, the Favorites Agent Manager stores a mapping of user names to favorite Agents in a UserFavoriteAgents table.

[0677] Compound Agent Manager. A Compound Agent Manager manages the creation, deletion, and update of compound Agents. As described above, compound Agents are Agents that are comprised of other Agents in the system, and are initialized to return the union or intersection of the query results in the contained Agents. The Compound Agent Manager manages all compound Agents in the system and maps compound Agents to the Agents they contain via the CompoundAgentMap table.

[0678] The Compound Agent Manager exposes functions to create compound Agents, delete, rename, add to and remove Agents from them, and indicate whether a union or an intersection is desired. Compound Agents can be added to other compound Agents. On invocation, the semantic query processor asks the Compound Agent Manager for its compound query. The Compound Agent Manager navigates through its Agent map graph and returns a complex query of all the queries of all Agents that it contains. If Agents are deleted, compound Agents “pick up” the new state when they are invoked, ignoring the Agent query. In other words, the compounding of queries is only done for Agents that still exist. If the compound Agent observes that one of its Agents has been deleted, it will delete the entry from its map.

[0679] User Profile Manager. The User Profile Manager (UPM) preferably uses the Inference Engine to infer the user’s profile on an ongoing basis. The UPM annotates the Semantic Network based on feedback from users as to their explicit preferences. In the preferred embodiment, this process involved use of the PREDICATEISINTEREST-
EDI\textsuperscript{N} predicate. The UPM infers semantic links and annotate the Semantic Network with the PREDICATE\textsuperscript{ID}_IS\text{\textemdash}LIKELY-TO\text{\textemdash}BE\text{\textemdash}INTERESTED\text{\textemdash}EDIN predicate. All query results to the user will be qualified (out-of-band) with a query to the Semantic Network for the PREDICATE\textsuperscript{ID}_IS\text{\textemdash}LIKELY-TO\text{\textemdash}BE\text{\textemdash}INTERESTED\text{\textemdash}EDIN predicate. Query results are based on the user’s habits, as the Inference Engine learns them over time.

Alternatively, the UPM may be configured with user profile information stored in the User State Store (USS). This is information manually entered at the client indicating the user’s preferences. This information is transferred and stored at the server that the user is interacting with. These preferences are tied to different schema. For example, for documents, the schema may be based on the preferred categories. For email messages, the schema may be based on preferred categories, authors, or attachments. These are two of many possible examples. The UPS annotates the Semantic Network based on the manually entered information in the USS.

**Server Notification Manager.** The Server Notification Manager (SNM) is responsible for batching server-side notifications and forwarding them to users. In the preferred embodiment, users register for server-side notifications at the Agent level. Each Agent is capable of firing notifications of its query results. The Server Notification Manager determines how to filter the query results and format them for delivery via email, voice, pager or any other notification mechanism, e.g., the Microsoft .NET Alerts notification services. The Server Notification Manager maintains information on the last time users “read” the notification. This is preferably indicated from the client via a user interface. The SNM preferably only notifies a user when there is new information on the Agent since the last “read” time for the particular user.

**Agent Discovery.** Using multicast-based Agent discovery, each Agency sends multicast announcements indicating its presence on the local multicast network. The Agency administrator sets the multicast TTL. The present invention preferably uses either use the Session Announcement Protocol (SAP) with a well-known port of 9875 and a TTL of 255, or a proprietary announcement port with a customizable TTL. For details on SAP, see [http://sunsite.cnlab-switch.ch/ftp/doc/standard/rfc/29xx/2974.php](http://sunsite.cnlab-switch.ch/ftp/doc/standard/rfc/29xx/2974.php), which is incorporated by reference.

**The Information Agent preferably includes a listener component that receives SAP announcements. In the preferred embodiment, the announcements are sent as XML and will include the following information:** The server ID (this is a unique identifier); The server URL (this is the HTTP URL to the Agency’s XML Web Service); The announcement period (T) — this indicates the time between each announcement; and Whether there are any new Agents in the Agency since the last announcement and the last Agent creation time (on the Agency’s clock).

Each Agency sends the XML announcement and uses Forward Error Correction (FEC) or Forward Erasure Correction to encode the packet. This makes the system robust to dropped packets. Alternatively, the Agency can be configured to send the XML announcements several times in succession (per announcement).

**The Information Agent multicast listener exposes directory-like semantics to the Semantic Environment Manager. The listener aggregates all the XML announcements from the Agencies from which it receives announcements. It will also cache the last time it received an announcement from each Agency. The listener flags Agencies that it thinks might be dead or inactive. It does this when it has not heard from the Agency for a time longer than the Agency’s announcement period. The listener might be configured to wait for several periods before flagging the Agency as inactive. This will handle the case of dropped announcements (due, perhaps, to traffic congestion). The listener will update the Agency list in the Semantic Environment Manager each time it receives announcements.**

The Semantic Environment Manager periodically inquires of the listener whether there are any new Agents. The Semantic Environment Manager checks the Agency list and asks each Agent that is active whether it has new Agents. The Semantic Environment Manager qualifies this request with the Agency’s last Agent creation time maintained locally and the current time based on the Agency’s clock. The Agency responds and also sends the new value of the last Agent creation time. The Semantic Environment Manager caches this value in the Agency entry. If there are new Agents, the browser informs the user via a dialog box and asks the user whether he or she wants to view the new Agents.

**The present invention also supports Agency announcements using a peer-to-peer Agent discovery. In this model, announcements are sent either to a directory server that all clients check or directly to the clients via a standard peer-to-peer publishing protocol.**

**Knowledge Base Server**

**The Knowledge Base Server (KBS) is the server that hosts knowledge for the KIS. In most applications, many instances of the KIS will be deployed, but only few (or one) KBS will be deployed for any given organization. This is because KBS can be reused (they are domain-specific but data-independent). For example, a pharmaceutical firm might deploy one KBS initialized with a pharmaceutical ontology, but have several KIS installations; perhaps per employee division or per employee group. The KIS preferably includes the following components:**

1. One or more ontologies that correspond to one or more semantic (knowledge) domains. A semantic domain is referred to using a semantic domain name. This is a name that refers to a domain path within a semantic hierarchy. Examples include Industries, Technology, Industries, Pharmaceuticals, Life Sciences, and General, Sports, Basketball. These names or paths may also be globally and uniquely qualified (e.g., with Internet domain names) as previously discussed.

2. One or more taxonomies that correspond to the supported semantic domains. These taxonomies contain a hierarchy of category names.

3. A categorization engine that take a piece of text or XML and the semantic domain name with which the categorization is to be performed, and returns the categories in that domain that the text or XML belong to, along with the categorization scores (on a scale of 0-10 or, preferably, 0-100).

4. An XML Web Service that exposes APIs to add new supported semantic domains (and corresponding ontologies and taxonomies), to enumerate the categories for a given semantic domain, and to categorize a text or XML data blob.

5. An XML Web Service reference to another KBS from which the KBS gets its knowledge. In this mode, the KBS acts as a proxy. The KBS can be initial-
ized to act as a proxy and to get its supported semantic domains, ontologies, and taxonomies from another KBS.

As explained above, the KIS (via the KDM) periodically sends XML objects to the KBS to categorize them for a given semantic domain.

4. Information Agent (Semantic Browser Platform)

a. Overview

The system client, in the preferred embodiment the Information Agent of the present invention, includes the semantic browser components and user interface that provide a semantic user experience. In the preferred embodiment, the Information Agent provides the following high-level services:

- Allow users the power of context and time-sensitive semantic information retrieval via local and remote Information Agents.
- Allow users to discover information on local and remote Agencies that are exposed via Agents through the XML Web Service of the present invention. This information is preferably classified into well-known semantic classes such as documents, email, email distribution lists, people, events, multimedia, and customers.
- Allow users to browse a semantic view of information found via Agents of the present invention.
- Allow users to publish information to an Agency.
- Allow users to dynamically link information on their hard-drive, local network or a specific Agency with information found on Agents from another Agency. This facilitates dynamic e-linking and user-controlled browsing.

An advantage of the Information Agent of the present invention is that users open up Agents similar how users open up documents from their file-system namespace. The Information Agent will have its own environment that opens up semantic "worlds" of information. For example, ABC company may have an internal KIS Agency that has Agents for internal documents, email, etc. In addition, third-parties may host Agencies on the Internet to hold information on industry reports, industry events, etc. In a preferred embodiment of the present invention, ABC company employees open Agents to discover information on the Internet that relates to their work as well as to semantically relate information that is internal to ABC company to information that is external but relevant to ABC company.

b. Client Configuration

In the preferred embodiment, the system client is able to semantically link information found locally as well as on remote Agencies. This is preferably accomplished through the use of an exposed Semantic Environment comprised of Agencies from a Global Agency Directory, Agencies on the local area network (published via multicast or a peer-to-peer publishing system) and Agencies from a custom Agency Directory using Agent Discovery. The preferred client configuration is based on a framework having Agents and local Agencies, and includes a Semantic Environment Manager, which manages locally saved Agents and Favorite Agents, essentially integrating the history and favorites metaphors. The Semantic Environment Manager uses Semantic Query Documents within the Semantic Environment to present knowledge to users via the Semantic Environment Browser. The client configuration will also include the Agent Discovery information (e.g., Agency lists, Agency directory information, etc.).

Client Framework Specification

Overview. The client framework specification provides the service infrastructure for the Information Agent user interface, and defines basic services and interfaces, includes core user interface components, and provides an extensible, configurable environment for the main building blocks of the user interface of the Information Agent. This section described the client framework specification according to a preferred embodiment of the present invention. The Framework Core defines base services, configuration, preferences and security mechanisms. The Core User Interface Components define the user interface services and modules that support server and Agent configuration, control and invocation, and some configuration for the Semantic Browser Framework. The Core User Interface Components are implemented as a Windows Shell extension and associated user interface (described below). The Semantic Browser Framework provides base query and results management services, and the framework for results presentation. The specifics of the user interface related to semantic object presentation are preferably configurable and extensible; even default presentation support is provided as a pre-installed "extension." The Semantic Browser Framework is preferably implemented as a set of behavior extensions to existing platforms used in Today's Web (e.g., Internet Explorer), and leverages the supported XML, XSLT, HTML/CSS and DOM functionality.

Context. The client framework builds upon semantic services components of the present invention including semantic query support, context and time-sensitive semantic processing and linking of information, etc. The client framework is preferably built as a shell extension and platform (e.g., Internet Explorer) extensions, which provides functionality to users in the context of their existing tools and environment. For example, the Information Agent may be implemented as a Shell Extension (which extends the Windows Shell and employs the standard Explorer view and user interface models). In an alternative embodiment, the present invention is equally applicable in a standalone semantic browser application.

Requirements. The preferred requirements for the client framework relate to flexibility and extensibility. This ensures that the user interface can be easily and quickly adapted as there are more information object types, user profiles, etc. Included are the following:

- Provide support for Skins to manage the entire set of query results.
- Allow for a wide range of approaches, include lists, tables, time slides, etc.
- Provide a screen-saver (or equivalent) mode.
- Provide support for Skins that can be associated with an object class.
- Ensure that there is a default Skin that can handle all classes.
- Skins should be as simple as XSLT, but should allow script support, and possibly even code (with appropriate security restrictions).
- Provide support for browsing the Semantic Environment in the results view (to complement the Agent Tree View), including Agents (Smart, Dumb, and Special), Agencies, and Blenders.
- Provide well-defined interfaces between components, and ensure that all communication must occur via the framework.
[0719] Provide a solid security model throughout the framework.

[0720] Framework Core

[0721] Semantic Environment Manager (SEM). The SEM manages the creation, deletion, updating and browsing of Agents, Blenders, and Agencies on users’ local machines. In addition, the SEM is responsible for listening to Agency multicast announcements, browsing Agencies on the enterprise directory (e.g., via LDAP), browsing Agencies on a custom directory, and browsing Agencies on the Global Agency Directory.

[0722] The SEM includes a storage layer that stores the metadata of every Agent on the system, including all the Agent attributes (such as the Agent name, description, creation time, last usage time, the Agent type (Smart, Dumb, Special, etc.), information object type the Agent represents (for Agents created based on information type), the context type the Agent represents (for Special Agents or Agents created based on a Context Template), the attributes of the Agent, a reference to the XSLT or other script file that represents the Agent’s Skin (including filter/sort preferences and other presentation schemes), the notification information and method (if requested for the Agent), and the buffer or file-path/URL to the Agent’s SQL query. The Information Agent (semantic browser) may store this Agent metadata in a local database, a store like the Windows registry, or in an XML file store on the local file-system.

[0723] The SEM also uses the Agent attribute to indicate whether an Agent is a Favorite Agent. In addition, the SEM automatically deletes Agents that are not favorites and which are older than a configurable age limit (e.g., two weeks).

[0724] The Information Agent’s Shell Extension and other components (such as the toolbar and the Open Agent dialog) employ the SEM to provide Agent creation, deletion, browsing, updating, and management of Agents via its user interface.

[0725] Preferences Manager. This component manages all client-side preferences, providing services to persist the preferences, communicates with servers as needed to share preferences or support roaming, and supports setting and obtaining preference values from other components. This component has associated user interface as well as some more specific preferences user interface components. The preferences are divided into sub-components, and may abstract the preferences for associated client classes. These include:

[0726] Core Preferences. This includes basic configuration such as user profile and persona information.

[0727] Skin Preferences. This also associates preferred Skins with object classes, as well as the preferred list Skin and screen saver Skins. These may be additional Skin-related preferences settings.

This component also manages the set of locally available Skins. downloadable Skins are preferably managed through this component.

[0728] Notification Manager. Notifications provide a means to indicate to users that there is new information available on a given Smart Agent. Users optionally configure a specific Smart Agent to support or provide notifications (it will be OFF by default for most Smart Agents), and will also configure how to present notifications to users. These notifications are presented by the Notification user interface component.

[0729] The Notification Manager is responsible for managing background, polling queries for the appropriate set of Smart Agents. The Live Information Manager is a parallel component that provides similar services to the Results Browser.

[0730] The Notification Manager gathers the list of Smart Agents marked for notification, and periodically polls the associated servers for new information. “New” is defined as “since the list poll [or query].” Each time the poll responds, it includes a timestamp indicator that the Notification Manager must persist, associated with the Agent.

[0731] The user interface associated with configuring the Notification Manager is preferably implemented in coordination with the Agent Tree View. This enables notifications (e.g., a “Notify” popup menu option of each Smart Agent). The Notification Manager may also support alternatives for notifying the agent when there are new results available. Some options include a display style (e.g., bold, colored, etc.) for the Agent in the Agent Tree View, a reminder dialog, audio notification, or more exotic actions like email, IM or SMS notification.

[0732] Client-Side Security. Client-side security issues relate to extension code and Skins. The Skins are preferably XSLT, but may also support script. In addition, the generated HTML may include references to ActiveX components and behaviors. The presentation sandbox may include security restrictions that prevent Skins from running potentially malicious code via script. For example, the implementation may completely disallow any unsigned code (including ActiveX and DHTML behaviors).

[0733] All client-server communication with Agencies are preferably hidden from the published interfaces (for Skins), which third parties will customize to provide custom Skins. By isolating the functionality outside of the primary client runtime, the risk of security compromise can be reduced.

[0734] Core User Interface Components

[0735] Agent Tree View. This is a Shell Extension Tree View that supports much of the core user interface for controlling and invoking Agents.

[0736] Semantic Environment Browsing User Interface. This provides user interface to allow users to browse the Semantic Environment. An example of this is the “Open Agent Dialog.” This complements the Agent Tree View, which also displays a hierarchical view of the workspace (see screenshots).

[0737] Agent Inspector. This provides user interface to view the properties or edit (in the case of user-created Smart Agents) an individual Agent, Blender or Agency.

[0738] Browser Host. This is preferably a “wrapper” on the semantic browser core (e.g., the Internet Explorer browser runtime), which allows the presentation of a custom view of the Agents, Agencies, and Blenders in the Agent Tree View. It preferably does not have any user interface itself, but can be a bridge component between the Shell Extension and the Browser Framework. This component is also preferably responsible for coordinating certain browser functionality with the Windows Shell user interface, including in particular the navigation (“back/forward”) mechanism, in order to provide a seamless “back/forward” user experience (wherein the user only has to deal with one “back/forward” history list).

[0739] Core Preferences UI. This provides a user interface for preferences related to Semantic Environment, server, persona and Agent management, as well as any other miscellaneous preference settings. This preferably includes primitive property sheet dialog, possibly divided up into separate sheets.
by functional area. In the preferred embodiment, this should be a tabbed dialog user interface.

[0740] Skin Preferences UI. This provides a user interface for preferences related to Skin management. This is preferably a property sheet dialog. The list of available Skins should be presented as a list, for selection. This user interface allows users to set the current Skins, as distinct from the default Skins it preferably allows users to make the current Skin be the default. For per-Agent Skin preferences, this preferably allows users to select a Skin for the currently selected or opened Agent.

[0741] Notification UI. The user interface associated with configuring the Notification Manager is preferably implemented in coordination with the Agent Tree View. The Notification Manager may also support alternatives for notifying users when there are new results available. Some options include a display style (e.g., bold, colored, etc.) for the Agent in the Agent Tree View, a reminder dialog, audio notification, or more exotic actions like email, IM or SMS notification. In the preferred embodiment, the user interface should include a tabbed dialog (or equivalent) to allow users to select out of the aforementioned notification schemes (and the like).

[0742] Screen Saver. The user interface preferably provides a special modality to the Results Browser that function like a screen saver, filling the screen in theater-mode display. In the preferred embodiment, special Skins should be used for the screen-saver mode. These Skins could emphasize a dynamic display that can leverage a larger screen area, but could also use larger fonts and more widely spaced layout.

[0743] Browser Framework

[0744] Results Browser. The Results Browser is responsible for displaying the results of queries, and the information on any local resources opened. The Results Browser preferably obtains one or more XML files from the Query Manager and merges these into a single XML file that represents a list of objects. The list itself may be filtered or sorted as an initial step. The list as a structure is transformed by a special class of Skin (an XSLT transform sheet, possibly including some script) that handles lists. The list-Skin creates the primary HTML (or the like) structure, e.g., a list, a table, or perhaps a timed sequence. Object Skins manage the individual DHTML items that present the information for each object instance. List-Skins may handle the dispatch of individual object Skins (mapping object class to Skin), but the Results Browser preferably provides default mappings of class to Skin for simplicity.

[0745] Users may prefer a given form of presentation, and may choose default Skins (both for the list as well as for object classes). The original query (i.e., the SQML) may also include parameters that indicate which Skins should be used (especially which list-Skin) These will be passed to the Results Browser along with the results. The Results Browser uses the facilities of the Skin Manager to select the right Skin to apply. Different rules may be employed for how user preferences and Agent (author) preferences are combined and prioritized.

[0746] When query results are composed of multiple distinct XML files, the Results Browser must merge these into a single XML document to provide a seamless user experience. The preferred embodiment provides for handling additional results dynamically. This dynamic update mode is preferably implemented by using a different template or perhaps a script method within the XSLT template. Alternatively, the list Skins may require a behavior (or local runtime component) to manage the logic of adding to the document without disturbing user context.

[0747] Query Manager (or Client-Side Semantic Query Processor). The Query Manager is responsible for handling the communication with the server(s), executing the requests for information and gathering the XML results. The resulting XML is passed to the Results Browser for presentation to users.

[0748] The Query Manager preferably provides the services to support the Smart Lens functionality. When a Smart Lens request is made, the results are returned as XML and are passed to the Results Browser, preferably marked to indicate that they are Smart Lens results for a given object. The Query Manager preferably includes the following sub-components that provide individual services to fulfill the query requests.

[0749] SQML Interpreter. This component must decompose passed SQML into a set of requests, possibly with linked resources. Each request or resource link resolves to a resource with an associated protocol (e.g., HTTP, or one of a number of local pseudoprotoocols like outlook): and is dispatched to the associated protocol handler. A given SQML file may include a mix of network and local resource types.

[0750] Resource Handler Manager. This is preferably a central registration mechanism for resource handlers. It is a minimal layer that associates protocols and pseudoprotoocols with handlers, and simplifies the dispatch of resource requests.

[0751] Resource Handlers. These are components that encapsulate the specifics of accessing the resources from a given "server." A resource handler does not resolve any linked resources. This is preferably the responsibility of the SQML Interpreter (i.e., the SQML Interpreter will have already resolved linked resources and provided the associated meta data as part of the resource request to this handler). When the resource is a Semantic Web service, the component preferably bundles up the request and issues it via http. When the resource is a local resource (e.g., a document: or Outlook: resource), the resource handler handles the resource directly. For documents, the resource handler passes the document (a file: URL) to the semantic meaning extraction, summarization, and categorization engine to extract meta-data. For email, the resource handler extracts messages from the exchange server, or local .PST files. Note that when there are links on a local resource, the local resource handler must perform the processing that filters results for semantic relatedness. This may be custom to the handler for efficiency, but a central, generic Relatedness Engine will provide services for most cases.

[0752] Relatedness Engine. This provides a place to gather the logic for comparing objects for relatedness. The comparison is preferably dependent on the mix of schemas involved, but is otherwise a simple operation—given two objects, provide a measure of relatedness.

[0753] Filter/Sort Manager. The Filter/Sort Manager supports the application of filters and sorts to the lists of results provided to the Results Browser. The Filter/Sort Manager leverages the services of the Filter/Sort Preferences component to obtain user preferences for current settings. The main function of this component is to resolve general preferences, per-Agent preferences, and any settings defined in the actual results (this may or may not be supported). This component is
notified by the Filter/Sort Preferences component when users change the currently applied filters and sorts. Because the associated user interface is part of a tool bar associated with the Shell Extension (i.e., its right-pane View), but the application of the functions happens in the Results Browser space, the control is typically indirect.

0754] Lens Mode. When a Smart Lens is invoked, the Results Browser must generate Lens requests (queries) for objects that users choose. The queries are asynchronous so that users can select Smart Lens queries for various objects and view the results as they are returned. A suggested user interface for this is to reserve some real-estate for a Smart Lens icon. When in Smart Lens mode and the user clicks (or hovers) over the Smart Lens icon, a query is issued, and the icon changes to indicate that the query is in progress. When results are returned, they are handled by the Results Browser and dedicated Smart Lens templates in the Skins, and the Smart Lens icon for an object changes to indicate that results are available. Clicking or hovering over the icon again will display the Smart Lens results in a Skin specific manner (see sample Smart Lens pane user interface). If the query is returned quickly enough, then the whole function preferably feels like a popup activated by a hover or simple click.

0755] Deep Info View. If Deep Information is not available in the original results, this component generates the associated query. The query is preferably asynchronous. When results are returned to the Results Browser, they are processed through the appropriate Skin (using a special Deep Information template for each Skin), and the resulting HTML is incorporated into the results document under the associated object. The primary Skin for the schema inserts a Deep Information element in the HTML for the object so that the Results Browser knows where to incorporate the results. When Deep Information is available (whether as part of the original results or in response to a Deep Information query), the Skin either displays it directly or will indicate that it is present, and some Skin-defined user interface will allow users to enable the display (e.g., as a popup window).

0756] Context Info Manager. For objects currently displayed in the Results Browser, certain notifications are preferably provided by default. Two classes of new or additional info will be provided to users:

0757] 1. Additional results that were added to the server since the user made the original request. This is especially useful for things such as headlines or active email threads. The results are handled by the Results Browser, by inserting the new objects into the view.

0758] 2. Context Templates and related information that would be of interest to the user. This is generated by additional queries to a specific Agent (Smart Agent, Special Agent, Blender or Agency), using a particular object as context. The results are handled similarly to the way that Deep Information View and Smart Lens Mode results are handled, by processing the XML returned from the query, and inserting the resulting HTML into the existing HTML for the object. The Skin controls the display mechanisms and UI. An example of related information is “Breaking News” associated with the object.

0759] Skin Manager. Maintain user preferences for list Skins, object Skins, and dependencies between list and object Skins (certain object Skins may only make sense for a given list-Skin) The Skin Manager also maintains parameters for each Skin that indicate constraints for the Skin, e.g., how much screen real-estate it requires, or modalities it best applies to. Considerable intelligence is preferably built in that assists the Results Browser to choose Skins for a range of screen and window size constraints, as well as for modalities, accessibility, language and other constraints. Initial versions will likely be much simpler.

0760] Skin Templates. This describes the structure of a Skin and how it is applied from within the Results Browser. A Skin is preferably XSLT templates that convert the results XML to XHTML (and/or other languages like SVG) or proprietary presentation platforms like Flash MX and ActionScript. The templates can also insert styling information, e.g., for CSS styling. The resulting presentation code (e.g., XHTML) can restrict the inclusion of code, for security reasons. Framework code in the Results Browser invokes the Skins. The preferred embodiment includes the following classes of Skins.

0761] List Skins (or layout Skins). A list Skin is used to transform a list of objects returned from a query into some overall presentation structure. This may be a simple list, a table, or a timed sequence of slides. List Skins are not schema or object specific, although they may only support certain Skins, which can work within the constraints that the associated presentation form defines. E.g., a list Skin that defines a table layout may require, or prefer, object Skins that can produce information in a small rectangular format.

0762] Object Skins. Object Skins are schema specific, and generate the presentation for an individual object of a given information object type (or information class). It is possible to define a Skin for the generic super-class (or any other super-class) that can serve as a default Skin for a range of derived classes or subclasses (presumably by omitting some details).

0763] Context Skins. Context Skins are tied to a particular Context Template, and generate the presentation that will most effectively convey the context indicated by the template.

0764] Blender Skins. Blender Skins are designed to present the results from Blenders. These Skins should allow the user to view the results via the Agents contained in the Blender, via information object type, or via a merged view that displays all the results as though they came from one source.

0765] Skins preferably model constraints such as modality and presentation display area by handling the constraints (passed as parameters either statically or dynamically by events within the browser core itself). This is preferably supported by imposing a restriction that list Skins must specify only acceptable object Skins. In an alternative approach, object Skins may be designed for a given list Skin, and the Results Browser/Skin Manager chooses object Skins for the current list Skin.

0766] List Skin Details. Users may choose a single list Skin for the current view and make it the default. List Skins may also be associated with individual Agents, in which case the generic default is overridden. The Results Browser invokes the list Skin to process the list of results, although the list Skin preferably does not actually handle the individual objects. It creates some per-object instance in the framework presentation (e.g., a timed entry in a sequence, or a cell in a table, or an item in a list), and then the object Skins will fill in the details.

0767] Object Skin Details. The object Skins convert a particular schema to XHTML. Support for asynchronous query results for things like Deep Information and Context Template information are provided by invoking associated templates from the Results Browser (through the DOM) on the
query results XML, and then inserting the resulting XHTML into the results document through DOM interfaces. There are preferably several individual templates within an object Skin, including:

[0768] Primary schema template. This is the main piece that generates XHTML, for default display. This must create the wrappers for Deep Information, Smart Lens information, Context Template information content, and any script that provides user control over the associated display.

[0769] Deep Information template. This template handles the meta-information for Deep

[0770] Information. It may be called for inline deep info provided with original results, or it may be called to handle asynchronously requested Deep Information. Either way, it preferably generates XHTML in some form, which is inserted under the wrapper element for Deep Information. The insertion probably happens in XSLT for inline deep info, and is effected through DOM insertion for Deep Information query results.

[0771] Context information template. This template handles the results-information for context information query results. It generates XHTML in some form, which is inserted under the wrapper element for live info. The insertion is effected through DOM insertion for Deep Information query results.

[0772] Smart Lens information template. This template handles the results-information for Smart Lens query results. It generates XHTML in some form, which is inserted under the wrapper element for live info. The insertion is effected through DOM insertion for Deep Information query results.

[0773] In the preferred embodiment, the template cannot modify the other contents of the XHTML (even for the same object), so it will be up to the Results Browser to coordinate the user interface changes that indicate when Deep Information, live information or Smart Lens results are available. The framework requires certain icons to be used (also for consistency), and for these to have regular names or element types, which will allow the Results Browser to find and modify them as needed. In addition, the Results Browser can create and raise events to indicate the state changes. The template-generated script can respond to these events, and display the associated information as desired.

[0774] Default Skins. In the preferred embodiment, a set of default Skins is provided. This preferably includes Skins for the basic object classes and a small set of list-Skins that allow a variety of views of query results. Preferable list-Skins include:

[0775] A detailed list display (like the Windows Explorer details view)

[0776] A tabular icon view (again, like the Windows Explorer icon view, but somewhat richer)


[0778] e. Client Framework

[0779] In the preferred embodiment, the system client includes Shell Extensions, a Presenter, and Skins used by the Presenter to display information with context and meaning.

[0780] Shell Extension. An Explorer Shell Extension is a Microsoft Windows software component that extends the Windows Shell with custom code. Shell Extensions allow applications to use the Shell as a custom client, and also provide services such as clean integration with the desktop, the file-system, Internet Explorer, etc. Examples of default shell extensions include “My Documents,” “My Computer,” “My Network Places,” “Recycle Bin,” and “Internet Explorer.”

[0781] The use of a Shell Extension in the preferred embodiment of the present invention has several advantages:

[0782] 1. It provides a very clean way to provide a user experience that seamlessly integrates with how knowledge workers currently browse for information. In turn, this obviates the need to develop a proprietary client and allows for non-standard integration with Microsoft’s Internet Explorer, “My Documents,” etc.

[0783] 2. It embraces Today’s Web and provides a migration path for the transfer of content in Today’s Web to the Information Nervous System of the present invention. For example, users preferable drag and drop documents from their hard drive (via Microsoft Explorer) or from the Internet (via Internet Explorer) into remote Agents on the Shell Extension of the present invention. This is difficult and non-intuitive with a proprietary client. Nevertheless, the present invention contemplates portability to a proprietary client or to the equivalent of Shell Extension on non-Windows operating system and operating systems for non-personal computer devices.

[0784] The Shell Extensions of the present invention provide a view of users’ Semantic Environment (e.g., history, favorites and other views). In the preferred embodiment, the Shell Extension provides for the following:

[0785] 1. Allows users to open an Agent, a document, a folder, or an address on the semantic browser’s Semantic Environment. For an Agent, the client displays a custom “Open Agent” dialog box that allows users to browse the semantic browser’s Semantic Environment. This preferably includes Agents in users’ My Agents list, Agencies on the Global Agency Directory, Agencies on the local area network (announcing via multicast), and Agencies on any custom Agency Directory that users have configured. [INSERT RELEVANT SCREEN SHOTS ON UI]

Opening an Agent results in the client displaying the results of the query of that Agent. Opening a document opens the XML metadata for that document, consistent with the schema for the document object type. Opening a folder opens the XML metadata for a file-system folder. Users are able to open the immediate or deep contents of the folder via the folder itself. Opening an address allows users to enter any address to be opened by the client framework. This includes URLs (which open the XML metadata for the document), documents on the file-system, Agents, or objects (see “URL Naming Conventions” below). In the case of Agents, the Agent URL is preferably entered as follows: Agent://<Agent name>@<Agency name>. This is analogous to the http://<URL> naming convention for HTTP URLs. The Agent:// prefix is required in this case because the Open Address option can open any address. In the case of the “Open Agent” option, users preferably do not need to add the prefix; the client framework automatically canonicalizes the URL to include the prefix. This is similar to how users are able to enter “www. foo.com” into Today’s browser without the qualifying http:// prefix.

[0786] It is anticipated that the client allows users the ability to open other objects, for example, Microsoft Outlook .PST files.
2. Allows users to browse, subscribe, and unsubscribe to or from Agents on a given Agency that supports User State.

3. Allows users to save invoked Agents or semantic query results into the My Agents list.

4. Allows users to create Blenders and to add and remove Agents to and from Blenders (including via drag and drop).

5. Notifies users when there are new Agencies on any of the Agency directories (for example, the Global Agency Directory, the Local Area Multicast Network or any custom Agency Directories) since the last time they checked.

6. Notifies users when there are any new Agents on any particular Agency since the last time they checked.

7. Provides drag and drop access to relational semantic queries for objects in the Semantic Environment. The Shell Extension allows users to drag and drop a document from the Semantic Environment (either on a local drive, the network neighborhood, the Intranet, or the Internet) to a shell folder representing an Agent. This triggers a remote procedure call to the XML Web Service for the given Agency with the document metadata as the argument.

8. Provides “paste” access to objects copied to the system clipboard. The present invention uses the system clipboard to allow users to copy any object for later access. In addition, the clipboard allows users to copy objects from other applications, for example, Microsoft Office applications (e.g., email items from Outlook), from multimedia applications, and to copy data from any application.

9. Allows users to select an Agent as a Smart Lens. A Smart Lens allows users to view objects in the results view based on context from an Agent or any object that can be copied to the system clipboard. For example, ordinarily, if a document object is in the results view and users hover over the link representing the object, the object metadata is displayed. If, however, a Smart Lens is selected for example by pasting it onto the results sheet, and users hover over the object, information that relates the object in the Smart Lens and the object underneath the cursor is displayed. For example, if users copy “People:Research:All” to the clipboard and paste it as a Smart Lens, then hover over a document, metadata may be displayed in a balloon popup as follows: “Found 15 people in PeopleResearch:All that are experts on this document.” Other examples are “Found 3 people that might have written this document” and “Found 78 email messages relating to this object posted by people in People:Research:All”. Users decide whether to invoke any of the links in the metadata in the balloon popup. In an alternative embodiment, the popup may be displayed in a sidebar and does not require a balloon. When a Smart Lens is pasted onto the clipboard, the Shell Extension preferably communicates with the system and changes the mouse cursor to reflect the name of the selected Agent. The Smart Lens preferably has global scope because it is copied from the clipboard. In other words, for example, all instances of Windows Explorer and Internet Explorer “see” the Smart Lens and respond to its actions. In the preferred embodiment there is a Smart Lens tool in the Information Agent toolbar that applies to the current object on the clipboard (e.g., Agent or other object). By default the Smart Lens tool will be deselected once a link is clicked in the system. Users are preferable able to “pin” the Smart Lens. When the Smart Lens is pinned, the Smart Lens remains active until users explicitly deselect it. In the preferred embodiment, to pin a Smart Lens, users select the “Paste as Smart Lens and Pin” tool on the toolbar.

10. Allows users to “tear-off” the results of an Agent from the Shell Extension and display it in docked view on the desktop. In this view, the Agent results browser window acts as a semantic ticker. This feature allows users to continuously display the semantic information while continuing to do other work.

11. Allows users to enable an Agent to be used as a screen-saver.


Presenter. The Presenter is a set of local components (e.g., browser plug-ins) that take semantic queries from scripts (or other plug-ins) and pass them off to a KIS Agency XML Web Service. The present invention translates the results of semantic queries and passes XML to other behaviors or scripts for eventual presentation to users.

In the preferred embodiment, the Presenter is invoked by the Shell Extension with an SQML file. The system preferably communicates with the XML Web Service directly. The system resolves the SQML file and invokes calls to open XML information sourced locally or remotely (via XML Web Services on Agencies referred to in the SQML file). Alternatively, if an Agent URL is passed to the system, the Presenter directly opens the URL by invoking it via a call to the XML Web Service of the Agency on which the Agent is hosted. In the preferred embodiment, the system calls the appropriate method with the appropriate semantic object type. Examples of default semantic object types are "SEMANTICOBJECTTYPEID_EVENT", "SEMANTICOBJECTTYPEID_EMAIL.MESEAGE", etc., which are defined in the header file (semantic.runtime.h). The preferred embodiment allows registration of new semantic object types via the RegisterSemanticObjectType API. This semantic query processor on the Agency returns the appropriate XML results using the semantic object type as a filter.

In the preferred embodiment, a Skin according to the present invention (see below) uses XSLT (and/or script) to transform the XML returned from the framework (en-route from the XML Web Service) into DIHTML. The Shell Extension allows users to select a new Skin for the current query.

Skins are preferably object-type specific. Context Template specific (for Special Agents) or Blender specific (for Blenders) Skins can also be customized based on the semantic domain name/path or ontology of the Agent, and based on other attributes such as the user’s persona, condition, location, etc. Each Agent is configured on an Agency with a default Skin. The present invention further contemplates custom Skins that may be published onto the root Agency (e.g., on the Global Agency Directory). The client preferably downloads the Skin either from the Agency for the declared Agent or from a central server (e.g., the Global Agency Directory), and applies it to the current presentation. The client optionally includes user preferences to ignore Agent Skins or to confine them to a portion of the user interface.
Aside from the Skin type (e.g., object Skin, list/layout Skin, Context Skin, Blender Skin, etc.), in the preferred embodiment, Skins are categorized as follows:

Design template Skins
Color template Skins
Animation template Skins
Semantic Skins are preferably required to be interactive, except when they are displayed as part of a tear-off (see above) or screenshot. Each Skin allows users to seek to a particular point in the "semantic presentation." For example, if the Skin initially displays only the first 25 items, the Skin must have a seek-bar (or other user interface mechanism) to allow the user to seek to the next 25 items, to fast-forward, to rewind, etc. Some Skins have a "Real-Time Mode" option. In this mode, the Skin continuously fetches new objects from the XML Web Service (via pull). Skins are responsible for polling the XML Web Service for new information based on the schema of the desired objects. In the preferred embodiment, there are no notifications to the client since the Agency does not maintain any client-specific state for scalability reasons.

Skins optionally include a real-time mode. These Skins are required to be intelligent in that they must cycle through (i.e., present, order or highlight) objects based on priority. For example, if the Presenter relays information indicating that a new object is posted on the Agency, the Skin immediately displays/headers/highlights this and continues the presentation of the remaining objects. The Presenter determines the ordering and the Skin deals with dynamism given various sort and filter settings. This creates the perception that the semantic presentation is occurring in real-time. In the preferred embodiment, this occurs when there is new data that users are allowed to access using Skins. If the list is time-sorted, the real-time presentation may confuse users due to jumping the user interface into an interactive mode. A user preference option in some modes (e.g., screen saver mode) automatically resets the Skin to display the new data (e.g., scrolling to the top of a sorted list when new data is inserted at the top of the list).

In an alternative embodiment, Skins are designed to customize their presentation based on the amount of available presentation window. For example, a Skin may change from static mode to dynamic mode by displaying information using fade-in and fade-out if, for example, the presentation window is relatively small. Skins are preferably modal depending upon the desired level of user interaction. For example, a screen saver works differently from a browser; a docked view is similarly different (not only because it is smaller, but because it is assumed to be a kind of background view rather than a focus of user interaction). Whea a view is minimized or hidden, an alternate mode may be used (especially to indicate new information). Examples are audio notification, reminder-like alerts, start-bar show and blink (like outlook reminders). Agents may be used to send email, telephony or Instant Messenger (IM) notifications. In an alternative embodiment, the present invention contemplates an Agent that posts to a Web site (e.g., automatic HTML content generation for event calendars).

Alternatively, Skins may generate audio-visual information. For example, a text-to-speech Skin may read out an email object. This feature has great potential value for disabled users and for users of auto-PCs, etc., as well as other uses.

In the preferred embodiment, the Skins framework exposes the following services:

1. Methods to open an SQML-based semantic query. This can be a local SQML document, an Agent, etc.
2. Methods to open an Agent URL directly.
3. Methods to browse the Information Agent Semantic Environment.
4. Methods to interface with the system clipboard using customizable clipboard formats.
5. Methods to persist the current Skin for a given query or for a given semantic class ID.

Skins. As introduced above, Skins are presentation templates that are used to customize users experience on a per-Agent basis. In the preferred embodiment, Skins are XSLT templates and/or scripts that are hosted on a centralized server Skins according to the present invention preferably generate XHTMLEM+TIME code (e.g., for Presenter display, text-to-speech, Structured Vector Graphics (SVG) via a plug-in, etc.) and access various system services. In the preferred embodiment, Skins support the following features:

1. Display some or all of the fields corresponding to the XML schema of the object(s) being displayed. The Skin optionally provides users a way to uniquely distinguish objects in a returned set or provides users with any conventional access means, for example, file name, URL or personal name (for people).
2. Display a user interface indicating whether the object is understood by the host Agency. Each object preferably includes an "understood" field that indicates this information.
3. For the semantic object type SEMANTICOBJECTTYPE_OBJECT, the Skin optionally displays the raw object metadata or displays the metadata for the XML schema for the class-specific objects that the raw objects represent. For Skins that display class-specific XML schema for queries that refer to raw objects, the Skins must be "smart" to display the class-specific information in different panes. Preferred ways of accomplishing this uses frames, tabbed boxes, or other user interface techniques. Since every semantic query points to raw objects, the Skin preferably either loads the query with the filter SEMANTICOBJECTTYPE_OBJECT (which simply returns raw objects) or the required object type ID. In the preferred embodiment, in order to prepare the presentation of an object list with raw objects of many classes, the Skin should first:
   - Get the object query
   - For each semantic object type, determine how many objects exist in the Agent resource for the given object type. This is preferably obtained by calling the Agent XML Web Service method GetNumObjectsOfClass(InAgent with the Agent URL and the object type ID name (email, document, event, etc.) as argument. The XML Web Service returns the number of objects in the Agent, satisfying the object type ID filter.
   - Depending on how many object types there are in the Agent query, the Skin displays frames or other user interface that are appropriate for the number of object types. In the preferred embodiment, when the Skin is ready to load the object type-specific metadata, it calls the Agency's XML Web Service method ExecuteSemanticQuery with the Agent URL and the semantic object type as the arguments
   - When users hover over an object, more metadata for the object is displayable.
[0824] 5. If a Smart Agent Smart Lens is selected, the Information Agent of the present invention displays contextual metadata that maps the object in the Smart Lens with the object underneath the mouse. In one embodiment, the Smart Lens applies to objects displayed within the Presenter. In alternative embodiment, the present invention allows the Smart Lens to be invoked in other applications (e.g., Microsoft Office applications, the desktop, etc.). This involves installing system hooks to track the mouse and invoke a Smart Lens application when the mouse moves anywhere in the system. The “hook” is called on all mouse events and the hook will also capture the mouse. The Smart Lens may alternatively be invoked asynchronously. In this embodiment, anytime the Presenter displays new results, it checks the clipboard to see if there is any semantic Smart Lens information present. In the asynchronous embodiment, the Presenter automatically caches all the Smart Lens results for all objects in its view. It displays an icon beside each object it presents indicating that there is context-specific related information therein. In a preferred embodiment, users are able to invoke a Smart Lens for any object in the view.

[0825] 6. Breaking Information. Each object preferably displays a user interface indicating whether there is “breaking information” relating to the object. This is the semantic equivalent of “breaking news.” The user interface is preferably presented to indicate the criticality of the information, yet must not be too intrusive in case users do not want to see the information. For example, the user interface may be shown as an icon that slowly blinks at a corner of the object display window. When users hover over the icon, metadata on the “breaking information” is displayed. In the preferred embodiment, “breaking information” is implemented by an implicit Special Agent that invokes calls to all Agents using the Breaking News Context Template.

[0826] 7. Each object is preferably displayable with a user interface indicating whether the object has any Annotations. This information is included as a field in all query results for all objects.

[0827] 8. Preferably, each object is displayable with a user interface indicating whether there is related information on any predefined Context Template or Special Agent on the client. This preferably includes Special Agents created by users, as well as default Special Agents (e.g., installed by the client). In the preferred embodiment, Context Palettes for the Context Templates are displayed with the user having the option of displaying one or more of the Context Palettes, hiding them, scrolling them (in order to navigate the Context Palettes), etc. Context Templates and Context Palettes are discussed in further detail below. In an alternative embodiment, Agency priorities preferably include the following:

[0828] Critical priority. This is the highest priority. For example, for a given document, this flag will be TRUE (on the Agency) if a related email message was just posted (in this example with a few minutes) or if there is an upcoming event that is imminent.

[0829] High Priority. This is the next highest priority. The user interface feedback preferably makes it clear that the priority is high enough to warrant users’ attention, albeit the feedback must not be very intrusive. The priority is optionally different for different Users, e.g., if there is an event that is local to users the priority might be higher than if the event is remote (particularly if there is no way for the remote user to participate in the event).

[0830] Medium Priority. This may merely indicate that there is information that users should look at if they have the time. The user interface feedback must make this clear.

[0831] Low Priority. This may indicate that there is related information that is germane but not recent.

[0832] The four priority virtual Blenders are preferably installed by default on the client. These Blenders automatically aggregate information from corresponding priority Agents on each Agency in the My Agencies list. There is preferably default priority Agents on every Agency. In the preferred embodiment, relational semantic queries take the context and the user into consideration.

[0833] In the preferred embodiment for each Context Template (or the currently selected Context Template), the Presenter enumerates the Agencies that users add to their My Favorite Agencies list or the recent Agencies, and queries appropriate Agencies using dynamically generated SQML to find out if there are any objects that relate to the current object based on the Context Template. If any of the Agencies in the favorites or recent lists are not accessible, the user interface preferably transparently handles this by ignoring the Agency. In the preferred embodiment, by default, the dynamically generated SQML is created by indexing the SQML of the currently selected object’s SRML and inserting the resource in the SQML as a link filter in the SQML of the Context Template (preferably using the default predicate “relevant to”). This intelligently handles the mapping of the object type of the currently selected object to the semantics of the displayed Context Palette. For example, if the currently selected object is a document, the Newsheadlines Context Palette uses the SQML based on a derivation of the SQML for the Headlines Context Template. Each Agency in the Semantic Environment semantically processes the resulting SQML appropriately using the default predicate. In another example, if the selected object is a person, the Headlines Palette shows the Headlines relevant to the person, e.g., the “Headlines” authored or annotated by the person, etc. Alternatively, if the currently selected object is a document or email message, the SQML (with the default predicate) produces semantic results that represent semantically related Headlines on each Agency. These results are preferably displayed in the Context Palette. The same applies to other Context Palettes (e.g., Classics, Newsmakers, etc.).

[0834] For a person object, the priority flag preferably refers either to objects the person has posted to or objects the person authored or is hosting. In this example, only metadata fields with semantic uniqueness are preferably used to make this determination (e.g., the person’s email address).

[0835] 9. Each object preferably displays a user interface including a number of manipulation options. By way of example only, a sample user interface illustrating an information object displayed in the Information Agent (semantic browser) Results Pane is shown in FIG. 25. FIG. 25 shows a balloon popup (for the object’s metadata) and user interface icons on the object allowing the users to invoke tool options such as a Recommendations context pane, a Breaking News context pane, a verbs popup menu, etc. Additional and other user interface options include the following:

[0836] Intrinsic Semantic Links. These are links that are intrinsic to the semantic class of the object. If there are no Intrinsic Semantic Links, nothing needs to be displayed. By was of example, an email object of the preferred embodiment includes the following Intrinsic Semantic Links:
In the preferred embodiment, any of these semantic links are invoked by users, the client fetches the metadata for the associated object (and not the object itself). This allows users to explore the semantic information for aspects of the original object. The Skin preferably calls the XML Web Service of the Agency that hosts the object with the appropriate method. In the preferred embodiment, the form of this method is ISemanticRuntimeService::LoadNativeSemanticLink. This embodiment includes the semantic class ID, the name of the semantic link, the name of the argument, and the string form of the argument. For example, to "navigate" to the third attachment (with a zero-based index), the Skin should call LoadNativeSemanticLink(SEMANTIC_CLASS_EMAIL_MSG_ATTACHMENTS, "Attachments", "Index", 2). This preferably generates the SQML that represents this relational semantic query, creates a new temporary Smart Agent that has this SQML and loads the Smart Agent. This illustrates preferred semantic navigation. The process is optionally recursive. The user can navigate off the new results using any of the new objects and pivots, etc.

An example of a balloon popup associated with an Intrinsic Semantic Link showing an email sample according to the present invention is shown in FIG. 26. In this sample user interface, the popup menu is displayed when users selects the “Intrinsic Links” icon on an information object in the Results Pane. This illustration shows what Intrinsic Semantic Links users see for an email object. In the preferred embodiment, the popup menu items invoke a new SQML query (what the proper resource and predicate links) when users hit the menu option. A new temporary Agent is created (with the SQML) showing the results of the query. Users are able to save the Agent in their favorites list. Also, the new results display the Intrinsic Semantic Links Context, Templates, etc., thereby support user-controlled browsing where in users can navigate information semantically. An alternative configuration and functionality for native verbs follows:

[0837]

Annotations. This preferably allows users to navigate to a summary view for all the Annotations for the current object. In the preferred embodiment, the Skin displays all the Annotations by calling the ISemanticRuntimeService::EnumAnnotations (with the object metadata as argument). This returns an XML representation of the property table containing the metadata for the Annotation objects. The Skin preferably displays some representation of the Annotation summary being displayed (e.g., names or titles of the Annotations). When an Annotation link is invoked by users, the Skin displays metadata for the Annotation object. These functions preferably come from filters applied on the client. Alternatively, these functions can be created as an Agent. This aspect of the present invention further illustrates semantic navigation. The Annotations are preferably loaded using an SQML representation of the “Annotations” query. This creates a new Smart Agent with this SQML. The Smart Agent is then added to the “recent” list and loaded (or navigated to). The process is optionally recursive. The user can navigate using the newly displayed Annotation(s) as pivots, etc.

[0839]

Related Objects. In the preferred embodiment, this optionally allows users to find related information on each Agency included in the users’ My Agencies list using the current object as an Information Object Pivot. This is preferably accomplished without resorting to a copy and paste or reliance on the shelf extension user interface. In the preferred embodiment, the user interface popup shows information in the following format:
-continued

Agencies that understand this object →
Agency Bar →
All, All
All, Understood, All
All, Critical Priority, All
All, High Priority, All
All, Medium Priority, All
All, Low Priority, All
All, My Favorites, All
All, Recommended, All

[0841] The “All my agencies” list is obtained by the Presenter simply by enumerating the Agencies users have registered locally. The Presenter returns the “Agencies that understand this object” list by “asking” each locally registered Agency whether it understands the object in question. The Presenter passes the XML representation of the object to the Agency, which attempts to semantically process the XML representation. The Agency returns a flag indicating whether it understands the object. The Presenter optimizes the returned list by excluding the Agency on which the object itself is hosted since each object has a field that indicates whether the Agency understands its contents.

[0842] Verbs. This allows users to invoke any actions that relate directly to the current object. For example, a document or an email message can have an “Open” verb. This opens the word processor or email client and displays the information. An event can have an “Add to Outlook Calendar” verb. In the preferred embodiment, verbs, preferably class-specific, are invoked on the client by the system framework. The Agency need know nothing about verbs. In the preferred embodiment of the present invention, there are several verbs for every object. These verbs are preferably displayed first in the popup menu. In the preferred embodiment the verbs include:

[0843] 1. Annotate. When the user invokes this verb, the Skin preferably communicates with the client runtime and calls the Annotate method. This method initiates the default mail client with the appropriate subject line (which the Agency parses to interpret the Annotate). Users send a regular email message as an Annotation for the object. Email Annotations optionally include attachments that also constitute semantic links. This allows users to navigate from an object (e.g., a document) to its Annotation to its attachment and then to an external content source (e.g., via a Smart Lens). Alternative embodiments are also supported for Annotations, e.g., simple form-based or dialog-based annotations. But email provides the most semantic richness.

[0844] 2. Copy. This copies the object XML to the system clipboard.

[0845] 3. Hide. This indicates that users have no interest in viewing the object.

[0846] 4. Open. This is qualified with the link of what is being opened. In the example of a document, “Open Document” may be displayed. For an email message, “Open Email” may be displayed. The client opens the object with the default application registered in the system for the link’s MIME type. In an alternative embodiment, the present invention supports other related open verb form, such as “Open with . . .”, which allows users to open the object with a specific application.

[0847] 5. Mark as Favorite. This is preferably displayed if the Agency supports User State and if the object is not a favorite.
contexts Results Pane. In this sample, users have the option of selecting a template for the Deep Information that filters what kind of Deep Information to display, of viewing the "stories" of the Deep Information, along with semantic (SQML) links to objects that are in the Semantic Environment (for example, the "Steve Judd" person object, the "experts" Context Template results objects, the "direct reports" objects using the "direct reports" predicate filter), etc. In addition, users have the option of previewing the results of the semantic queries in-place using the Preview Player/Control.

[0866] e. Semantic Query Document

[0867] From the client's perspective, everything it understands is a query document. In the present invention, the client opens "query documents" in a way analogous to how a word processor opens "textual and compound documents." The client is primarily responsible for processing a Semantic Query Document and rendering the results. A Semantic Query Document is preferably expressed and stored in form of the Semantic Query Markup Language (SQML). This is akin to an "semantic file format." In the preferred embodiment, the SQML semantic file format consists of the following:

[0868] Head. The head tag includes tags that describe the document.

[0869] Head: Title—This indicates the title of the document.

[0870] Filters. The Presenter filters all returned objects using the entries in the "filter" tag. These entries optionally contain object type names (documents, events, email, etc.) If no filters are specified, no objects are filtered. The tag has a qualifier that indicates whether the entries are to be included or excluded. In the event of redundant entries (indicated with both "include" and "exclude" tags), the interpreter excludes the entries (i.e., in the event of a tie, "exclude" is presumed).

[0871] Attributes. This tag indicates the attributes of the document.

[0872] Skins. This is the parent tag for all Skin-related entries

[0873] skin:<objecttypename>. This contains information for the Skin to manage objects of the object type indicated in "object type name." The Presenter uses default and Agent Skins for objects that do not have corresponding Skin entries in the SQML document. Options preferably include the following:

[0874] skin:<objecttypename>:color. This has information on the color template to be used with this document. The primary entry is an XSLT URL.

[0875] skin:<objecttypename>:design. This has information on the design template to be used with this document. The primary entry is an XSLT URL.

[0876] skin:<objecttypename>:animation. This has information on the animation template to be used with this document. The primary entry is an XSLT URL.

[0877] Query. This is the parent tag for all the main query entries of the query document, and may include:

[0878] Resource. The reference to the resource being queried. Examples include file paths, URLs, cache entry identifiers, etc. These will be mapped to actual resource manager components by the interpreter.

[0879] resource: type. The type of resource reference, qualified with the namespace. Examples of defined resource reference types are: nervana:url (this indicates that the resource reference is a well-formed standard Internet URL, or a custom URL like "agent://...") and nervana:filepath (this indicates that the resource reference is a path to a file or directory on the file-system).

[0880] resource:arg. This indicates an optional string which will be passed to the resource when the interpreter converts the resource references to actual resources. It is the equivalent of a command line argument to an executable file. Note that some resources might interpret the arguments as part of the ref, and not as part of the ref argument. For example, standard URLs can pass the ref argument at the end of the URL itself (prefixed with the "#" tag)

[0881] resource:version. See below

[0882] resource:link. All link tags.

[0883] resource:link: predicate. This indicates the type of predicate for the link. For example, the predicate nervana:relevant indicates that the query is "return all objects from the resource R that relate to the object O," where R and O and the specified resource and object, respectively. Other examples of predicates include nervana:reportsto, nervana:teammateof, nervana:from, nervana:to, nervana:cc, nervana:bcc, nervana:attachedto, nervana:sentby, nervana:sentto, nervana:postedto, nervana:contains, etc.

[0884] resource:link. This indicates the reference to the object of the semantic link.

[0885] resource:link: type. This indicates the type of object reference indicates in the "ord" tag. Examples include standard XML types including xml: string, xml:integer; custom types including nervana: datetimeref (which may refer to object references like "today" and "tomorrow"), and any standard Internet URL (HTTP, FTP, etc.) or system URL (objects://, etc.) that refer to an object that the present invention can process as a semantic XML object.

[0886] resource:link: version. This indicates the version of the resource semantic link. This allows the Agency's semantic query processor to return results that are versioned. For example, one version of the semantic browser can use V1 of a query, and another version can use V2. This allows the Agency to provide backwards compatibility both at the resource level (e.g., for Agents) and at the link level.

[0887] Query Return Type. This indicates the type of query (or Agent) this SQML buffer file represents. In the preferred embodiment, this includes Agents, Agencies, Special Agents and Blenders.

[0888] Query Return Type. This indicates the type of objects the query returns (e.g., documents, email, Headlines, Classics, etc.). Alternatively, this may indicate names of information object types, Context Templates, etc.

[0889] By way of example, SAMPLE B of the Appendix hereto illustrates a Semantic Query Document in accordance with the present invention.

[0890] In the preferred embodiment, the Presenter includes an SQML interpreter. When the Presenter opens an SQML file, it preferably interprets it by first parsing it, validating it, creating a master entry table, and then executing the entries in the entry table. Effectively, it "compiles" the SQML file before "executing" it, not unlike how a language compiler compiles source code into an object module before it is then linked with other modules and executed. In the case of the
SQML interpreter, this process optionally involves loading other SQML files via references. This process is preferably not cyclical. The client uses the XSLT templates in the "<skin/>" tags (if available and if not overridden by default or Agent Skins) to display the information for each declared object type. Any returned objects that do not have a declared Skin are displayed with the default Skin of the object type or, in the case of a single Agent entry, that of the Agent (if one is specified).

[0891] In an alternative embodiment, the client may load a new Skin to display each object type even after the Semantic Query Document is opened. In this embodiment, the "<skin/>" tags preferably inform the client which Skin to load the query with initially. In this embodiment, the specified Skin is preferably appropriate for the declared object type.

[0892] In the preferred embodiment, the framework executes the document in two phases: the validation phase and the execution phase. For the validation phase, the interpreter first builds a master semantic entry table. The table is keyed with the resource URL and also has columns for the operator, the resource, the resource type, the predicate, the predicate type, and the link. The interpreter excludes all redundant entries as it adds entries into the table. Also, interpreter preferably canonicalizes all URLs before it adds them into the table. For example, the URLs "http://www.abc corp.com" and "www.abc corp.com!" are interpreted as being identical since they both share the same canonical form. The interpreter builds and maintains a separate SQML reference table. This table includes the canonical path to the SQML file. When the interpreter loads the original SQML file, it adds the canonical file path to the reference table. If the SQML file points to itself, the interpreter ignores the entry or returns an error. If the SQML file points to another SQML resource, it adds the new file to the reference table. It then recursively loads the new resource and the process repeats itself. During the process, the interpreter comes across an SQML entry that is already in the reference table, the interpreter returns an error to the calling application (indicating that there is a recursive loop in the SQML document). As the interpreter finds more resources in the document graph path, it adds them to the master entry table for the given resource. It dynamically adds links for a given resource to that resource’s entry in the entry table. As a result, the interpreter effectively flattens out the document link graph for each resource in the graph.

[0893] The interpreter then proceeds to the execution phase. In this phase, the interpreter reviews the semantic entry table and executes all the resources recursively asynchroneously, or in a sequential fashion. Next, it processes each resource based on the resource type. For example, for file resources, it opens the property metadata for the file and displays the metadata. For HTTP resources that refer to understood types (e.g., documents), the interpreter downloads the URL, extracts it, and displays it. For Agent resources, it calls the XML Web Service for each Agent and passes the links as XML arguments, qualifying each link with the operator. In the preferred embodiment, operators for links that cross document boundaries are always AND. In other words, the interpreter will AND all links for identical resources that are not declared together because recursive queries are assumed to be filters. The interpreter issues as many calls to a component representing the resource as there are Agent resources. For each link, the interpreter resolves the link by converting it into a query suitable for processing by the resource. For example, an Agent with a link with the attributes:

```xml
<predicate>nerva:relevance</predicate>
<core>c:/foo.doc</core>
<coretype>nerva:filepath</coretype>
```

is resolved by extracting the XML metadata of the object (e.g., c:/foo.doc) and calling the XML Web Service for the Agent resource with the XML as argument. This illustrates how local context is resolved into a generic (XML-based) query that the server can understand and process.

[0894] In order to optimize the query, the Agency XML Web Service exposes methods for passing several arguments qualified with operators (and, or, etc.). The interpreter preferably sends one call to the XML Web Service for the Agent resource with all the link arguments.

[0895] Semantic Query Implementation Scenarios. The following are exemplar scenarios illustrating the implementation and operation of Semantic Query Documents according to a preferred embodiment of the present invention.

[0896] Scenario 1: Loading an SQML Document. The client creates a temporary file and writes into it a buffer containing the attributes of simple, local HTML page. This page includes the client framework component (e.g., an ActiveX control, a Java applet, an Internet Explorer behavior, etc.). The page is initialized with this component opening the SQML file and a unique ID identifying the Information Agent instance. The component itself opens the SQML file. In other words, the client framework tells the plug-in what SQML query document to open. The plug-in opens the Semantic Query Document by interpreting it as described above.

[0897] Scenario 2: Open Documents. The client opens the standard dialog box, which allows users to select files to be opened. The dialog box is initialized with standard document file extensions (e.g., PDF, DOC, HTM, etc.). When users select the documents, the dialog box returns a list of all the opened documents. The client creates a new SQML file and adds resource entries with the paths of the opened documents. The new SQML file is given a unique name (preferably based on a globally unique identifier (GUID)). Because this is a temporary file, the name is preferably not exposed to users. The methodology proceeds to Scenario 1 as described above.

[0898] Scenario 3: Open Folder in Documents. The client creates an SQML file (as described above) and initializes it with one resource entry: file=/<folderpath>?includesubfolders=(true|false). The SQML file is loaded as in Scenario 1 by enumerating all the documents in the folder and displaying the metadata for the documents.

[0899] Scenario 4: Save as Agent. The client opens a dialog box allowing users to set the Agent name. The client renames the Agent in the Semantic Environment (see below) to the new name. The Agent being saved may be temporary or may already have been saved under a different name. The Information Agent preferably suggests an Agent name.

[0900] Scenario 5: Save into Blender. The client opens a dialog box that allows users to select a Blender. The dialog box preferably allows users to create a new Blender. When the Blender is selected, the client opens the Blender’s SQML file into the SQML object model and adds the new entry (the currently loaded SQML file). It then increments the reference count of the current entry.
Scenario 6: Drag and Drop. The client creates and opens an SQML file with a single resource entry, for example, similar to the following:

```
<resource type="nervana:ai">  
  agent://documents.all@abc corp.com  
  <link predicate="nervana:relevancy"  
    type="nervana:filepath"  
    c:\foo.doc  
    </link>  
  </resource>
```

This example assumes that an icon representing "c:\foo.doc" is dragged and drop over an icon in the Information Agent referring to the Agent "agent://documents.all@abc corp.com."

Scenario 7: Multiple Drag and Drop. The client creates and opens an SQML file with a single resource entry, for example, similar to the following:

```
<resource type="nervana:ai">  
  agent://documents.all@abc corp.com  
  <link predicate="nervana:relevancy"  
    type="nervana:filepath"  
    c:\foo1.doc  
    c:\foo2.doc  
    c:\foo3.doc  
    </link>  
  </resource>
```

This example assumes that multiple icons representing "c:\foo1.doc," "c:\foo2.doc" and "c:\foo3.doc" are dragged and dropped over an icon in the Information Agent referring to the Agent "agent://documents.all@abc corp.com."

Scenario 8: Smart Lens. When a Smart Lens is selected in the Information Agent, the Information Agent indicates to the Semantic Environment Manager (see below) that a Smart Lens has been selected for the Information Agent identifier. When the Skin notices that the mouse is over an object (e.g., via the "mouseover" event in the document object model (DOM)), it calls the Presenter first to find out whether the Information Agent is in Smart Lens mode. The client framework determines this by asking the Semantic Environment Manager if an Information Agent with the identifier is in Smart Lens mode. Because the Semantic Environment Manager caches this information from the Information Agent itself, it can answer the question on behalf of the Information Agent. If the Information Agent is in Smart Lens mode, the client framework preferably obtains the SQML buffer from the system clipboard via the Semantic Environment Manager. This is because a Smart Lens is a virtual "paste" in that it obtains its information from the clipboard. In other words, any object or Agent that is copied to the clipboard can be used as a Smart Lens (even regular text). The framework obtains the SQML buffer and instantiates resource components for every resource in the SQML buffer. The client framework calls the resource API GetInformationForSmartLens passing the XML information for the currently displayed object to the resource. All resources preferably return Smart Lens metadata to the client framework. Each resource preferably returns metadata in the form of a list of Smart Lens information nuggets. Each nugget contains a text entry and a list of query buffers (in SQML). The text entry contains simple text or a custom text format, for example, similar to the following:

```
[0904] Steve reports to <A>Patrick</A>. 54 critical-priority messages</A> relating to this one.

[0905] Each "<A>" tag pair preferably includes a corresponding SQML query buffer in the information nugget. The client framework formats the text into DHTML (or similar presentation format) for display in the Information Agent (e.g., as a balloon popup or another user interface, preferably not to block or conceal the object that the mouse is over). The client framework displays a user interface for links (analogous to HTML links) where the containing "<A>" and "</A>" tags are found. When a link is invoked, the client framework calls the Semantic Environment Manager to create a new cache entry. The Semantic Environment Manager indicates what file-path the entry should be stored in. The client framework writes the SQML buffer for the <A> tag that was clicked into the file. The client framework pushes the SQML document to the Semantic Environment Manager and loads the SQML into the Information Agent (via Dynamic HTML). Because the Semantic Environment Manager includes this SQML document as the current document, users are able to save the document via the "save" button in the Information Agent (e.g., "save as Agent" or "save into Blender"). An example of information that a Smart Lens can display is as follows:

```
[0906] The Agent Email. Technology.All@Marketing has a total of 300 objects that relate to this object. Critical Priority: 50 objects. High Priority: 50 objects. Medium Priority: 100 objects. Low Priority: 145 objects.
```

In the preferred embodiment, if users do not click any of the links in the balloon, no SQML document is created and nothing gets added to the Semantic Environment. This is because the Smart Lens preferably represents only a "potential query."

In the preferred embodiment, any information that can be contained in SQML can be invoked as a Smart Lens (e.g., Agents, people, documents, Headlines, Classics, Agencies, text, HTTP URLs, FTP URLs, files from the file-system, folders from the file-system, email URLs from an email application such as Microsoft Outlook, email folder URLs, etc.). For example, users are able to copy regular text from text-based applications to the clipboard. If users enter the Information Agent and select the Smart Lens, the SQML version of the text will be invoked as a Smart Lens (via a "document" resource). If the "text Smart Lens" is then hovered over a document object, the document resource representing the text Smart Lens optionally displays the similarity quotient, indicating to users similarities between the Smart Lens object and the object underneath the mouse. If the object underneath the mouse is a person object, the document resource may decide to "ask" the Agent representing the person object whether the Agent is an expert on the information contained in the text.
Alternatively, the Smart Lens might display links to similar documents or email messages the person has authored that relate to the text.


[0909] Copy: On invocation of a Copy command from within the Semantic Environment, the client framework copies an SQML buffer to the system clipboard with a custom clipboard format. This ensures that other applications (e.g., Microsoft Word, Excel, Notepad, etc.) do not recognize the format and attempt to paste the information. The SQML buffer is preferably consistent with the semantics of the object being copied. For example, a copy operation from an object being displayed in the Presenter is copied as a resource with the appropriate resource type and URL from whence the metadata came. Copying an icon representing an Agent copies the URL of the Agent or the cache entry referring to the Agent’s entry in the Semantic Environment. Copying information from a desktop application (e.g., Microsoft Outlook) copies SQML with a resource type referring to the source application and URLs pointing to the objects within the application. These URLs are preferably resolvable at runtime by the interpreter to objects within that application. For example, copying an email message from Outlook to be copied into the Semantic Environment may create a resource entry as follows:

```
<resource type="nervana:outlook:email:message">
  outlook://c:/temp/fox.html
</resource>
```

[0910] Paste: On the invocation of a Paste command, the client framework creates an SQML file based on the clipboard format of the information being pasted. For example, if the clipboard contains a file path, the SQML file contains a link (from the resource on which the Paste was invoked) to an object with the file path. This file is opened as described above. If the clipboard format is an URL, the object is of the URL object type. If the format is regular text, the object contains the actual text with, in this example, the resource type `nervana:text`. Alternatively, the client framework creates a temporary cache entry, stores the text there (e.g., as a `.TXT` file), and stores the SQML object with a reference to the file path and the object type, in this example, `nervana:file:pathname`. When the interpreter is invoked, it creates an XML metadata version of the text and invokes the resource with the XML link argument. If the clipboard format is the SQML clipboard format of the present invention, a similar process is performed, except that if a file is created, the extension will be SQML (or `.SQML`). This indicates to the interpreter that the object is an SQML file and not just a regular text file.

[0911] f. Semantic Environment

[0912] A preferred embodiment of the Semantic Environment of the present invention provides a view of every Agent and Agency available to user via the Information Agent. This preferably includes Agents that have been saved locally into the favorites “My Agents” list, recently used Agents, Agents on local Agencies, and Agents on remote Agencies. Remote Agencies include Agencies that announce their presence via multicast on the local area network, Agencies available on a Global Agency Directory, and Agencies available on a custom Agency Directory. Agents can be dynamically added to the Semantic Environment by invoking their URL. In the preferred embodiment, the Semantic Environment hierarchy has the pattern shown in SAMPLE C of the Appendix hereafter.

“Recently Used,” “Recently Created” Agents are preferably collapsed to “Recent Agents.” Optionally, “All Agents,” “Deleted Agents,” and “Custom View” may be added.

[0913] The Agencies view allows users to see the Agents in the main view by Agency. The object type view allows users to see the same Agents, but filtered by object type. Other views operate in similar fashion, e.g., “By Context” (based on Context Templates) and “By Time.” The Semantic Environment merges the notion of “favorites” with the notion of “history.” The Semantic Environment optionally adds dynamically managed views such as “recently used Agents,” etc. These views are preferably updated by code running within the Semantic Environment Manager (see below).

[0914] Icons incorporated into the Semantic Environment may include the following:

<table>
<thead>
<tr>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>All container object types</td>
</tr>
<tr>
<td>All document file types</td>
</tr>
<tr>
<td>Breaking News Agent Icon Qualifier (e.g., an exclamation point)</td>
</tr>
<tr>
<td>Special Agent Icon Qualifier (e.g., a halo)</td>
</tr>
<tr>
<td>Standard Agent for each object types</td>
</tr>
<tr>
<td>Agency</td>
</tr>
<tr>
<td>Agent View Containers</td>
</tr>
<tr>
<td>My Agents</td>
</tr>
<tr>
<td>Breaking News Agents</td>
</tr>
<tr>
<td>Favorite Agents</td>
</tr>
<tr>
<td>Special Agents</td>
</tr>
<tr>
<td>Recently Used Agents</td>
</tr>
</tbody>
</table>

[0915] Snapshots. Users are preferably able to save a snapshot of the Semantic Environment. A Semantic Environment snapshot essentially is a time-based cache of the state of the Semantic Environment. In the preferred embodiment, a snapshot includes locally stored state with the following information:

[0916] All the Agencies at the snapshot time that have new Agents.

[0917] The last Agent creation time of each Agency (based on the Agency’s clock).

[0918] The current time of each Agency (based on the Agency’s clock).

Snapshots are preferably accessible to users. The Information Agent filters the Semantic Environment to show only Agencies in the snapshot list, and the Agents in each of those Agencies created between the last Agent creation time and the snapshot time for each Agency.

[0919] g. Semantic Environment Manager

[0920] The present invention provides a Semantic Environment Manager that exposes APIs to manage the Semantic Environment objects. In the preferred embodiment, the managed Semantic Environment objects are comprised primarily of Agent references via SQML buffers. The Semantic Environment Manager also exposes APIs to navigate the Semantic Environment. In the preferred embodiment, the Semantic Environment Manager allows instances of the Information Agent to:

[0921] 1. Register itself at the Semantic Environment Manager. The Semantic Environment Manager preferably maintains information on all open Information Agent instances. It does this because a number of services (e.g., clipboard access, Smart Lens access, etc.) are performed across applications such as the shell extension application and the Presenter component running.
inside a browser control. For example, when the Presenter loads a new SQML document into the display area, it needs to get a cache entry from the Semantic Environment Manager. It asks the Semantic Environment Manager to create a new cache entry for a given SQML buffer. The Semantic Environment Manager creates the cache entry, writes the SQML buffer to the file path corresponding to that entry, creates a temporary HTML file initialized with an ActiveX control, Dynamic HTML Behavior, Java applet (for an equivalent client runtime engine) pointing to the cache entry, and returns the cache entry identifier and the file path to the temporary HTML file to the Presenter. For example, in the preferred embodiment, the temporary HTML file may be named as follows:

```
[0922]  c:\windows\temp\nervana_39fc54bc-81e5-4954-8cecf=3d1a54935a0d.htm
```

where 39fc54bc-81e5-4954-8cecf=3d1a54935a0d is the cache entry identifier. The containing Information Agent automatically detects new documents being loaded (via events in the contained Information Agent control). The containing Information Agent is able to respond when users hit "save" (e.g., "save as Agent" or "save into Blender"). The Information Agent accomplishes this by getting the current document file path, getting the cache entry identifier from the file path (since the file path is partially named with the identifier), and displaying the metadata for the cache entry (name, description, etc.) when users hits "save as." The Information Agent optionally asks the Semantic Environment Manager to resave the cache entry with a new name. The Information Agent registers itself (preferably at startup) with the Semantic Environment Manager with the process ID of its instance. The Semantic Environment Manager allocates a new identifier for the Information Agent and stores metadata for the Information Agent instance (for example, whether it is currently in Smart Lens mode). The Information Agent stores this identifier. The Information Agent preferably passes the identifier to the Semantic Environment Manager each time it makes a call. The Information Agent initializes the Presenter with the identifier. In the preferred embodiment, the client framework calls the Semantic Environment Manager with the identifier each time it needs cross-application services. The Semantic Environment Manager stores the process identifier of the Information Agent instance in order to garbage collect all Information Agent entries when the Information Agent processes have terminated. The Semantic Environment Manager preferably accomplishes this in order to remove the Information Agent entry because the Information Agent may not "know" when it is terminated.

```
[0923]  2. Add new Agent references to the Semantic Environment. Agent reference entries are preferably stored in a database, the file-system or a system store (e.g., the Windows registry). In the preferred embodiment, each Semantic Environment entry contains:
```

```
[0924]  a. Identifier. This uniquely identifies the Agent in the Semantic Environment.
```

```
[0925]  b. Name. This indicates the name of the Agent.
```

The Information Agent sets a default Agent name when a new Agent is created. This Agent name is set based on the manner of creation. For example, if document "foo" is copied and pasted over Agent "bar," the Information Agent may create a temporary Agent named "bar" related to "foo" (current time). The current time is stored to uniquely name the Agent (in the event that users reissue the same query. Users are able to rename the Agent as desired.

```
[0926]  c. Query Buffer. This indicates the buffer containing the SQML for the Agent.
```

```
[0927]  d. Type. This indicates the Agent type (e.g., Standard Agent, Blender, Search Agent, Special Agent, etc.)
```

```
[0928]  e. CreationTime. This indicates when the Agent entry was created
```

```
[0929]  f. LastModifiedTime. This indicates when the Agent entry was last modified
```

```
[0930]  g. LastUsedTime. This indicates when the Agent entry was last used
```

```
[0931]  h. UsageCount. This indicates the number of times the Agent has been used either standalone, as a filter, or as a Smart Lens.
```

```
[0932]  i. Attributes. These are the Agent attributes (e.g., normal, temporary, virtual, and marked for deletion). If the entry is temporary, it means users have not explicitly saved it as a local Agent. Temporary entries are preferably used in cases where users compose complex queries using drag and drops, but without saving any of the intermediate queries as Agents. When users save a query as an Agent, the Information Agent resets the temporary flag indicating that the query entry is now permanent.
```

```
[0933]  j. ReferenceCount. This indicates the number of references to the Agent by other Agents and Blenders. The count is initialized to 0 when a new Agent entry is created.
```

```
[0934]  3. Delete Agents from the Semantic Environment. This is preferably accomplished in two phases. Agents can be marked for deletion, in which case the Semantic Environment Manager sets a flag indicating that the Agent entry is in the "trash can." The Agent entry can also be permanently deleted, in which case the entry is removed from the cache all together.
```

```
[0935]  4. Change the properties of an Agent in the Semantic Environment (e.g., reset the temporary flag for an Agent when users save the Agent).
```

```
```

```
[0937]  6. Enumerate the cache to retrieve entries preferably corresponding to:
```

```
[0938]  a. All Agents
```

```
[0939]  b. Deleted Agents
```

```
[0940]  c. The most frequently used Agents
```

```
[0941]  d. The most recently used Agents
```

```
[0942]  e. The most recently created Agents
```

```
[0943]  f. Filters for each object type underneath the aforementioned views (e.g., Documents, Email, Events, etc.)
```

```
[0944]  g. Filters for Agencies that host Agents in the aforementioned views, filters for object types on the Agencies, and the Agents that fit those views (Documents, Email, etc.)
```

```
[0945]  h. Filters for Special Agents based on the Context Template (e.g., Headlines, Classics, Newsmakers, etc.).
```

For samples of these enumerations and views, FIGS. 12-14 and 17-19 showing the Semantic Environment Tree View.
7. Filter the Agents list based on counters updated via invocations from instances of the Information Agent. Each instance of the Information Agent preferably communicates with the one Semantic Environment Manager. That way, updates are user-oriented rather than session-oriented. For example, if users open an Agent in one Information Agent, the Agent entry will show up in the recently used Agents view in another Information Agent. The Semantic Environment Manager maintains information on the number of times each Agent has been used, the last time each Agent has been used, etc. It filters the Agents. For example, the most frequently used Agents are filtered based on the N Agents with the highest usage counts, where N is configurable and where the filter is only applied after some stabilization wait period (e.g., after the total usage count is at least Y, where Y is also configurable, for example, based on simple heuristics such as the expected number of Agent uses in a two week period). The recently used Agents are filtered based on the usage time (which is stored on a per-Agent basis and which is updated by instances of the Information Agent each time the Agent is used). The recently created Agents are filtered based on the Agent creation time. The deleted Agents are filtered by examining the “marked for deletion” flag on each Agent. The Favorites Agents are filtered by examining the “marked as favorites” flag on each Agent. For each of the aforementioned parent views, the underlying views are populated using simple filters. The Agencies view is populated by examining each Agent returned in the parent view and extracting unique Agencies there from. The object type views underneath each of the Agencies displayed therein are then populated by filtering the Agents based on the Agent object type (e.g., document, email, event, etc.). The Blenders view is filtered by displaying only Agents that have the “Blender” type. The object type views are directly filtered using the Agent object type. The “My Agencies” view displays local Agencies. Each view underneath this is preferably an object type view filtered using each available Agent on the Agency. The “Fly Context” view is populated by filtering only for Special Agents (preferably created with a Context Template) and checking for the context name (e.g., Headlines, Classics, etc.).

8. Maintain a reference count for Agents in the Semantic Environment. It is the responsibility of the calling component (the Information Agent) to increment and decrement a document entry’s reference count. The Information Agent preferably accomplished this by way of a drag and drop, copy and paste, etc. In other words, actions that create new queries that refer to existing Agents.

9. Empty the Semantic Environment. This deletes all Agents.

10. Perform garbage collection. The Semantic Environment Manager automatically deletes all old (and temporary) Agents. The cache may be configured to keep a history of Agents up to a certain age. For example, if the cache is configured to only maintain information for two weeks worth of Agents, it periodically checks for temporary Agents that are older than two weeks. If it finds any, it automatically deletes Agent entries that have a reference count of zero. This preferably occurs in cases where the Information Agent creates a new cache entry but does not create another entry (Agent or Blender) that refers to it. In other words, the Information Agent performs link-tracking for the immediate link (to avoid complexity). The Semantic Environment Manager optionally performs deep garbage collection. This occurs periodically on a configurable schedule. This applies to entries that have a reference count greater than zero but have no actual references because links were not maintained when other entries were deleted. This feature is incorporated into the preferred embodiment to minimize complexity because the Information Agent preferably does not track references between Agents and Blenders when Agents and Blenders are saved or edited. In an alternative embodiment, the Presenter performs lazy Agent link-tracking when an Agent is invoked. The client framework ignores all references that have been deleted from the Semantic Environment, analogous to how a Web page returns a 404 (file not found) error when one of its links has been deleted. In other words, the present invention provides for the situation of incomplete queries. By way of example, a possible scenario may be as follows: Blender B1→referred to Blender B2→referred to Agent A1→referred to Agent A2

In this case, the reference count of each entry will be 1, even though the reference count of the chain is 4. As such, it is possible to have stale entries even though the reference counts are greater than zero. For each entry being garbage collected, the garbage collector searches for any reference to the entry in all SQML documents. If no reference is found, the entry is removed (if it is temporary and older than the age limit).

11. Handle notification management. Users are able to register for notifications from any

Agent in the Semantic Environment (e.g., saved or local Agents, Standard Agents, Blenders, etc.). In the preferred embodiment, notification methods include sending email, instant messages, pager messages, telephony messages, etc. The Semantic Environment Manager includes a Notification Manager (see below), which will manage notification requests from users via the Information Agent. The Notification Manager stores a list of notification requests. A notification request preferably includes the Semantic Environment object ID (which identifies the Agent), the type of notification (email, IM, etc.) and the destination, e.g., the email address, etc. The Notification Manager periodically polls each Agent in the notification request list to “ask” if there are any new objects. The Notification Manager also passes the “last requested time” (based on the destination Agent’s clock). The Agent responds with the number of new objects (by invoking its stored query and passing back the number of objects in the query results that were created since the “last requested time”). The Agent responds with the current time (on its clock). The Notification Manager stores the Agent’s time to avoid time synchronization problems. Alternatively, the client and all Agencies use the same time server (a time Web service) to get their time to ensure that all time comparisons will be on the same scale.

Agency Directories. In the preferred embodiment, the Semantic Environment Manager preferably maintains an Agency list for each Agency “directory.” The multicast network preferably looks to the Semantic Environment Manager as a directory of Agencies. In the preferred embodiment, there
is a default Global Agency Directory configured with the URL to an XML Web Service on a public system. This XML Web Service stores a cache of all registered Agencies (preferably with the information described above, including ID, URL, etc.). The XML Web Service exposes methods to allow Agencies to register their presence on the Agency Directory. The XML Web Service filters redundant entries. The XML Web Service also exposes methods to allow users to enumerate all Agencies on the Agency Directory. The Semantic Environment Manager enumerates the directory in this manner. Preferably, the Information Agent considers the Agency Directory as an extension of the Semantic Environment, and allows users to browse and open Agents on the Agencies listed on the Agency Directory. Users are preferable able to add URLs to custom Agency Directories that may be installed on the internal network. The present invention contemplates the creation and integration of customizable Agency Directories. This essentially is an alternative to using multicast for discovery in cases where multicast may not be enabled on the network (for bandwidth conservation reasons) or where certain subnets on the wide area network do not support multicast.

[0954] h. Environment Browser (Semantic Browser or Information Agent™)

[0955] The Environment Browser, or Information Agent, hosts a regular Web browser component (such as the Internet Explorer ActiveX control), and is primarily responsible for taking an SQML file and rendering the results via the Presenter. In the preferred embodiment, it does this by opening a local HTML file initialized with a reference to the SQML document cache entry of the SQML file. The HTML file loads the Presenter through a control (e.g., ActiveX, Java, Internet Explorer behavior, etc.). This control retrieves the SQML document from the cache (via the Semantic Environment Manager) and loads the SQML file as described above. The control adds objects to the Web browser document object model (DOM) as it received callbacks from resources indicating that objects are available to be converted to XHTML (or equivalent presentation format, preferably via the current XSLT and/or script-based skin) and pushed into the DOM for presentation. The Information Agent allows users to open an SQML file or an entry in the cache (via the cache ID). The Information Agent also allows users to navigate back and forward, and to navigate the first document in the stack (analogous to the “back,” “forward,” and “home” options in today’s Web browsers, the difference being that in this case SQML documents are being opened for interpretation and display (of the results) as opposed to HTML and other documents.

[0956] In a preferred embodiment, the Semantic Environment can include a toolbar popup menu option having tools allowing users to import local search results into the Semantic Environment, e.g., via a Dumb Agent, to create a new Special Agent, a new Blender, or a new local Agency. Alternatively, these tools can be collapsed into one tool button that invokes a wizard from which users can select the kind of Agent (Dumb, Smart, Special) or Agency they wish to create. A sample dialog can allow users to search the Semantic Environment using keywords, for example, creating a new Smart Agent (with the appropriate SQML). Users are preferably able to customize the name of the new Smart Agent and add an optional description. A “Save” tool popup menu options of the toolbar that can allow users to save a newly created or opened Agent permanently into the Semantic Environment (e.g., into the “favorites” list), or to save the Agent into a Blender. A Smart Lens tool menu option of the toolbar can allow users to invoke the Smart Lens (based on the Smart Agent or object that is currently on the clipboard, communicating to the Presenter that the user wishes to use the clipboard contents as a Smart Lens. The Presenter can preferably automatically invoke the Smart Lens functionality for any object users hover over (e.g., with the mouse). The menu can also show a “Paste as Smart Lens and Pin” option that keeps the Smart Lens turned on (even across Agent navigations) until the user explicitly turns off the Smart Lens. An “Open Agent” dialog view allows users to open server-side Agents from the Semantic Environment and change the “view” of the Environment (e.g., Large Icons, Small Icons, List, etc.). A standard Windows “Open” dialog allows users to import a “regular” document from the file system into the Semantic Environment of the Information Nervous System. For example, a Dumb Agent can be created that refers to the document(s). In use, when the Dumb Agent is invoked, the document(s) is opened in the Information Agent and all of the semantic tools (e.g., smart copy and paste, Context Templates, etc.) are enabled with the document(s). By this method, the browser can make a regular, “stupid” document on the file system semantically “smart.” A custom “Open Documents in Folder” dialog can allow users to search for documents on a folder on the local file system and import them into the Semantic Environment. This makes the documents “smart” by “exposing” them via the semantic tools of the Information Nervous System (e.g., smart copy and paste, Context Templates, etc.). A “Browse for Folder” dialog box can be displayed when users select a browse option, allowing users to select a folder to open (from the local file system). A page from the “Add Blender” wizard can allow users to select whether they want to create a standard Blender or a virtual Blender.

[0957] i. Additional Application Features

[0958] Additional Menu Extensions and other Framework Features. The system client preferably installs a menu extension to applications that support programmatic extensions but that do not already support copying data to the clipboard. These include applications such as Microsoft Windows Media Player and Microsoft Outlook (for email message headers). In the preferred embodiment the menu extension reads “Copy.” The system copies the selected object as an XML object to the Windows system clipboard. For example, the system plug-in for an email Microsoft Outlook copies a selected email object as an Email XML Object. For applications that already support the clipboard, no extension is needed.

[0959] Server-Side Favorite Objects. On Agencies that support User Stute, users are able to mark objects as “favorites.” When an object is marked as a favorite, the Presenter invokes a method on the Agency’s XML Web Service. The XML Web Service adds a semantic link between the user object and the object in question. In the preferred embodiment, users are able to view favorite objects via the All.MyFavorites.All Default Agent. This Agent returns all objects that have been marked as favorites. The Agency administrator is able to create sub-Agents such as All.MyFavorites.Technology.XM- L All.

[0960] The Presenter allows users to mark and unmark favorites, which is also a means of redefining the structure that the servers and Agencies export. The use of “favorites” scenario is especially valuable in cases where users may see
objects of interest and not want to navigate them immediately. The favorites feature may optionally also be used by the Agency to recommend objects to users. In the preferred embodiment, these recommended objects are retrievable via the All.Recommended.Agent. The Agency recommends objects based primarily on objects that users have marked as being favorites. Server-side favorites will also preferably be used with the “favorites,” Classics and Recommendations Context Templates.

Agent Screen Savers. A preferred embodiment of the present invention allows users to select any subscribed Agent as a screen-saver. Users are preferably warned that Agents may expose sensitive data and given an opportunity to determine whether it is safe to use a particular Agent as a screen-saver. In the preferred embodiment, the system client is capable of loading any subscribed Agent as a screen-saver. In an alternative embodiment, users may combine Agents to provide a desired screen-saver presentation. Alternatively, a screen-saver may be a structured Skin that includes displayed parallel Agents, for example, in four quadrants of the screen.

Agent-Agent Smart Lens. In an alternative embodiment, the system client supports the use of a Smart Lens (invoked either through an Agent or a Blender) as a context to invoke another Agent or Blender. For example, users may select All.Critical.Priority. All and want to use that Agent as a Smart Lens to browse All.Understood. All in order to find out all objects that are critical priority and which are also understood by the destination Agent.

Smart Lens Sample User Interface Illustrations. FIGS. 29-31 provide exemplar balloon popup menus associated with the Smart Lens feature of an Information Agent according to the present invention. FIG. 29 shows a sample of a balloon popup menu in the context Results Pane with a Smart Agent as the Smart Lens. This shows a popup window that is displayed when users select the Smart Lens icon on an information object. This sample shows a case where the Smart Agent titled “Documents on Reuters Related to [My Nervana UI Specification]” is on the clip board and is “posted” as a Smart Lens over an email object titled “Yuying’s Thoughts on the Nervana UI.”. FIG. 30 shows a sample of a balloon popup menu in the context Results Pane with an object as the Smart Lens (and “hovered” over an Agent). This sample illustrates that the Smart Lens is connative (A [SMART LENS] B ⇒ B [SMART LENS] A). The results section of the context pane is identical to the example shown in FIG. 29, indicating that the Smart Lens in the preferred embodiment is connative. FIG. 31 shows a sample of a balloon popup menu in the context Results Pane with an information object as the Smart Lens and an information object as the item being “lensed over.” In this sample, an object titled “My Nervana UI Specification” has been copied to the clipboard (its SQML representation) and pasted as a Smart Lens over another object (in the Results Pane) titled “Yuying’s Thoughts on the Nervana UI” (an email object). In this sample, the user has the option of selecting a predicate that is semantically consistent with the combination of a document to an email message. FIG. 32 shows a sample of a variant of the balloon popup menu of FIG. 31 showing the relatedness measure of the two objects (the Smart Lens object and the “lensed over” object), both as a percentage and graphically, in this example as a bar chart.

F l o w: 33-35 show sample tables illustrating the behaviors and relational contains objects types predicates when using Smart Lenses. FIG. 33 shows the Agent-Object scenario for all information wherein the Smart Lens behavior is connative, for example, A [Smart Lens] B ⇒ B [Smart Lens] A. FIGS. 34-35 show the Object-Object scenario for document and email, respectively, also wherein the Smart Lens behavior is connative, for example, A [Smart Lens] B ⇒ B [Smart Lens] A.

Blender Skin User Interface Illustrations. FIG. 36 is a user interface sample illustrating semantic results Player/Preview Control. The Information Agent Presenter preferably attaches this control to each Results Pane. The Player/Preview Control allows users to navigate the results in the Results Pane, to animate the results (play, stop, pause, change, speed up, etc.) and to filter the results (e.g., in the case of the results of a Blender). FIG. 37 is a user interface sample showing the semantic results of a Blender. In this sample, the Blender Skin has reserved parts of the display area as separate frames for each Agent in the Blender, and attached a Player/Preview Control to each frame, thereby allowing users to individually navigate, control and animate the results of each Agent in the Blender. Alternatively, a Blender Skin can display the merged results from all the Agents in the Blender (with one Player/Preview Control attached), can display the results in frames according to information object type, etc.

Multiple Drag and Drop. In an alternative embodiment, the system client allows users to select multiple documents or folders from the desktop and use them as the basis of relational queries on an Agent or Blender. This allows users to further refine a query using multiple documents as the refining tool. For example, the user may optionally indicate whether they want the union or intersection of the results (using each of the documents as a filter). This creates an SQML file with one resource (the object over which the links were dragged) and multiple links (one per document or dragged object). The client’s SQP preferably interprets this by retrieving the XLM metadata for all the object filters and calling the destination Smart Agent’s XML Web Service with the XML arguments. In the preferred embodiment, the Agency’s XML Web Service categorizes the XML metadata arguments, forms the proper SQP representation of the query and returns the results.

URL Shortcut Conventions. Agencies of the present invention may share the Internet Web since they are optionally installed as Web applications. As a result, Agencies can be referred to using the Web’s naming scheme (e.g., a regular HTTP URL). In the preferred embodiment, the present invention exposes shortcut naming conventions and URLs that are specific to the Information Agent’s Semantic Environment.

Agent Shortcut URL Convention. The Agent shortcut URL convention is:

```
agent://<agentname>@<agencyurl>?start=<start>&end=<end>&skin=<skinurl>
```

When invoked, this is preferably mapped to a fully-qualified HTTP URL, for example:

```
http://path to Agency ASP; or CGI script?agentname=<agentname>&start=start&end=end&skin=SkinURL.
```
An example of an Agent shortcut URL convention is as follows:

agent://email.technology.wireless.all@marketing.abccorp.com/?start=0&end=25&skin=html://www.nervana.net/skins/email/abccorp/emailskin.xml

This URL is resolved by the client as follows: Start the Web service proxy, open the WSDL file http://abc.com/nervanaroot/webservice.wsdl and ask the Web service for the statistics of the Agency named “Marketing.” For HTTP access, this will be resolved to a path to the ASP or CGI. For example:


The start argument indicates the zero-based starting index of the object to return first. The end argument indicates the end index. The Skin URL is optional. If no Skin URL is specified, the client loads the Agent with the Agent’s default Skin.

A locally saved Agent may be accessed with agent://<agentname>@localhost. For example: agent://Documents.[Related to My Business Plan] @localhost will load the locally saved Agent (in My Agents) named “Documents.[Related to My Business Plan].”

Agency URL Convention. An example is as follows:

agency://<agencyname>/<domainname>/query= getproperties&getstats&getagents&agentviewfilter= <agentviewfilter>&agentnamecontainsfilter= <agentnamecontainsfilter>&agenttypefilter= <agenttypefilter>&agentobjecttypefilter=agentobjecttypefilter>

In this example, the query argument is “get-properties.” The URL retrieves the properties of the Agency itself (e.g., the name, the display name, whether it is local or remote, etc.). Alternatively, if the property is “getstats,” the URL retrieves the statistics of the Agency (total number of Agents, number of Standard Agents, number of Compound Agents, number of Domain Agents, total number of objects, number of document objects, number of email objects, etc.). In the preferred embodiment, the getproperties flag is the default, meaning that the properties are retrieved if no other argument is specified. It is either the getproperties or getstats arguments are specified, preferably no other arguments are specified alongside.

The agentviewfilter argument is optional and allows the caller to specify an Agent view within with to restrict the search. For example, an Agent view “Reuters News” may be installed on the server to only return Agents that manage news objects from Reuters. The agentnamecontainsfilter argument is optional and allows users to filter the results by a search string for the Agent name. The agenttypefilter is optional and allows users to filter Agents based on Agent type (Standard Agent, Compound Agent, or Domain Agent). The agentobjecttypefilter argument is optional and allows users to filter the results with the object type the Agent manages (e.g., email, documents, people, etc.). Examples include the following:


Objects URL Convention. Agency objects can be accessed directly from a client. The URL convention is:

objects://<querystring>/<agencyname>/<domainname>/querytype= <objectid>searchstring&objecttype=objecttypefilter

The objecttypefilter argument is optional and can be used to filter the returned objects by object type. It is an enumeration of known object types (e.g., document, email, event, etc.). Examples include the following:

objects://34547848@support.attwireless.com/querytype=objectid will return the object with the objectid 34547848.

objects://80211@support.attwireless.com/querytype=objectid&objecttype= will return the email objects matching the query string “80211”

Category URL Convention. The URL convention is:

category://categoryname/@<kburl>?semanticdomainname= <semanticdomainname>

The semanticdomainname argument is optional. In the preferred embodiment, if it is left out, the default domain of the KBS will be selected. An example is as follows:

category://technology.wireless.all@abccorp.com/marketingknowledge.asp

This corresponds to the “Technology.Wireless.All” category for the default domain on the knowledge-base installed on the abccorp.com/marketingknowledge.asp web service. This will be resolved to the following HTTP URL: http://abccorp.com/marketingknowledge.asp?category=“technology.wireless.all.” An example of a fully qualified version of the category URL may be:

category://technology.wireless.all@abccorp.com/ marketingknowledge.asp?semanticdomainname= “InformationTechnology”
Sharing and Roaming Client Information. In the preferred embodiment, users are able to share Agents (including Blenders) with others by sending them via email, instant messaging, etc. Local information users are preferably able to either store Agent information locally or have the information roam with them (e.g., via ActiveX Mirror support in Windows 2000 for department-wide roaming, via a proprietary XML Web Service on a Global Agency Directory (using passwords for identity), or via integration with Microsoft .NET from Services, which employs Microsoft’s Passport identity service).

Local Agencies. The system client preferably also allows users to create and add local Agencies that run a local instance of the KIS to the “My Agencies” list. In this embodiment, the client also allows users to delete a personal Agency.

User-Experience Consistency and Non-Disruptiveness. The Information Agent (semantic browser) of the present invention provides a consistent and undisturbing user experience. In other words, the Information Agent seamlessly coexists with Today’s Web browser. Tools such as “Back,” “Forward,” “Home,” “Stop,” “Refresh,” and “Print” preferable work as they do with Today’s Web browser so as not to confuse the user. Many of the tools remain the same albeit the functionality is different. In addition, new tools are preferably added to the toolbar and menu options reflecting the new functionality in the semantic browser (these can be seen by observing the toolbar in the screenshots).

FIGS. 78 and 79 illustrate exemplary functional mappings of the present invention demonstrating preferred mappings for introducing new functionality to users while maintaining metaphor consistency. FIG. 38 is a comparison of default user interface toolsets for Today’s Web browser and a preferred embodiment of the Information Agent of the present invention. FIG. 39 is a comparison of default user interface toolsets for the file system Microsoft Explorer/Document Viewer and a preferred embodiment of the Information Agent of the present invention.

Providing Context in the Present Invention

The present invention provides Context Templates, or scenario-driven information query templates that map to specific semantic models for information access and retrieval. Essentially, Context Templates can be thought of as personal, digital semantic retrieval “channels” that deliver information to a user by employing a predefined semantic template. In the preferred embodiment, the semantic browser 30 allows the user to create a new “Special Agent” using Context Templates to initialize the properties of the Agent. Context Templates preferably aggregate information across one or more Agencies.

By way of example only, the present invention defines the following Context Templates. Additional Context Templates directed towards the integration and dissemination of varied types of semantic information are contemplated within the scope of the present invention (examples includes Context Templates related to emotion, e.g., “Angry,” “Sad,” etc.; Context Templates for location, mobility, ambient conditions, users tasks, etc.).

“Headlines” Context Template. The Headlines Context Template (and its resulting Special Agent) can be analogized to a personal, digital version of CNN’s “Headlines” program that delivers news in a time-sensitive manner. To access information from specific Agencies, the user activates the “Headlines” Context Template for one or more Agencies, sorted according to the information creation or publishing time and a configurable amount of time that defines information “freshness.” For example, CNN’s “Headlines” displays headlines every 30 minutes (around the clock). In a preferred embodiment, the Information Agent 30 of the present invention allows users to create a Headlines Special Agent using the following filters and parameters:

Information Object Pivots. The resulting Blender shows result that relate to these object. This is an optional parameter. If it is not specified, headlines are displayed for the entire Agency (without any object-based filter).

Predetermined “freshness” period. For example, 30 minutes, 1 hour, etc.

Predicate. This will define how the Information Object Pivot links to the information to be retrieved. Examples are: “related to,” “possibly related to” (uses a text-based search), “authored” (in the case of a person object), “possibly authored,” “has expertise on,” etc. The default predicate “relevant to” is preferably used by default. This default predicate is resolved by the Agent by intelligently mapping it to specific predicates.

Agency(ies). This includes the Agencies on which to check for headlines. At least one Agency must be specified and there is no limit to the number of Agencies that can be specified. The user may indicate whether all Agencies in the “recent” and/or “favorites” lists should be used.

Category list. For example “Technology, Wireless, All”. This acts as an additional filter for the query.

In addition to freshness, the Headlines Context Template preferably incorporates how “hot” the result items are in order to determine the ranking of the results. This may be accomplished by querying the Agency to find out the number of semantically related objects on the Agency, which is a good indicator of whether an object’s topic is “hot.” In addition, returned objects (or items) are preferably sorted by freshness or as new.

By way of example, SAMPLE D of the Appendix hereto illustrates an SQML output from a Headlines Context Template of the preferred embodiment. In this example, the Context Template retrieves all information from four different Agencies (marketing, research, sales, and human resources), with a freshness time span of 30 minutes, and with a “relevant to” predicate (indicating a semantic query). In the preferred embodiment, the SQML of this example, as for all Context Templates, can optionally form the basis of a Smart Lens, smart copy and paste, drag and drop and other tools in the semantic toolbox.

“Breaking News” Context Template. The Breaking News Context Template (and its resulting Special Agent) can be analogized to a personal, digital version of CNN’s “Breaking News” program that delivers news in a time-sensitive manner. To access information from one or more Agencies, preferably sorted by the information creation or publishing time or the event occurrence time (in the case of event), and with a configurable amount of time that defines freshness and a configurable “deadline” for events to define time-criticality. For example, the Context Template can be defined to filter information objects posted in the last one-hour, or events holding in the next one day.
In the preferred embodiment, the Breaking News Context Template is different from Breaking News Agents. The Context Template is a template that defines static query parameters that are passed to one or more Agencies. A Breaking News Agent is any Smart Agent users may have created and is essentially user-created and user-customizable. By way of example, a Breaking News Special Agent based on the Breaking News Context Template may inform users of information objects posted in the last hour or events holding in the next day that relate to a local document (or any other local context, if specified). But a Breaking News Agent gives users the flexibility of receiving alerts for “Events on wireless technology being given by a member of my team and holding either Seattle or Portland in the next 24 hours and which relate to this document on my hard drive.” The Breaking News Agent provides users much greater flexibility and personalization than the Breaking News Context Template. An advantage of the Breaking News Context Template is that it preferably forms the basis for intrinsic alerts by using parameters that qualify as “breaking” for typical users.

“Conversations” Context Template. The Conversations Context Template (and its resulting Special Agent) can be analogized to a personal, digital version of CNN’s “Crossfire” program in how it conveys semantic information. Like “Crossfire,” which uses conversations and debates as the context for information dissemination, in the preferred embodiment, the Conversations Special Agent tracks email postings, annotations, and threads for relevant information. The Conversations Context Template may be thought of as the Headlines Context Template filtered with email object type. In addition to the “Headlines” parameters, the Conversations Context Template preferably (but optionally) contains the following parameters:

Minimum thread length to return. The user optionally indicates that he or she only wants email threads with at least one reply, two replies, etc. In many instances, the number of threats provides an indication of semantic significance. The default is zero.

Distribution list filter. The user optionally restricts the returned email to those that have members of one or more distribution lists on the “from,” “to,” “cc,” or “bcc” lines. This allows the user wants to monitor debates from preferred groups, divisions, etc.

Distribution line filter. The user optionally restricts the returned email to those that have the filter address on the “from,” “to,” “cc,” or “bcc” lines.

“Upcoming Events” Context Template. The Upcoming Events Context Template (and its resulting Special Agent) can be analogized to a personal digital version of special programs that convey information about upcoming events. Examples include specials for events such as “The World Series,” “The NBA Finals,” “The Soccer World Cup Finals,” etc. The equivalent in a knowledge-worker scenario is a user that wants to monitor all upcoming industry events that relate to one or more categories, documents or other Information Object Pivots. The Upcoming Events Context Template is preferably identical to the Headlines Context Template except that only upcoming events are filtered and displayed (preferably using a semantically appropriate “context skin” that annotates events and time-criticality). Returned objects are preferably sorted based on time-criticality with the most impending events listed first.

“Discovery” Context Template. The Discovery Context Template (and its resulting Special Agent) can be analogized to a personal, digital version of the “Discovery Channel.” In this case, the emphasis is on “documentaries” about particular topics. Unlike in the case of “Headline News,” the primary axis for semantic information access and retrieval is not time. Rather, it is one or more category with an intelligent aggregation of information around those categories. In a preferred embodiment of the present invention, the Discovery Context Template simulates intelligent aggregation of information by randomly selecting information objects that relate to a given set of categories and which are posted within an optionally predetermined, configurable time period. While there is an optional configurable time period, the semantic weight as opposed to the time is the preferred consideration for determining how the information is to be ordered or presented. The present invention allows for different axes to be used, for example, the semantic weight for the category or categories being “discovered,” time, randomness, or a combination of all axes (which would likely increase the effectiveness of the “discovery”). The Discovery Context Template preferably has the same parameters as the Headlines Context Template, except that the freshness time span is replaced by an optional maximum age limit, which indicates the maximum age of information (posted to the Agency) that the Agent should return.

“History” Context Template. The History Context Template (and its resulting Special Agent) can be analogized to a personal, digital version of the “History Channel.” In this case, the emphasis is on disseminating information not just about particular topics, but also with a historical context. For this template, the preferred axes are category and time. The History Context Template is similar to the Discovery Context Template, further in concert with “a minimum age limit.” The parameters are preferably the same as that of the Discovery Context Template, except that the “maximum age limit” parameter is replaced with a “minimum age limit” parameter.
(or an optional “history time span” parameter). In addition, returned objects are preferably sorted in reverse order based on their age in the system or their age since creation.

[1012] “All Bets” Context Template. The All Bets Context Template (and its resulting Special Agent) represents context that returns any information that is relevant based on either semantics or based on a keyword or text-based search. In this case, the emphasis is on disseminating information that may be even remotely relevant to the context. The primary axis for the All Bets Context Template is preferably the mere possibility of relevance. In the preferred embodiment, the All Bets Context Template employs both a semantic and text-based query in order to return the broadest possible set of results that may be relevant.

[1013] “Best Bets” Context Template. The Best Bets Context Template (and its resulting Special Agent) represents context that returns only highly relevant information. In a preferred embodiment, the emphasis is on disseminating information that is deemed to be highly relevant and semantically significant. For this Context Template, the primary axis is relevancy. In essence, the Best Bets Context Template employs a semantic query and will not use text-based queries since it cannot guarantee the relevancy of text-based query results. The Best Bets Context Template is preferably initialized with a category filter or keywords. If keywords are specified, categorization is performed by the server dynamically. Results are preferably sorted based on the relevance score, or the strength of the “belongs to category” semantic link from the object to the category filter.

[1014] “Favorites” Context Template. The Favorites Context Template (and its resulting Special Agent) represents context that returns “favorite” or “popular” information. In this case, the emphasis is on disseminating information that has been endorsed by others and has been favorably accepted. In the preferred embodiment, the axes for the Favorites Context Template include the level of readership interest, the “reviews” the object received, and the depth of the annotation thread on the object. In one embodiment, the Favorites Context Template returns only information that has the “favorites” semantic link, and is sorted by counting the number of “votes” for the object (based on this semantic link).

[1015] “Classics” Context Template. The Classics Context Template (and its resulting Special Agent) represents context that returns “classical” information, or information that is of recognized value. Like the Favorites Context Template, the emphasis is on disseminating information that has been endorsed by others and has been favorably accepted. For this Context Template, the preferred axes includes a historical context, the level of readership interest, the “reviews” the object received and the depth of the annotation thread on the object. The Classics Context Template is preferably implemented based on the Favorites Context Template but with an additional minimum age limit filter, essentially functioning as an “Old Favorites” Context Template.

[1016] “Recommendations” Context Template. The Recommendations Context Template (and its resulting Special Agent) represents context that returns “recommended” information, or information that the Agencies have inferred would be of interest to a user. Recommendations will be inserted by adding “recommendation” semantic links to the “Semantics lulks” table and by mining the favorite semantic links that users indicate. Recommendations are preferably made using techniques such as machine learning and collaborative filtering. The emphasis of this Context Template is on disseminating information that would likely be of interest to the user but which the user might not have already seen. For this Context Template, the primary axes preferably include the likelihood of interest and freshness. In the preferred embodiment, the Context Template is implemented by generating SQML that has the PREDICATETYPE_IDLIKELYTOBEINTERESTEDIN predicate as the primary predicate filter on the Agencies in the Semantic Environment.

[1017] “Today” Context Template. The Today Context Template (and its resulting Special Agent) represents context that returns information posted or held (in the case of events) “today.” The emphasis with this Context Template is preferably on disseminating information that is deemed to be current based on “today” being the filter to determine freshness. In the preferred embodiment, the Today Context Template results are a subset of the Headlines Context Template results wherein the results posted “today” or events holding “today” are displayed.

[1018] “Variety” Context Template. The Variety Context Template (and its resulting Special Agent) represents context that returns random information. The emphasis with this Context Template is preferably on disseminating information that is random in order for the user to get a wide range of possible information items. In the preferred embodiment, the primary axis is randomness, albeit the “random” items will be semantically relevant to the query filter (using the “relevant” to predicate).

[1019] b. Context Skins
[1020] The present invention includes a special class of Skins called “Context Skins” Context Skins include presentation information that conveys the semantics of the context that they represent. For example, a Context Skin for the Today Context Template may display a background or filter effects with a clock pointing to midnight, or some other representation of “Today.” In yet additional examples, a Context Skin for the Variety Context Template may show transform effects like bowling balls falling over randomly (indicating the randomness of the results); the Breaking News Context Skin may show effects and light animations with flashing text, ambulance red lights, etc. to indicate the criticality of the context; and the History Context Skin may show graphics that indicate “age”; for example, old cars, clocks, etc.

[1021] Context Skins preferably “honor” the presentation template for object types being displayed. For example, email objects may be displayed with a background showing stamps or a post office clock in addition to graphics that indicate the Context Template. Because some Context Templates cut across Agencies—and therefore cut across ontologies—they need not display any information that indicates ontology (e.g., industry information). However, Context Skins that are initialized with a category filter preferably indicate the category or ontology of the Context Template. Typically this will be represented with graphics elements (and filters, transforms, etc.) that indicate the industry or genre of the ontology. For example, a Pharmaceuticals Context Skin may have filter effects showing laboratory equipment; an Oil and Gas Context Skin may show pictures of oil rigs; and a Sports Context Skin may show pictures of sports gear, etc.

[1022] c. Skin Templates
[1023] The present invention allows a user to select different kinds of Skins, depending on the task at hand. The implication of having flexible presentation is that the user can select the best presentation mode based on the current task. For example, users may select a subtle Skin when working on
their main machine and where productivity is most critical and where effects are not. Users may select a moderate Skin in cases where productivity is also important but where effects will also be nice to have as well. Users may select an exciting Skin for scenarios like second machines, for example where users are viewing information in their peripheral vision, and features such as text-to-speech to alert them on breaking news is important. Exciting Skins may feature animations, storyboard like effects for deep information, objects displayed on motion paths, and other effects. Exciting Skins are most likely going to be used with screensavers. The choice of Skins is preferably user-definable.

[1024] d. Default Predicates
[1025] In the preferred embodiment, each object type includes a default predicate that links it with other object types. This provides users with an intuitive method of dynamically linking objects together without requiring a separate evaluation of the predicate to use for the semantic link. For example, a drag and drop operation from a document object to an Agent that returns documents can have the predicates “Related To” and “Possibly Related To.” When a document object is dragged on top of a document Agent, the semantic browser of the present invention displays a popup menu option that allows users to select the predicate to use for the semantic query. In an alternative embodiment, other related popup menus may be incorporated, e.g., a first popup menu that allows users to select the link or predicate template; child popup menus that display the actual predicates for the selected template. The default predicate is preferably inserted in the dynamically generated SQML from which the query will be invoked.

[1026] By way of example, a default predicate may be “relevant to.” This predicate maps a query that returns information in the document Agent that is relevant to the object being dragged. The advantage of having a default predicate in this case is that the semantic browser of the present invention may display a popup menu option named “Open” that in turn invokes a query using this predicate. The semantic browser may also display a popup menu option named “Open with Link” that has submenu options with specific predicates. The default predicate makes the system easier to use because users are able to browse the system using dynamic linking, knowing that the default predicate will be the sensible option giving the source object and that target Agent or object.

[1027] In addition to being used in drag and drop scenarios, Default Predicates are optionally used in Smart Lenses, smart copy and paste, etc. Default Predicates may be analogized to degenerate smart links that return “the right thing” given the context. Preferably the default predicate will be “relevant to,” which may in turn produce “The right thing” as the appropriate query result for a semantic distance of one. In an alternative embodiment, the Default Predicate may be a merger of several specific predicates. For example, the Default Predicate for a document-to-people drag or drop, copy or paste, or Smart Lens may be “relevant to” and may be invoked by the KIS Agency XML Web Service as, for example, a cascaded query involving “author,” “expert on,” and “annotated” predicates. In other words, “relevance” is interpreted smartly by the present invention and may involve merging together different predicates.

[1028] Default Predicates allow users to navigate the system quickly and efficiently and with little thought. Default Predicates provide the system with simplicity and make it intuitive to use. In addition, users are comfortable with Default Predicates because users are already used to invoking HTML links on today’s Web where there is only one predicate: “invoke”.

[1029] e. Context Predicates
[1030] Context Predicates are predicates that are defined at a high level of abstraction and which map to a relevant subset of the Context Templates. Context Predicates allow a user to select a predicate filter based on a Context Template, rather than on a low-level system predicate. When the query is invoked with the Context Predicate, filtering the containing SQML with the filter parameters of the Context Template generates a new SQML query. For example, the Context Predicate “Best Bets” maps to the Context Template of the same name and filters a query with those information objects that are “best bets” (typically, these will be those items that are returned from a semantic query and not from a text-based query). Similarly, the Breaking News Context Predicate filters items based on whether they qualify with the filter conditions of the Breaking News Context Template. In general, Context Predicates are applied for object types that are consistent with the Context Template (for example, the Context Predicates “Experts” and “Newsmakers” will only be valid for queries that return “Person” objects).

[1031] f. Context Attributes
[1032] Context Attributes are “virtual attributes” that are cached as part of each XML object that an Agency returns to the client. These attributes are dynamic in that they reflect the current context in which the results are being displayed. For example, where relevant, the Context Attribute “Best Bet” is attached to each XML result that satisfies the semantic query filter in the SQML of the current query. The results of a semantic query with default predicates might include both semantic and non-semantic (text-based query) results. The Agency processing the query may cache Context Attributes for the XML results that are “Best Bets” by running a semantic sub-query on the SQML with the result object as a filter. In this case, the schemas for the “Object” and derived types should include attribute fields for each relevant Context Template (e.g., a “Best Bet” attribute, “Headline” attribute, etc.). This is the preferred implementation. Alternatively, the semantic browser calls the Agency, passes each XML object as an argument and “asks” whether the object satisfies the Context Attribute. Other examples are a Headline Context Attribute that indicates whether the object qualifies as a “Headline” in the context of the current query, a “Classics” attribute, etc. The semantic browser should display a user interface indicating whether the context attribute is set or not.

[1033] Context Attributes provides further benefits over the prior art systems in that they make the system easier to use. For example, a user can perform a drag and drop operation to generate a relational query that includes both semantic and non-semantic query filters (as processed by the Agency when it receives the SQML arguments from the client). In one embodiment, the browser “asks” the user whether he or she desires a broad query or a “Best Bet” query. In this mode, the user effectively applies for an additional filter before the query is issued. Alternatively, the Agency, in concert with the semantic browser, preferably returns the results of the broad query, and also qualifies each result with a context attribute and corresponding user interface indicating whether each result object is “broad” or a “Best Bet.” The same applies to other object types like the “Person” object type. Rather than having the user indicate whether a relational query to a Person
Agent should return "authors," "experts," or "annotators," the browser can issue a broad query and then qualify the results (with help from the Agency) with whether each returned "Person" object is an "author," "expert," or "annotator," for the current context.

Context Palettes are a very powerful feature of the present invention that involves invoking Context Templates dynamically for the currently selected object within the semantic browser. Essentially, Context Palettes are preferably automatically invoked and displayed when users select any object in the Results Pane. Context Palettes enable users to always have the context for the currently displayed results at their disposal. In addition, the semantic browser constantly refreshes the palette for the currently selected object, thereby guaranteeing that the context for the object is always up to date. In a preferred embodiment, this is accomplished via a timer that triggers a refresh action or by querying the SQML query processor for the Context Palette for whether there is any new object since the last time the palette was refreshed.

In the preferred embodiment, results displayed in Context Palettes are "first-class" information objects in the same way as the information objects displayed in the main Results Pane. In other words, Context Palette results are preferably used with all of the present invention’s semantic tools, e.g., smart copy and paste, Smart Lens, Deep Information, etc. The same preferably is true for results displayed in other context panes anticipated in the present invention.

The present invention preferably includes the following Context Palettes. In the preferred embodiment, users have the option to "scroll" through the different Context Palettes for a selected object. The incorporation of additional and different Context Palettes is expressly anticipated, and may parallel the addition of Context Templates.

"Headlines" Context Palette. This uses the Headlines Context Template and employs SQML that has the SQML of the Headlines Context Template with an additional link to the currently selected object, and the default predicate for the object-type combination. In particular, the SQML will be keyed off resources that map to all the favorite Agents or recent Agents in the Semantic Environment. The user configures whether he or she wants Favorite Agents, recent Agents, or both to be used when generating the Context Palette. In addition, the Headlines Context Palette is also configurable to show headlines without any filter for the number of objects to be displayed or the "freshness" time limit. In this case, the palette will allow the user to navigate all the relational results sorted by the publication or post time.

"Breaking News" Context Palette. Contains relational results from every Breaking News Agent in the Semantic Environment using the default predicate of the object-type combination, and linked with the currently selected object. In addition, results for the default Breaking News Context Palette are displayed. The semantic browser of the present invention will dynamically generate SQML with as many (and identical) resource or link combinations as there are Breaking News Agents, with additional links that have the default predicate and the resource qualifier of the currently selected object (a file-path, folder-path, object://URL, etc.). The semantic browser of the present invention invokes the generated SQML query and loads the palette windows with the SSRML results. The Breaking News Context Palette preferably contains navigation controls to allow users to navigate the results in the Context Palette.

Conversations" Context Palette. Similar to the Headlines Context Palette except utilizing the Conversations Context Template.

"Newsmakers" Context Palette. Similar to the Headlines Context Palette except utilizing the Newsmakers Context Template.

"Upcoming Events" Context Palette. Similar to the Headlines Context Palette except utilizing the Upcoming Events Context Template.

"Discovery" Context Palette. Similar to the Headlines Context Palette except utilizing the Discovery Context Template.

"History" Context Palette. Similar to the Headlines Context Palette except utilizing the History Context Template.

"All Bets" Context Palette. Similar to the Headlines Context Palette except utilizing the All Bets Context Palette.

"Best Bets" Context Palette. Similar to the Headlines Context Palette except utilizing the Best Bets Context Template.

"Favorites" Context Palette. Similar to the Headlines Context Palette except utilizing the Favorites Context Template.

"Classics" Context Palette. Similar to the Headlines Context Palette except utilizing the Classics Context Template.

"Recommendations" Context Palette. Similar to the Headlines Context Palette except utilizing the Recommendations Context Template.


"Variety" Context Palette. Similar to the Headlines Context Palette except utilizing the Variety Context Template.

"Timeline" Context Palette. This Context Palette preferably contains merged results from the Headlines, Best Bets, History, and Upcoming Events Context Templates. The Timeline Context Palette preferably allows the user to navigate all objects on the semantic timeline based on the currently selected object. The timeline may contain information items based on their publish/post time, event items based on their appointment time, etc. Essentially, with the Timeline Context Palette, the user navigates relevant (and perhaps other semantically related) objects using time as the primary axis for information conveyance.

"Guide" Context Palette. The preferred embodiment of the present invention includes a unified Guide Context Palette. This Context Palette combines all Context Palettes. In other words, each window in the Guide Context Palette corresponds to one result from each of the other system Context Palettes. The user interface for the Guide Context Palette allows the user to scroll through the results for each Context Palette in each window or to animate the results using animation techniques, for example, fade-in/fade-out techniques. A preferred use of the Guide Context Palette is to view context for the currently selected object in a minimal viewing space. In the preferred embodiment, the use has the option of viewing all Context Palettes side-by-side (vertically, horizontally, diagonally, etc.), docked, or in other arrangement formats.

Context Palette User Interface. The user interface for Context Palettes is preferably configurable based on the layout Skin for the currently displayed Agent. In the preferred embodiment, Context Palettes may be docked on the left, right, top or bottom of the Results Pane. Context Palettes may
be collapsed in order to minimize intrusion into the viewing area and dynamically re-expanded to full view. Skins may also allow Context Palette windows to be resized to variable sizes or preset, fixed sizes. Alternatively, some Skins may also animate Context Palettes results.

[1055] By way of example, FIG. 40 illustrates a user interface showing Agent results and corresponding Context Palettes. In the example, several Context Palettes are collapsed and the Context Palettes are skinned (or presented) to be vertically docked on the right side of the display, or Results Pane.

[1056] h. Intrinsic Alerts

[1057] In a preferred embodiment, in addition to the Breaking News Agent, the present invention provides for Intrinsic Alerts. While conceptually similar to Breaking News Agents, Intrinsic Alerts are fundamentally different in operation. In the case of Breaking News Agents, the present invention signals the user as to breaking news notifications after polling each Breaking News Agent specified by the user and querying it to find if there is anything related to the current object that is breaking. An Intrinsic Alert does not require the user to specify a Breaking News Agent or otherwise perform any action in order to introduce breaking news notification. An Intrinsic Alert is automatically signaled in the user interface (for all currently displayed objects) when there is an event that relates to the object at issue in a fundamental, intrinsic way. For example, if the current object is a document, the present invention polls the Agency from whence the document came and asks the Agency if there is any recently posted information on the Agency that relates to the object. If the current object is a person, the present invention may poll the Agency and ask if the person recently sent email, recently posted a document, recently annotated a document, recently joined or exited a distribution list, etc. This allows the user to have in-place information within the native context of the object in a time-sensitive manner.

[1058] In the preferred embodiment, the default implementation for Intrinsic Alerts will poll only the Agency from whence the object came. This has the advantage of simplifying the user interface; if the user wants to perform cross-Agency queries, he or she has the option to drag and drop, copy and paste, etc. in order to invoke relational queries. In alternative embodiments, Intrinsic Alerts will poll multiple Agencies, including Agencies other than from whence the object came, in an effort to locate breaking news notifications.

[1059] In an alternative embodiment, the present invention is configurable to maintain information as to whether a user has accessed an object. This may be analogized to how an email server keeps track of what email messages a user has read. In an embodiment in which the Agency supports per-object, per-user server-side state, Intrinsic Alerts are always accurate because the Agency indicates that there is "intrinsic breaking news" only if there is information on the Agency that relates to the object in question that has not been accessed or read by the user. This alternative is preferably accomplished means of an additional filter on the SQML query.

[1060] The alternative of a per-object, per-user server-side state is required for this embodiment has disadvantages, especially for Agencies that will hold massive amounts of information and will have a huge number of users (e.g., Internet-based Agencies). In this situation, the system does not scale well if state is maintained per object and per user.

[1061] In an alternative embodiment where the Agency does not support per-object, per-user server-side state, the Agency may be configured with a static freshness time limit for Intrinsic Alerts. For example, the server may be configured with a freshness time limit of thirty minutes, in which case the server would respond in the affirmative if an Intrinsic Alert query is received within thirty minutes of the arrival of a new object that relates to the object in the query. In a preferred embodiment, the KIS Agency maintains information on the average information arrival rate. This way, a busy server will have a lower freshness time limit than a server that seldom receives new information. This embodiment is not as accurate as if the server kept per-object, per-User State because the average arrival rate produces only an approximation of whether an alert should be signaled. This embodiment will still result in reduced information loss. In the preferred embodiment, the present invention optionally signals Intrinsic Alerts in a non-intrusive manner that suggests their probabilistic nature (i.e., that an alert is only a best guess).

[1062] i. Smart Recommendations

[1063] Smart Recommendations represent semantic queries to the Semantic Network, for inferred semantic links, using an object as an Information Object Pivot. For example, the Inference Engine may infer that users would like to attend a certain event, based on events they have attended in the past, the fact that they have been engaged in many email conversations with the presenter of the event, etc. By way of example, in the preferred embodiment, this information is available in a Smart Recommendations popup context Results Pane such as that shown in FIG. 41. This is similar to what users see for a given object against the Recommendations Context Template.

[1064] In the preferred embodiment, each link is generated by the object Skin or a special recommendations information pane Skin and will link to SQML containing the predicates for the inferred semantic links.

[1065] 6. Property Benefits of the Present Invention

[1066] The Information Nervous System of the present invention provides proper context, meaning and efficient access to data and information to allow users to acquire actionable knowledge. Many of the advantages of the Information Nervous System over today's Web and the conceptual Semantic Web are derived from its use of the technology layers shown in FIG. 42. The various embodiments of the present invention demonstrate the advantages as they relate to the properties required to produce an integrated and seamless implementation framework and resulting medium for knowledge retrieval, management and delivery, which include Semantics/ Meaning; Context-Sensitivity; Time-Sensitivity; Automatic and Intelligent Discoverability; Dynamic Linking; User-Controlled Navigation and Browsing; Non-HTML and Local Document Participation in the Network; Flexible Presentation that Smartly Conveys the Semantics of the Information being Displayed; Logic, Inference, and Reasoning; Flexible User-Driven Information Analysis; Flexible Semantic Queries; Read/Write Web; Annotations; "Web of Trust"; Information Packages ("Blenders"); Context Templates; and User-Oriented Information Aggregation.

Semantics/Meaning

[1067] The present invention employs semantic links, ontologies, and other well-defined data models using XML. As a result, an Agency as described above has the power of a semantic Web site in that its information includes semantics. In addition, by providing meaning as an intrinsic part of the
XML Web Service, it further provides context-sensitivity, time-sensitivity, etc. associated with the subject matter information.

Context-Sensitivity

[1068] Intelligent system Agents described above monitor the private context of users and automatically alert users when there is relevant information on an information source (or sources) related to the specific context. By way of example, these specific contexts may include the following: My Documents, My Web Portal, My Favorite Websites, My Email, My Contacts, My Calendar, My Customers, My Music, My Location, “This” document, “This” Web site/page, “This” email message, “This” contact, “This” event in my calendar, “This” customer, and “This” music track, album or play-list.

[1069] The present invention provides a context-sensitive user experience via the use of information Agents associated with the server 10 and via the semantic browser 30 and associated XML Web Service. For example, users automatically connect information in “My Documents,” “My Email,” etc. (from application islands such as the file system, Microsoft Outlook, etc.) to remote information sources that have semantically relevant information. Users have the flexibility to make these connections in real-time via application-level innovations that reside on top of the Semantic Network such as the new query tools described above, for example, drag and drop, Smart Lenses, smart copy and paste, etc. It is also contemplated that such application tools can be used independent of a Semantic Network, for example, integrated into an existing browser of Today’s Web.

[1070] In a preferred embodiment, the KIS of the present invention pulls semantic information from the Semantic Web or other repository with semantic markup (preferably via RDF plug-ins) into its Semantic Network. Alternatively, the system 10 of the present invention exists without the Semantic Web. In this situation, the KIS builds its own Semantic Network (e.g., a private semantic web) from data sources that the system administrator selects (e.g., email, documents, etc.). The system 10 of the present invention is able to utilize the actual semantic applications with a semantic backend (which can optionally include the Semantic Web). The system 10 thus provides context-sensitivity via integration with client-side applications (including the proprietary semantic browser 30), location-tracking tools, etc. and the proprietary XML Web Service (which the Semantic Web does not describe). More specifically, while the conceptual Semantic Web describes architecture for semantic linking and knowledge representation, it does not address scenarios and innovations using XML Web Services to provide context-sensitivity, time-sensitivity, dynamic linking, Context Templates, Context Palettes, etc. In contrast, the present invention addresses semantic linking via the semantic data model and Semantic Network as well as provides software services for context sensitivity, time-sensitivity, semantic queries, dynamic linking, Context Templates, Context Palettes, etc. via integration with its proprietary XML Web Service.

Time-Sensitivity

[1071] The present invention has an intrinsic notion of time-sensitivity. For example, by providing features related to time-sensitivity such as Breaking News Agents, Breaking News Context Templates, Breaking News Context Palettes and intrinsic alerts, the present invention demonstrates the importance of time as an element in semantics and presentation. While not universally true, generally speaking old information is usually not as relevant as new information. For example, when CNN interrupts news broadcast to show breaking news, the interruption is based on a combination of semantics (the relevance of the breaking news about to be displayed) and the fact that the news is indeed breaking. Except in those rare cases where the Web author specifically builds in time-prioritized analysis, this time-sensitivity element as an axis for alerts and presentation is totally lacking in Today’s Web and in the conceptual Semantic Web.

[1072] The present invention allows users to select Smart Agents as Breaking News Agents. Any information being displayed will show alerts if there is relevant breaking news on a breaking-news Agent. For example, with the present invention, a user is able to create an Agent as: “All Documents Posted on Reuters today” or “All Events relating to computer technology and holding in Seattle in the next 24 hours” as Breaking News Agents. Because these Agents are personal (“breaking” is subjective and depends on the user), the user is able to schedule notification on events in Seattle in the next 24 hours, events on the West Coast in the next week (during which time he or she can find an inexpensive flight), events in the United States in the next fourteen days (the advance notice for most U.S. air carriers to obtain a competitively priced across-continental flight), events in Europe in the next month (likely because he or she needs that amount of time to get a hotel reservation), and events anywhere in the world in the next six months.

[1073] The present invention further supports a Breaking News Context Template based on which users can create Breaking News Agents. In addition, the present invention supports a Breaking News Context Palette that allows users to view all displayed results in the context of a template-based definition of “breaking news,” thereby seamlessly and intelligently integrating context and time-sensitivity.

[1074] The present invention further provides a powerful personal historian tool for performing historical analyses. Using browse history, past events, and document creation times, the system 10 can compensate for faulty memory by recalling details from an event. For example, showing results to the query “The coworkers who attended the design meeting from 6/1/98 through 6/99.” Alternatively, the system may seek for a cluster of events. For example, investigators may ask for “All stock market transactions greater than $10M related to the airline stocks from 7/1/01 up to 9/11/01” or “Show all documents created within a ten day window of this event”.

Automatic and Intelligent Discoverability

[1075] The system 10 of the present invention has an intrinsic notion of discovery. In a preferred embodiment, the KIS automatically announces its presence on a local multicast network, an enterprise directory (e.g., an LDAP directory or the Windows 2000 Active Directory), a peer-to-peer system or other system. Ideally, the semantic browser 30 periodically listens for multicast or peer-to-peer announcements and checks an enterprise directory or a Global Agency Directory. The browser also allows the user to navigate the system in a hierarchical fashion to locate additional Agencies. This way, users are notified when new Agencies are available and when existing Agencies expire. The semantic browser of the present invention preferably notifies users instantly when new Agen-
cies are available via namespace snapshots and periodic checks for announcements and directory presence.

The peer-to-peer aspect allows the system to scale and automatically populate the enterprise directory without any centralized maintenance (which is a large ongoing cost for organizations). The system preferably uses programmatic queries for new classes of servers, thereby eliminating the need for Web logs.

Dynamic Linking

The present system provides fundamental advantages over Today's Web and the conceptual Semantic Web by employing smart objects having intrinsic behavior. The system embeds behavioral characteristics in each Agency's XML Web Service, thereby make each node in the Semantic Network much smarter than a regular link or node on Today's Web or the Semantic Web. In other words, in the preferred embodiment, each node in the Semantic Network of the present invention links to other nodes independent of authoring. Each node has behavior that dynamically links to Agencies. Smart Agents also allow for such additional features as drag and drop and smart copy and paste, creating links to Agencies in the Semantic Environment, responding to lens requests from Smart Agents to create new links, including intrinsic alerts that will dynamically create links to time-sensitive information on its Agency, including presentation hints for breaking news (wherein the node can automatically link to breaking news Agents in the namespace), etc. These features dramatically increase the user's ability to, for example, find and navigate new links. Once the user reaches a node in the network, the user has many semantic means of navigating dynamically and automatically using context, time, relatedness to smart Agencies and Agents, etc. By making each node in the network smarter, the entire Semantic Network becomes a smart, virtual, self-healing and self-authoring network.

The dynamic linking technology of the present invention allows users to issue queries across local/remote information boundaries. For example, the present invention (preferably using SQML technology) allows a user to issue a query like: "Find me all email messages written by my boss or anyone in research and which relate to this specification on my hard disk." The client-side query processing technology (preferably via SQML) allows this flexible query because the processor links the metadata from the client with the remote XML Web Service that processes the relational query.

Smart and Dynamic Information Propagation. Dynamic linking as provided for in the present invention provide for intelligent information propagation. Because the Semantic Network can be navigated from many more axes than Today's Web or the Semantic Web, information sharing and propagation becomes much more efficient and information loss is minimized.

User-Controlled Navigation and Browsing

The dynamic linking property of the present invention allows for continuous semantic browsing as opposed to with Today's Web and the Semantic Web, where static links result in browsing "dead-ends." With Today's Web and the Semantic Web, the user typically browses to the desired location or effectively reaches an impasse where no further links are available. With dynamic linking, the user can, depending on the nature of the information space, continue browsing indefinitely since the node itself includes intelligence to dynamically update links.

For example, via the seamless integration of linking and semantic XML Web Services provided for by the present invention, users drag and drop files, links, etc. to Smart Agents to create new Smart Agents. Preferably this occurs recursively. Smart Agents, in turn, can, where appropriate, be made Breaking News Agents. Other nodes in the presentation display presentation hints indicating whether there is breaking news on any Breaking News Agent. To continue the example, the results of the Breaking News Agent query can be used as a Smart Lens, which shows further results. These results preferably include intrinsic alerts that provide the user with a context and time-sensitive path through the network. Subsequent results can be copied and pasted to any Agency, as well as dragged and dropped on other Smart Agents.

In the preferred embodiment, the dynamic linking of the present invention is applied both to objects within the semantic “sandbox” (objects that are in the system environment and displayed within the semantic browser) as well as to external objects that can be dynamically added to the environment. This provides a seamless, dynamic migration path from existing documents (on the file system, Today's Web, or other environments) to the system of the present invention.

FIG. 43 illustrates dynamic linking and user-controlled navigation and browsing according to a preferred embodiment of the present invention. Note that for purposes of this example, "Smart Links" refer to the dynamic, programmable semantic link of the present invention.

Non-HTML and Local Document Participation in the Network

The present invention does not require that documents be encoded as RDF or XML before inclusion in the network. Rather, the KIS (or Agency server) automatically extracts metadata from all sorts of documents and adds them to the Semantic Network. In addition, client-side dynamic linking, preferably via such features as drag and drop, smart copy and paste and Smart Lens, ensures that local documents of all types are linked to the network, thereby increasing the value and scope of the network. The present invention automatically extracts metadata from local documents and calls the KIS (via its XML Web Service) to retrieve semantically related information. Thus, the local document is not excluded from the network. The present invention empowers a user to drag and drop a document from a dumb environment (e.g., Today's Web or file system) into the system 10, thereby providing it semantic intelligence. Once the metadata is in the system 10, semantic tools such as semantic lenses, smart copy and paste, etc. may be performed to and with the object. Drag and drop is also supported directly from the user's file system and Today's Web into the system 10.

Flexible Presentation that Smartly Conveys the Semantics of the Information Being Displayed

The present invention empowers users with flexible presentation. Because the XML Web Service sends back XML, rather than HTML, and because the presentation is dynamically generated on the client, the user selects different "skins" with which to view semantic information. Skins preferably convert XML to a format suitable for presentation (e.g., XHTML+TIME, SVG, etc.), allowing the user to dynamically select Skin based on the capability of various display technologies. For example, SVG has many features...
that XHTML+TIME does not, and vice-versa. The user is able to select an SVG Skin for scenarios in which SVG is optimized. Alternatively, the user is able to select XHTML+TIME for other scenarios.

[1086] The flexibility of Skins as part of the present invention provide for application in additional situations. In various alternative embodiments, the user is empowered by text-to-speech Skins that may be running the semantic browser 30 on a second machine concurrently with a first or main machine, for example to assist blind users; dynamically resizable Skins that adapt to the size of the current view-port (thereby allowing the user to resize the window and yet retain a pleasant user experience); Skins that check local state to display semantic hints (e.g., the user’s calendar in the case of event information, e.g., free/busy information); Skins that display inline preview windows that save user navigation time and increase productivity; Skins that display different customizable hints for intrinsic alerts, breaking news, deep information, smart recommendations, intrinsic links, lens info, etc. Users are also allowed to select Skins to be used with smart screenshots, for example where users desire to view an Agent in a screenshot mode. In an alternative embodiment, the system 10 supports Skins for Context Templates (described above), e.g., Headlines, Newsmakers, Conversations, etc.

[1087] By virtue of allowing for flexible presentation, the present invention allows the user to select the best presentation mode based on the current task. For example, users can select a subtle Skin when working on their main machine where productivity is a higher priority than aesthetic effect. Users can select a moderate Skin in cases where productivity is important but where effects are desired or allowed. Users can select an exciting Skin for scenarios like wherein secondary machines are utilized—for example, where users are viewing information in their peripheral vision and desires features such as text-to-speech to alert them of breaking news, etc. Exciting Skins may alternatively feature animations, storyboard like effects for deep information, objects displayed on motion paths, and other special effects.

[1088] In addition, Skins according to the present invention are optionally configured with include and exclude object type filters. For example, a Skin may be configured to include only “documents” but exclude “analyt reports.” Because the Skin takes XML results to determine the ultimate presentation, the Skin can include or exclude objects in the XML (SRML) results based on an examination of the object type (or other attributes) of the returned objects.

Logic, Inference and Reasoning

[1089] The present invention provides for logic, inference, and reasoning. The semantic data model on KIS Agency preferably offers support for logic via database processing of the Semantic Network, conversion of semantic queries to SQL and other database query languages for logic processing, etc. In addition, the system 10 of the present invention preferably includes an Inference Engine for inferring links such as the experts on a particular category or information item, recommendations, probabilistic links (e.g., the probability that a person wrote a document), etc. As described above, an Inference Engine according to the present invention preferably observes the Semantic Network, mines it to infer new semantic links and represents resulting links in the Semantic Links table.

Flexible User-Driven Information Analysis

[1090] The present invention provides native support for flexible information analysis on the client. The Presenter of the present invention preferably utilizes Smart Lenses to allow a user to preview the results of a semantic query prior to issuing the query. The user is able to change relevant predicates and other filters in order to preview the results. In an alternative embodiment, the user has the option of invoking the query and using that as the basis of a new sub-query, if desired.

Flexible Semantic Queries

[1091] The present invention allows a user to issue very flexible semantic queries. The user is able to incorporate local context into queries, e.g., by using filters such as “relates to this document on my hard drive.” Neither Today’s Web nor the Semantic Web allow for this. In addition, the present invention preferably incorporates Smart Agents, which utilize references to a proprietary semantic query language (SQML) and includes local and remote resources, predicates, category references and objects. The present invention preferably incorporates the easy to use user interface for creating and editing Smart Agents (representing semantic queries) using a simple wizard model. As discussed above, the system 10 allows semantic queries to form the basis of new queries via the recursive drag and drop feature, e.g., a document or an HTML link can be dragged to an existing or new Smart Agents, thereby creating successive new Smart Agents. Smart Agents are alternatively used as lenses, can have objects pasted onto them to form new semantic queries, and can be added to Blenders, which in themselves are semantic query containers and which, in turn, can be filtered thereby creating sub-Blenders or containers of sub-Agents.

Read/Write Support

[1092] The system 10 of the present invention offers support for read/write functionality by providing an XML Web Service that allows a user to publish information directly into the Semantic Network. This could be any document, an annotation, or a semantic link that connects a broken link or provides a new link. This is all subject to security restrictions at the XML Web Service and operating system layer. The system 10 employs authentication, access control, and other services from the operating system and application server that sit underneath the XML Web Service layer. These security services are preferably used to secure read and write access to the Semantic Network.

Annotations

[1093] The present invention includes built-in support for Annotations. There is a special predicate “Annotated By” that defines an Annotation semantic link between a person object and any other information object (e.g., a document, email posting, online course, etc.). The system 10 includes presentation-layer support for Annotations by allowing users to navigate to Annotations via intrinsic links, Smart Lenses, etc. The manner in which the present invention incorporates Annotations provides advantages of existing techniques (such as in-place Annotation techniques that embed the Annotation as part of the information object it annotates). In the preferred embodiment of the present invention, Annotations are “first-class” information objects. This means that they can be linked to and from, “lens” over (using Smart Lens), copied and pasted (using smart copy and paste), etc. The present invention exposes Annotations to all of the semantic tools of the present invention, thereby facilitating a
user experience more powerful than capable with standard Annotation techniques. In addition, Annotations of the present invention are used with Context Templates. As a result, the Inference Engine is able to employ them to make the system smarter over time. In addition, the system provides a unique and easy means of annotating objects by sending specially formatted email (with a qualified message body) to the email Agent of an Agency.

“Web of Trust”

The present invention provides a “Web of Trust” via the XML Web Service. This service authenticates a user that wants to update the Semantic Network, make assertions, fix/update links, etc. This also allows rich content to be made available via the KIS Agency to registered subscribers for pay-per-view content. The value of the entire network increases when one can utilize the same platform tools to navigate seamlessly across many rich content sources.

Information Packages (Blenders)

The present invention provides for information packages or “Blenders.” Blenders are semantic containers that include references to semantic queries from Smart Agents. This allows a user to deal with related semantic information as a whole unit. The user is able to separately view the individual Agents within the Blenders or view the entire Blender as though the information therein was from one aggregate Agent. This is preferably accomplished by driving each Agent via calls to the XML Web Service. In the preferred embodiment, users drag and drop objects onto Blenders to create sub-Blenders. This is preferably accomplished recursively. Blenders can be created, deleted, and edited. The user is able to add and remove Smart Agents to or from Blenders.

Blenders can be thought of as a digital equivalent of a personal newspaper that contains different sections. For example, the USA Today, New York Times, Wall Street Journal, etc. contain different sections such as News, Business, Sports, Life/Entertainment, etc. Each of these sections corresponds to a Smart Agent entry in a Blender and the entire newspaper corresponds to the Blender. The flexible viewing and navigation provided by the present invention can be thought of as the digital equivalent of the user being able to browse each newspaper section completely and sequentially, one at a time, or browse the entire newspaper by starting as page one of each section, followed by page two of each section, etc.

Context Templates

As described in detail above, the present invention provides Context Templates, which are scenario-driven information query templates that map to specific semantic models for information access and retrieval. Essentially, Context Templates can be thought of as personal, digital semantic information retrieval “channels” that deliver information to a user by employing a predefined semantic template. In the preferred embodiment, the semantic browser allows the user to create a new Blender or Special Agent using Context Templates to initialize the properties of the Agent. Context Templates preferably aggregate information across one or more Agencies. In addition, Context Templates are preferably used with Context Palettes to provide intelligent, dynamic, in-place context for any information object that is displayed or selected by the user.

User-Oriented Information Aggregation

The present invention has intrinsic support for user-oriented information aggregation. Scenarios empower a user to view context and time-sensitive information as though they came from one source even if they cut across information repositories. This provides a significantly more productive user experience that with Today’s Web and the conceptual Semantic Web by providing user-oriented computing wherein the user is presented with the right information in the right context and at the right time, regardless of the source of the information. The Information Agent aggregates information dynamically, across information sources, using client-side semantic queries via SQML and aggregating the XML results that come from different Agencies’ response to SQML.

E. Scenarios

The following provides exemplary scenarios of the operation of preferred and alternative embodiments of the present invention as applied in different pragmatic situations.

1. Examples of Semantic Queries Utilizing the Present Invention

- Find All Context that Relate to the Specification on the File Path c:\spec.doc
- Drag and drop the icon representing a document to the icon representing the Information Agent. The file is opened in the semantic browser and the Context Palettes are displayed. In the preferred embodiment, these include some or all of the following Context Templates: Headlines, Discovery, NewsAgg, Upcoming Events, Timeline, Conversations, Variety, Classics, Best Bets, Today, Breaking News, etc. These palettes include relevant context from Agencies in the “recent” and “favorite” lists in the namespace.
- Find All Experts on the Agency Named “R&D” That Have Expertise on Wireless Technology
- Start the “New Smart Agent” wizard and select the “Use Context Template” option when creating the Agent. Select the “R&D” Agency from the “Select Agency” dialog and select the category called “wireless” from the category browser. Open the newly created Smart Agent.
- Find All Information on Reuters That Is Relevant to a Link on the Currently Viewed Web Page
- Drag and drop the link to the Agency icon representing “Reuters.” A new Smart Agent is created titled “Information on Reuters relevant to [link title]” and opened in the Information Agent.
- Find All Information on Reuters That Is Relevant to a Link on the Current Web Page and Which Is Relevant to the Specification on the File Path c:\spec.doc
- Drag and drop the icon representing the document to the Agent that was just created above (“All information on Reuters relevant to [link title]”). This creates a new Smart Agent titled “Information on Reuters relevant to [link title]” and relevant to spec.doc.” This illustrates user-controlled browsing and dynamic linking
- Find All Email on the Internal Agency Named “Marketing” Relevant to the First Article on Reuters That Was Returned in the Previous Query
Highlight the Reuters article object and click on the button for "Verbs." This displays a popup menu. Select "Copy." Find the icon representing the Agency titled "Marketing" (on the Shell Extension Tree View). Right-click the icon. Hit "Paste." This creates and opens a new Smart Agent titled "Information on 'Marketing' relevant to [Reuters's article title]." Focus on the frame in the results window showing email objects.

Navigate to the Author of the Email
Highlight the email object and click on the button for "Links." This displays a popup menu showing the intrinsic links. Navigate to the menu item titled "From:" This displays a popup menu showing the person object on the "from" line of the email object. Select the desired object. This opens a new Smart Agent in the Information Agent showing the metadata of the email that authored the email object. The context of the person is also displayed in the Context Palettes. Users are able to continue browsing using the person object or its context (on any of the Context Palettes).

Navigate to the Attachments in the Email
Highlight the attachment object and click on the button for "Links." This displays a popup menu showing the intrinsic links of the email object. Navigate to the menu item titled "Attachments." This displays popup menu showing the titles of the attachments. Select the desired attachment. This opens the attachment as a new Smart Agent in the Information Agent window. The context for the attachment is displayed in the Context Palettes.

Find All Events on the "Energy Industry Events" Agency That are Relevant to the Attachment
Highlight the attachment object and click on the button for "Verbs." This displays a popup menu. Select "Copy." Find the icon representing the Agency titled "Energy Industry Events" (on the Shell Extension Tree View). Right-click the icon. Hit "Paste." This creates and opens a new Smart Agent titled "Information on Energy Industry Events relevant to [email attachment title]."

Browse the "My Documents" Folder Using Reuters as a Context
In the Information Agent, select "Open Documents in Folder." Alternatively, drag and drop the "My Documents" folder to the icon representing the Information Agent. Indicate whether sub-folders are to be included. This creates and opens a new Dumb Agent titled "My Documents." When you click this Agent, the metadata for the documents in this folder are opened in the Information Agent. When one of the documents is selected, the Context Palettes for the document are displayed. To browse the documents using Reuters as a context, the user finds the icon representing the Reuters Agency, right-clicks the icon and hits "Copy." The user hovers over any of the results showing the documents metadata in the Information Agent and selects the icon indicating the Smart Lens. A Smart Lens window is displayed showing information on the results of the relational query. The number of items found on Reuters that are relevant to the document is displayed, in addition to information such as the most recently posted item. In addition, a preview control is displayed to allow the user to preview the results in place. The user is able to choose to click on the results to open an Agent representing the new, relational query. If done, the context for the first object in the results is displayed using the Context Palettes.

Notify by Email, Voice or Pager When There is Breaking News That Relates to Anything on XML Technology and Which Relates to This Document
Create a new Smart Agent using the "Breaking News" context and using the "XML" category as a category filter. Drag and drop the icon representing this document to the Agent. This creates a new Smart Agent with an appropriate title. Go to the "Options" menu in the Information Agent and enter the proper information in the notification section (your email address, pager number, telephone number, etc.). Right-click the Smart Agent and select "Notify."

2. Business Problem

Today's Web, John Head-Master works at FastServe, a marketing consulting service company in San Diego. Everyday, he comes in to work and fires up his Web browser. On this day, he decides to browse the corporate Web to see if he can discover new and interesting information. The browser home page is set (using an Enterprise Information Portal) to the corporate home page. The corporate home page has links for the home pages for different divisions within the company. John navigates to these links and from there, keeps clicking links. After a while, he gets frustrated because he knows that there are more sources of information that he cannot navigate to, only because he does not know what paths to take. Eventually, he gives up.

Information Nervous System. John fires up his Information Agent (semantic browser). This opens the home Agent. On the page, he sees a list of knowledge links corresponding to products, product groups, reports, corporate events, online courses, and video presentations. He hovers over the "product groups" link. Automatically, a balloon popup appears indicating the number of product groups and other data about the link. He then opens the link. A list of product group objects is then displayed with a customizable look or "skin." He then hovers his mouse over the first one. A popup menu immediately appears over the link with the actions: "Show Members," "List Similar Product Groups," and "Subscribe to Group Events." He then clicks on "Subscribe to Group Events" and he will now be notified by email (via the Enterprise Information Agent) about all events that relate to this product group. He then clicks "Show Members." This then opens a new "Knowledge Page" with icons corresponding to people. He then hovers over the icon for Susan Group-Lender. A balloon pop-up then appears showing information on Susan. A right-click menu then appears with the actions, "Reports To," "List Direct Reports," "Member Of," "Authored Documents," and "Recently Attended Meetings." John then selects "Recently Attended Meetings." This opens up a new knowledge page with one meeting object. John then hovers over this and continues browsing.

At some point, John decides to search for a co-worker he met the previous day. He then types in "Wilbur Jones." This then returns a person object corresponding to Wilbur. John then continues to browse using Wilbur as an Information Knowledge Pivot.

Eventually, John realizes that Wilbur does not seem to have the information he (John) needs. John then types the following query into the search box on his Information Agent: "List all online courses and documents that relate to the upcoming 2002 sales meeting." The Information Agent (via the Email Agent) then returns a list of actionable online courses and documents that conform to the knowledge query.

b. Knowledge-Driven Customer Relationship Management

Customer Touchpoints. AnySoft is a software manufacturer with 50 products in 100 different languages. They employ their web-site (anysoft.com) to provide up-to-date information to their customers. However, customers have complained that their Web site is very hard to navigate and that they find it very hard to find information on products and to subscribe for notifications.
By deploying an Information Nervous System based on an embodiment of the present invention, AnySoft has deployed an Information Nervous System that co-exists with their existing Web site. The Information Agent is accessible from the home page and from the search bar. Customers now have a much more intuitive way of navigating the Web site for products, relevant white papers, announcements, press releases, corporate events, etc. Customers can now issue natural language queries that return self-navigable and actionable knowledge objects. This feature alone gives customers access to knowledge at their fingertips. Customers can also now use natural language to navigate the AnySoft.com Web site from their handheld devices.

Customer Feedback and Tracking. Comp-Mart is a reseller of computer peripherals with multiple distribution channels. The Company gets customer feedback from its Web site, its call center, its direct sales force, its telemarketing agents, etc. The feedback comes in as documents and email. The Company has identified a problem wherein customer feedback does not get properly routed around the Company to the people that need the information. Employees in product development have complained to management that they find it hard to integrate customer feedback into the product development process because they don’t know where to find the information and because critical knowledge is not shared within the organization.

With an Information Nervous System in place, email that contains customer feedback now gets semantically integrated into the Company’s Semantic Environment. The KIS of the present invention automatically adds semantic links between customer feedback email and semantic objects like documents, projects, and employees that work on the germane products. Customer feedback intelligently bubbles up in the right places in the knowledge space. The Email Agent sends out periodic notifications to people that are likely to be interested in reading customer feedback email.

Also, with the Information Nervous System, the customer becomes an Information Knowledge Pivot. This makes it much quicker and easier to act on customer feedback and to track customer-related knowledge across the organization. The Information Nervous System automatically annotates the customer object with relevant email messages, documents, similar customers, etc. This way, links to the customer can be forwarded via email and co-workers can navigate relevant information from there. The customer object can be searched for, can be browsed, etc.

Knowledge-Driven Direct-Sales/Field-Servic

Marsha Mindset is a customer service agent for JustInTime Support Services, a computer service firm in Kansas City, Mo. Marsha visits customers around the Kansas City metro area, and always takes her wireless PDA so she can send email to the support headquarters anytime she is in difficulty. JustInTime recently deployed the KIS and the Email Agent. Now, whenever she has support questions, Marsha can send email to the Email Agent and ask it questions in natural language. The Email Agent replies to her email with direct answers or with “knowledge links” that allows Marsha to instantly access relevant support email, documents, or people that she could then email or call up on the phone. The JustInTime Direct Sales force also uses the technology of the present invention when in the field selling solutions to customers. The sales representatives also carry wireless PDAs and can issue requests to the Email Agent.

Corporate Training, Knowledge Transfer, and Sharing. WaveGen is a biotech company providing “managed care” solutions to doctors around the United States. The company recently deployed the Saba Learning Management System platform for training its employees (especially its sales reps). This reduces travel costs and enables the Company’s sales-force to be better prepared to serve physicians in different healthcare regions in the country. It also assists the Company’s researchers to be regularly informed of recent discoveries in the biotech research community.

The Company also has other software assets in place that hold valuable sources of knowledge. It has deployed content management solutions that host documents and media files, Microsoft Exchange for email, and collaboration software for online conferences. However, the Company has noticed that knowledge transfer is not very effective because it is not integrated across all these solutions. Sales representatives have indicated that they do not have the tools to discover important sources of knowledge within and outside the organization to assist them in pitching the Company’s products to doctors. Enterprise Information Portals are currently used to inform the sales force of upcoming online courses and of important events. However, the sales reps complain that a lot of knowledge (stored in email, documents, etc.) is not brought to their attention because no one knows who else might need them.

In addition, the sales representatives use Microsoft Outlook to add appointments to their calendars for upcoming doctor visits. However, they complain that they only get reminders for the appointments, and that a lot of information that could help them sell products more effectively is not made available to them automatically, ahead of their doctors’ appointments.

WaveGen recently deployed an Information Agent based on technology from the present invention. The company deployed the KIS and the Email Agent to facilitate intelligent information connections and routing to help their sales and research teams make better decisions to serve customers and improve the Company’s products. Using the Information Agent, the sales force has instant access not just to documents but to “knowledge objects” that are more directly tied to their task at hand. For instance, the sales representatives now have an Agent with “Doctor Jones” as an XML object. This is not a document or a Web page. Rather, it is a semantic representation of the customer. A sales representative can then see semantic links like “Recent Email Messages”, “Relevant Documents,” “Properties,” “Important Dates,” “Relevant upcoming online courses,” etc. This way, the customer becomes the pivot with which the sales agent is navigating the internal Web. These links might generate results from file-shares, Email stores, Microsoft Exchange, etc. But rather than searching or navigating for these knowledge sources as islands, the sales representative can discover new knowledge based on semantic relationships as they relate to the sales representative’s task.

This way, the sales representative can have much more powerful knowledge at the sales representative’s fingertips, thereby enabling much better customer service. And this knowledge emanates from co-workers, documents that were published by other sales agents, email sent on distribution lists that might not be known to exist, etc. The KIS does the smart thing by automatically making semantic connections from all these disparate sources. The sales representative can
then email this “page” to a co-worker. This then becomes a very powerful form of knowledge sharing because the co-worker can then navigate the Information Agent using the same “Dr. Jones” pivot.

The Email Agent also allows the sales representative to issue knowledge queries via natural language. The query results are derived from the inference engine and could be based on knowledge that was deduced from existing knowledge. A powerful feature of the Information Nervous System of the present is that knowledge transfer, sharing, discovery all happen automatically based on the Semantic Network.

3. Situations

Joe Knowledge-Worker starts the Information Agent (the XML-based semantic browser of the present invention). When he logs in, he is prompted with a dialog box indicating that there are new Agents available on the semantic intranet. He then sees a list of Agents from within and outside the organization that may include the following:

- Documents.Technology.All
- Documents.Marketing.All
- People.Divisions.Sales.All
- OnlineCourses.Sales.101
- OnlineCourses.Technology.XML.101
- Meetings.ThisWeek.All
- Meetings.LastWeek.All
- Newsgroups.Microsoft.PublicSoap
- Email.Mine.All
- Email.Mine.ProjectX.All
- Events.Technology.Wireless.All
- Reports.Gartner.Software.All
- Reports.IDC.All
- Videos.ExecutivePresentations.All

He then selects Meetings.ThisWeek.All. The Information Agent then displays a list of objects that represents meetings that he attended this week. This information comes from Microsoft Exchange but this is not exposed to him. Joe then hovers over a link for the first meeting object. A balloon pop-up is then displayed indicating that a training course was just made available on the intranet. The balloon also indicates that there is a new report on IDC that might be relevant to Joe. In addition to the balloon, a pop-up menu is displayed to the right of the object. This menu has the following verbs:

- List participants
- List possible replacement participants
- Show Related Objects>
- On News.Reuters.MarketsForecasts.All
- On Documents.Technology.All
- On Events.Corporate.Today.All
- Subscribe for follow-up

Joe then selects “Subscribe for follow-up.” This contacts the Meeting Follow-up Agent on the server. This Agent then sends periodic updates of relevant information to the participants of the meeting. This could be done either through the browser or through email. Joe then selects related objects on Events.Corporate.Today.All. This then displays a list of event information objects. Joe then hovers over the first object and a pop-up menu gets displayed. Joe then selects “Add to calendar” and the event is added to his calendar. Joe then decides that he wants to find all industry events that relate to the corporate event. He then drags the object to the Agent Events.Technology.All and releases his mouse. When the mouse is released, the browser then loads information objects from Events.Technology.All (across web-sites and other islands) and which are related to the corporate event the object of which he dragged.

The next week Joe gets email from the Email Agent. In the email, the Agent informs Joe that it has noticed that everyone that added the event to his or her calendar also watched a corporate training video from the corporate media server. The email contains an XML link, which takes Joe back into the Information Agent. The browser then displays the metadata for the video. One of the items on the pop-up is “Watch Video.” Joe then selects it and watches the video.

The next time Joe logs in to his workstation, he notices that there are new Agents. He then subscribes to Books.Ebay.Computers.All and adds it to his My Agent list. Automatically, an embodiment of the present invention adds this Agent into Joe’s Semantic Environment. The Information Agent performs implicit queries and provides recommendations (ranked by relevance and time-sensitivity) that include this Agent. He then clicks on this Agent and semantic information objects (representing books) are displayed in the Results Pane. When he hovers over one of the objects, a pop-up balloon is immediately displayed, alerting him to the fact that there is a related industry conference being hosted by the author of the book. When he clicks the pop-up link, the event object is loaded in the browser, complete with verbs that allow him to add the event to his calendar (either Microsoft Outlook or an Internet-based calendar like the MSN Calendar (accessible via Microsoft’s HailStorm Web services), AOL Calendar, etc.)

Explanation of the Scenario. This scenario shows how with the present invention, knowledge-workers are able to obtain access to “federated knowledge.” In this example, Joe’s company has “imported” knowledge Agents from Gartner, IDC, Reuters, Ebay, etc. into its knowledge space. As such, these Agents automatically add knowledge into the company’s Semantic Network. The scenario also showed how Joe was able to get an “object model” view of the entire organization’s knowledge-space via intuitively named Smart Agents. Joe was able to use these Agents to “enter” the Semantic Environment, and then navigate his way from there. All the information objects were delivered in real-time and were actionable (with relevant verbs that were displayed in place). This way, Joe did not have to care about what information island the objects were coming from, or what applications generated them.

The scenario also shows how Joe was able to discover not just new information but also new Agents. And the scenario shows knowledge collaboration in action—via collaborative filtering—wherein the Information Agent gave recommendations to Joe based on what it noticed others in the enterprise were doing.

Lastly, the scenario illustrates how time-sensitive information is automatically brought to the user’s attention at the point of context where it makes sense. The Email Agent automatically connected the book from Ebay with the upcoming industry event, inferred and assigned a relevance and time-sensitivity ranking to the event, and decided that the event was critical enough to warrant displaying the information immediately via an alert in the semantic browser.
b. Peer-to-Peer Knowledge Sharing and Capture

Nancy Hard-worker works at a Fortune 500 company with 40,000 employees. She subscribes to a variety of Web sites and has information forwarded to her by email from friends and co-workers. She just got a bunch of documents from someone at a partner company and she would like to share the information within the organization. She sends the documents to all the distribution lists of which she is a member. The Enterprise Information Agent is a member of these lists also (the Agent adds itself to all public distribution lists when the server is installed). When the Agent receives the information, it classifies it and adds it to the Semantic Network. The Inference Engine then picks up the information.

Several thousand co-workers are not members of any of the distribution lists to which Nancy forwarded the documents. However, they all use the Integrator and all of them have subscribed to the Email.Public.All Agent. While they browse other related parts of the knowledge-web, a balloon popup gets displayed indicating that there is new and relevant email on the Email.Public.All Agent. The co-workers then open up the Agent and the email object is displayed. One of the menu items on the email item is “Show distribution lists to which message was forwarded.” The co-workers then select this and the distribution list information objects are then displayed in the browser. The co-worker then hovers over the distribution list and a pop-up menu item gets displayed. The first item is “Show Members.” The second is “Join.” The co-workers then join the distribution list.

Explanation of the Scenario. This scenario illustrates how information was published, shared and captured via email and how, by use of the Semantic Network, other co-workers found out about this information (and about distribution lists the existence of which they were not aware) from different but related “knowledge angles.” The scenario shows peer-to-peer knowledge sharing in a way that is completely seamless and does not require users to public information to repositories, or to classify information themselves. With certain embodiments of the present invention, everything just happens automatically (in the background) and the knowledge gets bubbled up in relevant places.

A. Additional Illustrative Scenarios

The following scenarios help to explain the utility and operation of the system, and will thereby make the rest of the detailed description easier to follow and understand.

1. Patent Examiner Prior Art Search Tool

Largely because of PTO fee diversion, there is a great deal of pressure on U.S. Patent Examiners to conduct a robust prior art search in very little time. And, while the research tools available to Examiners have improved dramatically in the last several years, those tools still have many shortcomings. Among the shortcomings are that most of the research tools are text based, rather than meaning based. So, for example, the search tool on the PTO website will search for particular words in particular fields in a document. Similarly, the advanced search tool on Google enables the Examiner to locate documents with particular words, or particular strings of words, or documents without a particular word or words. However, in each case, the search engine does not allow the Examiner to locate documents on the basis of meaning. So, for example, if there is a relevant reference that teaches essentially the same idea, but uses completely different words (e.g., a synonym, or worse yet, a synonymous phrase) than those in the query, the reference, even though perhaps anticipating, may well not be discovered. Even if the Examiner could spare the time to imagine and search every possible synonym, or even synonymous phrase to the key words critical to the invention, it could still overlook references because sometimes the same idea can be expressed without using any of the same words at all, and sometimes the synonymous idea is not neatly compressed into a phrase, but distributed over several sentences or paragraphs.

The reason for this is that words do not denote or connote meaning one to one as, for example, numerals tend to do. Put differently, certain meanings can be denoted or connoted by several different words or an essentially infinite combination of words, and, conversely, certain words or combinations of words can denote or connote several different meanings. Despite this infinite many-to-many network of possibilities human beings can isolate (because of context, experience, reasoning, inference, deduction, judgment, learning and the like) isolate probable meanings, at least tolerably effectively most of the time. The current prior art computer-automated search tools (e.g., the PTO website, or Google, or Lexis) cannot. The presently preferred embodiment of my invention bridges this gap considerably because it can search on the basis of meaning.

For example, using some of the search functions of the preferred embodiment of the present invention, the Examiner could conduct a search, and with no additional effort or time as presently invested, obtain search results relevant to patentability even if they did not contain a single word in common with the key words chosen by the Examiner. Therefore, the system would obtain results relevant to the Examiner’s task that would not ordinarily be located by prior search systems because it can locate references on the basis of meaning.

Also on the basis of meaning, it can exclude irrelevant references, even if they share a key word or words in common with the search request. In other words, one problem in prior art research is the problem of a false positive; results that the search engine “thought” were relevant merely because they had a key word in common, but that were in fact totally irrelevant because the key word, upon closer inspection in context, actually denoted or connoted an irrelevant idea. Therefore, the Examiner must search for the needle in the haystack, which is a waste of time.

In contrast, using some of the search functions of the preferred embodiment of the present invention, the density of relevant search results increases dramatically, because the system is “intelligent” enough to omit search results that, despite the common key words, are not relevant. Of course, it is not perfect in this respect any more than human beings are perfect in this respect. But, it is much more effective at screening irrelevant results than present systems, and in this respect resembles in function or in practice an intelligent research assistant than a mere keyword based search engine. Thus, using the system, the Examiner can complete a much better search in much less time. The specific mechanics of using the system this way, in one example, would work as follows:

Imagine the Examiner is assigned to examine an application directed to computer software for a more accurate method of interpreting magnetic resonance data and thereby generating more accurate diagnostic images. To search for relevant prior art using the search functions of the preferred embodiment of the present invention, the Examiner would:

a. Using the Create Entity wizard, create a “Topic” entity with the relevant categories in the various contexts in
which “Magnetic Resonance Imaging” occurs. For example, FIG. 44 shows where “Magnetic Resonance Imaging” occurs in a Pharmaceuticals taxonomy. Notice that there are several contexts in which the category occurs. Add the relevant categories to the entity and apply the “OR” operation. Essentially, this amounts to defining the entity “Magnetic Resonance Imaging” (as it relates to YOUR specific task) as being equivalent to all the occurrences of Magnetic Resonance Imaging in the right contexts—based on the patent application being examined.

1185 b. Name the new entity “Magnetic Resonance Imaging” and perhaps “imaging” and “diagnostic” or some variations and combinations of the same.

1186 c. Drag and drop the “Magnetic Resonance Imaging” Topic entity to the Dossier (special agent or default knowledge request) icon in the desired profile (the profile is preferably configured to include the “Patent Database” knowledge community). This launches a new Dossier request/agent that displays each special agent (context template). Each special agent is displayed with the right default predicate as follows:

1187 All Bets on Magnetic Resonance Imaging

1188 Best Bets on Magnetic Resonance Imaging

1189 Breaking News on Magnetic Resonance Imaging

1190 Headlines on Magnetic Resonance Imaging

1191 Random Bets on Magnetic Resonance Imaging

1192 Experts in Magnetic Resonance Imaging

1193 Newsmakers in Magnetic Resonance Imaging

1194 Interest Group in Magnetic Resonance Imaging

1195 Conversations on Magnetic Resonance Imaging

1196 Annotations on Magnetic Resonance Imaging

1197 Annotated Items on Magnetic Resonance Imaging

1198 Upcoming Events on Magnetic Resonance Imaging

1199 Popular Items on Magnetic Resonance Imaging

1200 Classics on Magnetic Resonance Imaging

1201 d. Alternatively, the request can be created by using the Create Request Wizard. To do this, select the Dossier context template and select the “Patent Database” knowledge community as the knowledge source for the request. Alternatively, you can configure the profile to include the “Patents Database” knowledge community and simply use the selected profile for the new request. Hit Next—the wizard intelligently suggests a name for the request based on the semantics of the request. The wizard also selects the right default predicate based on the semantics of the “Magnetic Resonance Imaging” “Topic” entity. Because the wizard knows the entity is a “Topic,” it selects the right entities that make sense in the right contexts. Hit Finish. The wizard compiles the query, sends the SQMI to the KISes in the selected profile, and then displays the results.

1202 In the foregoing example, the results could be drawn, ultimately, from any source. Preferably, some of the results would have originated on the Web, some on the PTO intranet, some on other perhaps proprietary extranets. Regardless of the scope or origin of the original documents, by use of the system they have been automatically processed, and automatically “read” and “understood” by the system, so that when the Examiner’s query was initiated, and also “read” and “understood” semantically, and by context, the system locates all relevant, and only relevant results. Again, not perfectly, but radically more accurately than in any prior systems. Note also that the system does not depend on any manual tagging or categorization of the documents in advance. While that would also add in accuracy, it is so labor intensive as to utterly eclipse the advantages of online research in the first place, and is perfectly impractical given the rate of increase of new documents.

1203 In this scenario, the Examiner may also wish to use additional features of the preferred embodiment of the invention. For example, the Examiner may wish to consult experts within the PTO, or literature by experts outside the PTO, as follows (note that Experts in Magnetic Resonance Imaging would be included in the Dossier on Magnetic Resonance Imaging; however, the examiner might want to create a separate request for Experts in order to track it separately, save it as a “request document,” email it to colleagues, etc.). Find all Experts in Magnetic Resonance Imaging:

1204 a. Follow steps 1-4 above.

1205 b. Drag and drop the “Magnetic Resonance Imaging” entity to the Experts (special agent or default knowledge request) icon in the desired profile. This automatically launches a new request/agent appropriately titled “Experts in Magnetic Resonance Imaging.” The semantic browser selects the right default predicate “in” because it “knows” the entity is a “Topic” entity and the context template is a “People” template (Experts). As such, the default predicate is selected based on the intersection of these two arguments (“in”) since this is what makes sense.

1206 2. BioTech Company Research Scenario

1207 Biotech companies are research intensive, not only in laboratory research, but in research of the results of research by others, both within and outside of their own companies. Unfortunately, the research tools available to such companies have shortcomings. Proprietary services provide context-sensitive and useful results, but those services themselves have inferior tools, and thus rely heavily on indexing and human effort, and subscriptions to expensive specialized journals, and as consequence are very expensive and not as accurate as the present system. On the other hand, biotech researchers can search inexpensively using Google™, but it shares all the key word based limitations described above.

1208 In contrast, using the search features of the preferred embodiment of the present invention, a biotech researcher could more efficiently locate more relevant results. Specifically, the researcher might use the system as follows. For example, if some researchers wanted to find Headlines on Genomics and Anatomy written by anyone in Marketing or Research, they would do that as follows:

1209 a. Using the wizard, launch an information-type request/agent for distribution lists with the keywords “Marketing Research”.

1210 b. Select the Marketing distribution list result and click “Save as Entity”—this saves the object as a “Team” entity (because the semantic browser “knows” the original object is a distribution list—as such, a “Team” entity makes sense in this context).

1211 c. Select the Research distribution list result and click “Save as Entity”—this saves the object as a “Team” entity (because the semantic browser “knows” the original object is a distribution list).

1212 d. Using the Create Entity Wizard, create a new “Team” entity and select the “Marketing” and “Research” team entities as members. Name the new entity “Marketing or Research”.

1213 e. Using the Create Request Wizard, select the Headlines context template, and then select the “Marketing or
Research” entity as a filter. Also, select the Genomics category and the Anatomy category. Next, select the “AND” operator. Hit Next—the wizard intelligently suggests a name for the request based on the semantics of the request. The wizard also selects the right default predicates based on the semantics of the “Marketing or Research” team entity (“by anyone in”). Because the wizard knows the entity is a “Team,” it selects “by anyone in” by default since this makes sense. Hit Finish. The wizard compiles the query, sends the SQML to the KIS engines selected in the selected profile, and then displays the results.

In addition, the researchers may wish to Find all Experts in Marketing or Research:

a. Follow steps 1-4 above.

b. Drag and drop the “Marketing or Research” entity to the Experts (special agent or default knowledge request) icon in the desired profile. This launches a new request/agent appropriately titled “Experts in Marketing or Research.” The semantic browser selects the right default predicate “in” because it “knows” the entity is a “Team” entity and the context template is a “People” template (Experts). As such, the default predicate is selected based on the intersection of these two arguments (“in”) since this is what makes sense.

If the researchers expect to need to return to this research, or to supplement it, or to later analyze the results, they may wish to Open a Dossier on Marketing or Research, as follows:

a. Follow steps 1-4 above.

b. Drag and drop the “Marketing or Research” entity to the Dossier (special agent or default knowledge request) icon in the desired profile. This launches a new Dossier request/agent that displays each special agent (context template). Each special agent is displayed with the right default predicate as follows:

All Bets by anyone in Marketing or Research
Best Bets by anyone in Marketing or Research
Breaking News by anyone in Marketing or Research
Headlines by anyone in Marketing or Research
Random Bets by anyone in Marketing or Research
Experts in Marketing or Research
Newsmakers in Marketing or Research
Interest Group in Marketing or Research
Conversations involving anyone in Marketing or Research
Annotations by anyone in Marketing or Research
Annotated Items by anyone in Marketing or Research
Upcoming Events by anyone in Marketing or Research
Popular Items by anyone in Marketing or Research
Classics by anyone in Marketing or Research

The researchers may be interested in Finding “Breaking News on my Competitors”, and would do so as follows:

a. For each competitor, create a new “competitor” entity (under “companies”) using the Create Entity Wizard. Select the right filters as needed. For instance, a competitor with a well-known English name—like “Groove” should have an entity that includes categories in which the company does business and also the keyword.

b. Using the Create Entity Wizard, create a portfolio (entity collection) and add all the competitor entities you created in step a. Name the entity collection “My Competitors.”

c. Using the Create Request Wizard, select the Breaking News context template and add the portfolio (entity collection) you created in step b. as a filter. Keep the default predicate selection. Hit “Next”—the wizard intelligently suggests a name for the request using the default predicate (“Breaking News on My Competitors”). Hit Finish. The wizard launches a new request/agent named “Breaking News on My Competitors.”

In addition, the researchers may wish to be kept apprised. They could instruct the system to alert them on “Breaking News on Our Competitors”, as follows:

a. Create the “Breaking News on My Competitors” request as described above.

b. Add the request to the request watch list. The semantic browser will now display a watch pane (e.g., a ticker) showing “Breaking News on My Competitors.” Using the Notification Manager (NM), you can also indicate that the semantic browser send alerts via email, instant messaging, text messaging, etc. when there are new results from the request/agent.

In addition, the researchers may wish to keep records of competitors for future reference, and to have them constantly updated. The system will create and update such records, by the researchers instructing the system to Show a collection of Dossiers on each of our competitors, as follows:

a. Create entities for each of your competitors as described in 4a. above.

b. For each competitor entity, create a new Dossier on that competitor by dragging the entity to the Dossier icon for the desired profile—this creates a Dossier on the competitor.

c. Using the Create Request Wizard, create a new request collection (blender) and add each of the Dossier requests created in step b. above to the collection (you can also drag and drop requests to the collection after it has been created in order to further populate the collection). Hit Next—the wizard intelligently suggests a name for the request collection. Hit Finish. The wizard launches a request collection that contains the individual Dossiers. You can then add the request collection as a favorite and open it everyday to get rich, contextual competitive intelligence.

d. The researchers may wish to review a particular dossier, and can do so by instructing the system to Show a Dossier on the CEO (e.g., named John Smith):

a. Using the wizard, launch an information-type request/agent for People with the keywords “John Smith”.

b. Select the result and click “Save as Entity”—this saves the object as a “Person” entity (because the semantic browser “knows” the original object is a person—as such, a “Person” entity makes sense in this context).

c. Using the Create Entity Wizard, select the Dossier context template, and then select the “John Smith” entity as a filter. Hit Next—the wizard intelligently suggests a name for the request based on the semantics of the request. The wizard also selects the right default predicates based on the semantics of the “John Smith” person entity. Hit Finish. The wizard compiles the query, sends the SQML to the KISes in
the selected profile, and then displays the results (as subqueries/agents) as follows:

[1249] All Bets by John Smith
[1250] Best Bets by John Smith
[1251] Breaking News by John Smith
[1252] Headlines by John Smith
[1253] Random Bets by John Smith
[1254] Experts like John Smith (this returns Experts that have expertise on the same categories as those in which John Smith has expertise)
[1255] News-makers like John Smith (this returns News-makers that have recently “made news” in the same categories as those in which John Smith has recently “made news”)
[1256] Interest Group like John Smith (this returns the people that have shown an interest in the same categories as those in which John Smith has shown interest—within a time-window (2-3 months in the preferred embodiment))
[1257] Conversations involving John Smith
[1258] Annotations by John Smith
[1259] Annotated Items by John Smith
[1260] Upcoming Events by John Smith
[1261] Popular Items by John Smith
[1262] Classics by John Smith
[1263] The foregoing scenarios illustrate the operation of the system. The system itself is described in greater detail below.

B. Subject Matter for the Presently Preferred Embodiment of the Information Nervous System

[1264] Several improvements, enhancements and variations have been developed since the filing of my co-pending parent application and prior provisional applications referenced above. Some of these are improvements on, or only clarifications of, features previously included in the parent application, and some are new features of the system altogether. These are listed and described below. They are not arranged in order of importance, or in any particular order. While the preferred embodiment of the present invention would allow the user to use any or all of these features and improvements described below, alone or in combination, no single feature is necessary to the practice of the invention, nor any particular combination of features.
[1265] Also, in this application, reference is made to the same terms as are defined in my parent application Ser. No. 10/179,651, and the Description throughout this application is intended to be read in conjunction with the definitions, terminology, nomenclature and Figures of my parent application except where the context of this application clearly indicates to the contrary.

[1266] 1. Smart Selection Lens Overview
[1267] The Smart Selection Lens is similar to the Smart Lens feature of the Information Nervous System information medium. In this case, the user can select text within the object and the lens will be applied using the selected text as the object (dynamically generating new “images” as the selection changes). This way, the user can “lens” over a configurable subset of the object metadata, as opposed to being constrained to “lens” over either the entire object or nothing at all. This feature is similar to a selection cursor/verb overloaded with context. For example, the user can select a piece of text in the Presenter and hit the “Paste as Lens” icon over the object in which the text appears. The Presenter will then pass the text to the client runtime component (e.g., an ActiveX object) with a method call like:

[1268] bstrSRML = GetSRMLForText(bstrText);
[1269] This call then returns a temporary SRML buffer that encapsulates the argument text. The Presenter will then call a method like:

[1270] bstrSQLML = GetQueryForSmartLensOnObject (bstrSRMLObject);
[1271] This method gets the SQLML from the clipboard, takes the argument SRML for the object, and dynamically creates new SQLML that includes the resource in the SRML as a link in the SQLML (with the default predicate “relevant to”). The method then returns the new SQLML. The Presenter then calls the method:

[1272] ProcessSemanticQuery(bstrSQLML);
[1273] This method passes the generated lens SQLML and then retrieves the number of items in the results and the SRML results, preferably asynchronously. For details on this call, see the specification “Information Nervous System Semantic Runtime OCX.” The Presenter then displays a preview window (or the equivalent, based on the current skin) with something like:

[1274] [Lens Agent Title]
[1275] Found 23 items
[1276] [PREVIEW OBJECT]
[1277] [PREVIEW WINDOW CONTROLS]
[1278] where the “Lens Agent Title” is the title of the agent on the clipboard. For details of the preview window (and the preview window controls), please refer to my parent application Ser. No. 10/179,651.

[1279] In the preferred embodiment, the preview window will:

[1280] Disappear after a timer expires (maybe 500 ms)—on mouse move, the timer is preferably reset (this will avoid flashing the window when the user moves the mouse around the same area).
[1281] Fade out slowly (eventually).
[1282] The preferred embodiment also has the following features:

[1283] 1. One selection range per object but multiple selections per results-set is the best option. Otherwise, the system would result in a confusing user experience and complex UI to show lens icons per selection per object (as opposed to per object).
[1284] 2. Outstanding lens query requests (which are regular SQLML queries, albeit with SQLML dynamically generated consistent with the agent lens) should be cancelled when the Presenter no longer needs them (e.g., if the Presenter is navigating to a new page, or if we are requesting new lens info for an object). In any case, such cancellation is not critical from a performance (or bandwidth) standpoint because lens queries will likely only ask for a few objects at a time. Even if the queries are not cancelled, the Presenter can ignore the results. Regardless, because the Presenter also has to deal with stale results, dropping them on the floor—the Presenter will have to do this anyway (whether or not lens queries are also cancelled). There will be a window of delay between when the Presenter issues a cancel request and when the cancellation actually is complete. Because some results can trickle in during this time, they need to be discarded. Thus, the preferred embodiment has asynchronous cancellation implementations—the software component has been designed to always be prepared to ignore bad or stale results.
3. The Presenter preferably has both icons (indicating the current lens request state) and tool-tips: When the user hovers over or clicks on an object, the Presenter can put up a tool-tip with the words, “Requesting Lens Info” (or words to that effect). When the info comes back, hovering will show the “Found 23 Objects” tip and clicking will show the results. This interstitial tool tip can then be transitioned to the preview window if it is still up when the results arrive.

In addition, note that the smart selection lens, like the smart lens, can be applied to objects other than textual metadata. For instance, the Smart Selection Lens can be applied to images, video, a section of an audio stream, or other metadata. In these cases, the Presenter would return the appropriate SRML consistent with the data type and the “selection region.” This region could be an area of an image, or video, a time span in an audio stream, etc. The rest of the smart lens functionality would apply as described above, with the appropriate SQML being generated based on the SRML (which in turn is based on the schema for the data type under the lens).

2. Pasting Person Objects Overview

The Information Nervous System (which, again, is one of our current shorthand names for certain aspects of our presently preferred embodiments) also supports the drag and drop or copy and paste of ‘Person’ objects (People, Users, Customers, etc.). There are at least two scenarios to illustrate the operation of the preferred embodiment in this case:

1. Pasting a Person object on a smart request representing a Knowledge community (or Agency) from whence the Person came. In this case, the server’s semantic query processor merely resolves the SQML from the client using the Person as the argument. For instance, if the user pastes (or drags and drops) a person ‘Joe’ on top of a smart request ‘Headlines on Reuters™’, the client will create a new smart request using the additional argument. The Reuters™ Information Nervous System Web service will then resolve this request by returning all headlines published or annotated by ‘Joe.’ In this case, the server will essentially apply the proper default predicate (‘published or annotated by’) — that makes sense for the scenario.

2. Pasting a Person object on a smart request representing a Knowledge community (or Agency) from whence the Person did not come. In this case, because the Person object is not in the semantic network of the destination Knowledge community (on its SMS), the server’s semantic query processor would not be able to make sense of the Person argument. As such, the server must resolve the Person argument, in a different way, such as, for example, using the categories on which the person is an expert (in the preferred embodiment) or a newsman. For instance, taking the above example, if the user pastes (or drags and drops) a person ‘Joe’ on top of a smart request ‘Headlines on Reuters™’ and Joe is not a person on the Reuters™ Knowledge community, the Reuters™ Web service (in the preferred embodiment) must return Headlines that are “relevant to Joe’s expertise.” This embodiment would then require that the client take a two-pass approach before sending the SQML to the destination Web service. First, it must ask the Knowledge community that the person belongs to for “representative data (SRML)” that represents the person’s expertise. The Web service resolves this request by:

a. Querying the Knowledge community (e.g., Reuters™) on which the person object is pasted or dropped for that community’s semantic domain information which comprises and/or represents that community’s specific taxonomy and ontology. Note that there could be several semantic domains.

b. Querying the Knowledge community from whence the person object came for that person object’s semantic domain information.

c. If the semantic domains are identical or if there is at least one common semantic domain, the client queries the Knowledge community from whence the person came for the person’s categories of expertise. The client then constructs SQML with these categories as arguments and passes this SQML to the Knowledge community on which the person was pasted or dropped.

If the semantic domains are not identical or there is not at least one common semantic domain, the client queries the Knowledge community from whence the person came for several objects that belong to categories on which the person is an expert. In the preferred embodiment, the implementation should pick a high enough number of objects that accurately represent the categories of expertise (this number is preferably picked based on experimentation). The reason for picking objects in this case is that the destination Web service will not understand the categories of the Knowledge community from whence the person came and as such will not be able to map them to its own categories. Alternatively, a category mapper can be employed (via a centralized Web service on the Internet) that maps categories between different Knowledge Communities. In this case, the destination Knowledge community will always be passed categories as part of the SQML, even though it does not understand those categories—the Knowledge community will then map these categories to internal categories using the category mapper Web service. The category mapper Web service will have methods for resolving categories as well as methods for publishing category mappings.

3. Saving and Sharing Smart Requests Overview

Users of the Information Nervous System semantic browser (the Information Agent or Librarian) will also be able to save smart requests to disk, email them as an attachment, or share them via Instant Messenger (also as an attachment) or other means. The client application will expose methods to save a smart request as a sharable document. The client application will also expose methods to share a smart request document as an attachment in email or Instant Messenger.

A sharable smart request document is a binary document that encapsulates SQML (via a secure stream in the binary format). It provides a safe, serialized representation of a semantic query that, among other features, can protect the integrity and help protect the intellectual property of the specification. For example, the query itself may embody trade secrets of the researcher’s employer, which, if exposed, could enable a competitor to reverse engineer critical competitive information to the detriment of the company. The protection can be accomplished in several ways, including by strongly encrypting the XML version of the semantic query (the SQML) or via a strong one-way hash. The sharable document has an extension (`.REQ`) that represents the request. An extension handler on the client operating system is installed to represent this extension. When a document with the extension is opened, the extension handler is invoked to open the document. The extension handler opens the document by extracting the SQML from the secure stream, and then creating a
smart request in the semantic namespace with the SQML. The handler then opens the smart request in the semantic namespace.

When a smart request in the semantic namespace is saved or if the user wants to send it as an email attachment, the client serializes the SQML representing the smart request in the binary .REQ format and saves it at the requested directory path or opens the email client with the .REQ document as an attachment. [1298]

Fig. 45 shows the binary document format that encapsulates the SQML buffer with the smart request and also illustrates how the extension handler opens the document. A similar model can also be employed for sharing results (via SRML). In this case, a binary document encapsulates the SRML, rather than the SQML as in the case above. In an exemplary embodiment, the operating system can be associated with the semantic browser application (Nervana Librarian) and the document. When the document is opened, the semantic query gets opened in the application.

Saving and sharing entities—the same process applies as above except with a .ENT extension to represent an entity. When an entity document is invoked, the Nervana Librarian opens the entity SQML in the browser.

Extension Property Sheet—this will create a temporary smart request or entity (depending on the kind of document) in the semantic environment and display the property sheet for a smart request or entity.

Extension Tool tips—this will display a helpful tool tip when the user hovers over a librarian document (a request, .REQ or an entity. .ENT).

4. Saving and Sharing Smart Snapshots Overview

The Information System Network also supports the sharing of what the inventory calls “Smart Snapshots.” A smart snapshot is a smart request frozen in time. This will enable a scenario where the user wants to share a smart request but not have it be “live.” For instance, by default, if the user shares the smart request “Breaking News on Reuters™ related to this document” with a colleague, the colleague will see the live results of the smart request (based on the “current time”). However, if the user wants to share “[Current] Breaking News on Reuters™ related to this document,” a smart snapshot will be employed.

A smart snapshot is the same as a smart request (it is also represented by an SQML query document) except that the “attributes” section of the SQML document contains attributes marking it as a snapshot (the flag QUERYATTRIBUTES_SNAPSHOT). The creation date/time of the SQML document is also stored in the SQML (as before—the SQML schema contains a field for the creation date/time).

When the user indicates that he/she wants to share the smart request, the user interface (the semantic browser, Information Agent, or Librarian) prompts him/her whether he/she wants to share the smart request (live) or a smart snapshot. If the user indicates a smart request, the process described above in Part 3 is employed. If the user indicates a smart snapshot, the binary document is populated with the edited SQML (containing the snapshot attribute) and the remainder of the process is followed as above.

When the recipient of the binary document receives it (by email, instant messaging, etc.), and opens it, the extension handler opens the document and adds an entry into the semantic namespace as a smart request (as described above). When the recipient opens the smart request, the client’s semantic query processor will send the processed SQML to the server’s XML web service (as previously described). The server’s semantic query processor then processes the SQML and honors the snapshot attribute by invoking the semantic query relative to the SQML creation date/time. As such, results will be relative to the original date/time, thereby honoring the intent of the sender.

Virtual Knowledge Communities

Virtual Knowledge Communities (agencies) refer to a feature of the Information Nervous System that allows the publisher of a knowledge community to publish a group of servers to appear as though they were one server. For instance, Reuters™ could have per-industry Reuters™ Knowledge Communities (for pharmaceuticals, oil and gas, manufacturing, financial services, etc.) but might also choose to expose one “Reuters™ knowledge community. To do this, Reuters™ will publish and announce the SQML for the virtual knowledge community (rather than the URL to the WSDL of the XML Web Service). The SQML will contain a blender (or collection) of the WSDLs of the actual Knowledge Communities. The semantic browser will then pick up the SQML and display an icon for the knowledge community (as though it were a single server). Any action on the knowledge community will be propagated to each server in the SQML. If the user does not have access for the action, the Web service call will fail accordingly, else the action will be performed (no different from if the user had manually created a blender containing the Knowledge Communities).

6. Implementing Time-Sensitive Semantic Queries

Semantic queries that are time-sensitive are preferentially implemented in an intelligent fashion to account for the rate of knowledge generation at the knowledge community (agency) in question. For instance, ‘Breaking News’ on a server that receives 10 documents per second is not the same as ‘Breaking News’ on a server that receives 10 documents per month. As such, the server-side semantic query processor would preferably adjust its time-sensitive semantic query handling according to the rate at which information accumulates at the server. To implement this, general rules of thumb could be used, for instance:

The most recent N objects where N is adjusted based on the number of new objects per minute.

All objects received in the last N minutes with a cap on the number of objects (i.e., min(cap, all objects received in the last N minutes)).

N can also be adjusted based on whether the query is a Headline or Breaking News. In the preferred embodiment, newsmaker queries is preferably implemented with the same time-sensitivity parameters as Headlines.

Text-to-Speech Skins Overview

Text-to-speech is implemented at the object level and at the request level. At the object level, the object skin runs a script to take the SRML of the object, interprets the SRML, and then passes select pieces of text (in the SRML fields) to a text-to-speech engine (e.g., using the Microsoft™ Windows™ Speech SDK) that generates voice output.

Fig. 46 shows a diagram illustrating text-to-speech object skin. When executed, the pipeline shown in Fig. 5 results in the following voice output:

Reading Email Message

Appropriate Delay

Message From Nosa Omoigui

Appropriate Delay

Message Sent to John Smith

Appropriate Delay
This example assumes a voice skin template as follows:

1. Reading Email Message
2. Appropriate Delay
3. Message From <message author name>
4. Appropriate Delay
5. Message Sent To <message to: recipient name>
6. Appropriate Delay
7. Message Copied To <message cc: recipient name>
8. Appropriate Delay
9. Message Subject Is <message subject text>
10. Appropriate Delay
11. Message Summary is <message body summary>
12. Appropriate Delay
13. [Optional] Message Body is <message body>

Other templates can also be used to render voice that is easily understandable and which conveys the semantics of the object type being rendered. Like the example shown above (which is for email), the implementation should use appropriate text-to-speech templates for all information object types, in order to capture the semantics of the object type.

At the request level, the semantic browser's presentation engine (the Presenter) loads a skin that takes the SRML for all the current objects being rendered (based on the user-selected cursor position) and then invokes the text-to-speech object skin for each object. This essentially repeats the text-to-speech action for each XML object being rendered, one after another.

Email Object (SRML)
Object Interpretation Engine (Object Skin)
Text-To-Speech Engine
From: Nosa Omoigui
To: John Smith
Cc: Joe Somebody
Subject: Web services
Summary: Web services are software building blocks used for distributed computing
Body: Web services . . .
Voice Output
Reading Email Message
Delay
Voice Output
Message From Nosa Omoigui
Delay
Voice Output
Message Sent To John Smith
Delay
Voice Output
Message Copied To Joe Somebody
Delay
Message Subject is Web services are software building blocks used for distributed computing
Voice Output
Delay
Voice Output
Message Summary is Web services
Delay
Voice Output
Message Summary is Web services
Fig. 47 shows an illustration of several email objects being presented in the semantic browser via a request skin.
From: Nosa Omoigui
To: John Smith
Cc: Joe Somebody
Subject: Web services
Summary: Web services are software building blocks used for distributed computing
Body: Web services . . .
Email Object 1
Object Skin (Object 1)
Email Object 2
Email Object 3
Email Object 4
Language Translation Skins
Language translation skins are implemented similar to text-to-speech skins except that the transform is on the language axis. The XSLT skin (smart style) can invoke a software engine to automatically perform language translation in real-time and then generate XML that is encoded in Unicode (16 bits per character) in order to account for the universe of languages. The XSLT transform that generates the final presentation output then render the output using the proper character set given the contents of the translated XML.
Language Agnostic Semantic Queries
Semantic queries can also be invoked in a language-agnostic fashion. This is implemented by having a translation layer (the SQML language translator) that translates the SQML that is generated by the semantic browser to a form that is suitable for interpretation by the KDS (or KIBS) which in turn has a knowledge domain ontology seeded for one or more languages. The SQML language translator translates the objects referred to by the predicates (e.g., keywords, text, concepts, categories, etc.) and then sends that to the serverside semantic query processor for interpretation. The results are then translated back to the original language by the language translation skin.
Categories as First Class Objects in the User Experience
This refers to a feature by which categories of a knowledge community are exposed to the end user. The end user will be able to issue a query for a category as an information type—e.g., 'Web services.' The metadata will then be displayed in the semantic browser, as would be the case for any first-class information object type. Visualizations, dynamic links, context palettes, etc. will also be available using the category object as a pivot. This feature is useful in cases where the user wants to start with the category and then use that as a pivot for dynamic navigation, as opposed to starting off with a smart request (smart agent) that has the category as a parameter.
Categorized Annotations
Categorized annotations follow from categories being first-class objects. Users will be able to annotate a category directly—thereby simulating an email list that is mapped to a category. However, for cases where there are many categories (for instance, in pharmaceuticals), this is not
recommended because information can belong to many categories and the user should not have to think about which category to annotate—the user should publish the annotation directly to the knowledge community (agency) where it will be automatically categorized or annotate an object like a document or email message that is more contextual than a category.

11. Additional Context Templates
1. Experts—The Experts feature was indicated as a special agent in my parent application Ser. No. 1/179,651. As should have also been understood from that application, the Experts feature can also operate in conjunction with the context templates section. Experts are a context template and as the name implies indicate people that have expertise on one or more subject matters or contexts (indicated by the PREDICATETYPEID_EXPERTON predicate).

2. Interest Group—this refers to a context template which as the name implies indicate people that have interest (but not necessarily expertise) on one or more subject matters or contexts (indicated by the PREDICATETYPEID_INTERESTIN predicate). This context template returns People that have shown interest in any semantic category in the semantic network. A very real-world scenario will have Experts returning people that have answers and Interest Group returning results of people that have questions (or answers). In the preferred embodiment, this is implemented by returning results of people who have authored information that in turn has been categorized in the semantic network, with the knowledge domains configured for the KIS. Essentially, this context template presents the user with dynamic, semantic communities of interest. It is a very powerful context template. Currently, most organizations use email distribution lists (or the like) to indicate communities of interest. However, these lists are hard to maintain and require that the administrator manually track (or guess) which people in the organization preferably belong to the list(s). With the Interest Group context template, however, the “lists” now become intelligent and semantic (akin to “smart distribution lists”). They are also contextual, a feature that manual email distribution lists lack.

Like with other context templates, the Interest Group context predicate in turn is interpreted by the server-side semantic query processor. This allows powerful queries like “Interest Group on XML,” “Interest Group on Bioinformatics.” Similarly, this would allow queries (via drag and drop and/or for a smart copy and paste) like “Interest Group on My Local Document” and “Interest Group on My Competitor (an entity).” The Interest Group context template also becomes a part of the Dossier (or Guide) context template (which displays all special agents for each context templates and loads them as sub-queries of the main agent/request).

In the preferred embodiment, the context template should have a time-limit for which it detects “areas of interest.” An example of this would be three months. The logic here is that if the user has not authored any information (most typically email) that is semantically relevant to the SQML filter (if available) in three months, the user either has no interest in that category (or categories) or had an interest but doesn’t any longer.

3. Annotations of My Items—this is a context template that is a variant of Annotations but is further filtered with items that were published by the calling user. This will allow the user to monitor feedback specifically on items that he/she posted or annotated.

12. Importing and Exporting User State
1. The semantic browser will support the importation and exportation of user state. The user will be able to save his/her personal state to a document and export it to another machine or vice-versa. This state will include information (and metadata) on:

2. Default user state (e.g., computer sophistication level, default areas of interest, default job role, default smart styles, etc.)
3. Profiles
4. Entities (per profile)
5. Smart requests (per profile)
6. Local Requests (per profile)
7. Subscribed Knowledge Communities (per profile)

8. The semantic browser will show UI (likely a wizard) that will allow the user to select which of the user state types to import or export. The UI will also ask the user whether to include identity/logon information. When the UI is invoked, the semantic browser will serialize the user state into an XML document that has fields corresponding to the metadata of all the user state types. When the XML document is imported, the semantic browser will navigate the XML document nodes and add or set the user state types in the client environment corresponding to the nodes in the XML document.

13. Local Smart Requests
1. Local smart requests would allow the user to browse local information using categories from an knowledge community (agency). In the case of categorized local requests, the semantic client crawls the local hard drives, email stores, etc. extracts the metadata (including summaries) and stores the metadata in a local version of the semantic metadata store (SMS). The client sends the SMS metadata (per object) to an knowledge community for categorization (via its XML Web Service). The knowledge community then responds with the category assignment metadata. The client then updates the local semantic network (via the local SMS) and responds to semantic queries just like the server would. Essentially, this feature can provide functionality equivalent to a local server without the need for one.

1. Integrated Navigation allows the user to dynamically navigate from within the Presenter (in the main results pane on the right) and have the navigation be integrated with the shell extension navigation on the left. Essentially, this merges both stacks. In the preferred embodiment, this is accomplished via event signaling. When the Presenter wants to dynamically navigate to a new request, it sets some state off the GUID that identifies the current browser view. The GUID maps to a key in the registry that also has a field called ‘Navigation Event,’ ‘Next Namespace Object ID’ and ‘Next Path.’ The ‘Navigation Event’ field holds a DWORD value that points to an event handle that gets created by the current browser view when it is loaded. When the Presenter wants to navigate to a new request, it creates the request in the semantic environment and caches the returned ID of the request. It then dynamically gets the appropriate namespace path of the request (depending on the information/context type of the request) and caches that too. It then sets the two fields (‘Next Namespace Object ID’ and ‘Next Path’ with these two values). Next, it sets the ‘Navigation Event’ (in Windows®), this is done by calling a Win32 API named “SetEvent”

13. To catch the navigation event, the browser view starts a worker thread when it first starts. This thread waits on
the navigation event (and also simultaneously waits on a shutdown event that gets signaled when the browser view is being terminated— in Windows®), it does this via a Win32 API (WDF) named ‘WaitForMultipleObjects’). If the navigation event is signaled, the ‘Wait’ API returns indicating that the navigation event was signaled. The worker thread then looks up the registry to retrieve the navigation state (the object id and the path). It then calls the shell browser to navigate to this object id and path (in Windows™, this is done by retrieving a ‘PIDL’ and then calling IShellBrowser::BrowseTo off the shell view instance that implements IShellView).

[1414] 15. Hints for Visited Results

[1415] The Nervana semantic browser empowers the user to dynamically navigate a knowledge space at the speed of thought. The user could navigate along context, information or time axes. However, as the user navigates, he/she might be presented with redundant information. For instance, the user can navigate from a local document to ‘Breaking News’ and then from one of the ‘Breaking News’ result objects to ‘Headlines.’ However, semantically, some of the Headlines might overlap with the breaking news (especially if not enough time has elapsed). This is equivalent to browsing the Web and hitting the same pages over and over again from different ‘angles.’

[1416] The Nervana semantic browser handles this redundancy problem by having a local cache of recently presented results. The Presenter then indicates redundant results to the user by showing the results in a different color or some other UI mechanism. The local cache is aged (preferably after several hours or the measured time of a typical ‘browsing experience’). Old entries are purged and the cache is eventually reset after enough time might have elapsed.

[1417] Alternately, at the users option, the redundant results can be discarded and not presented at all. Specifically, the semantic browser will also handle duplicate results by removing duplicates before rendering them in the Presenter—for instance if objects with the same metadata appear on different Knowledge Communities (agencies). The semantic browser will detect this by performing metadata comparisons. For unstructured data like documents, email, etc., the semantic browser will compare the summaries—if the summaries are identical the documents are very likely to be identical (albeit this is not absolutely guaranteed, especially for very long documents).


[1419] Client-Side Knowledge Federation

[1420] Client-side Knowledge Federation which allows the user to federate knowledge communities and operate on results as though they came from one place (this federation feature was described in my parent application Ser. No. 10/179,651). In the preferred embodiment, such Client-side Knowledge Federation is accomplished by the semantic browser merging SRML results as they arrive from different (federated) KISes.

[1421] Server-Side Knowledge Federation

[1422] Server-Side Knowledge Federation is technology that allows external knowledge to be federated within the confines of a knowledge community. For instance, many companies rely on external content providers like Reuters™ to provide them with information. However, in the Information Nervous System, security and privacy issues arise relating to annotations, personal publications, etc. Many enterprise customers will not want sensitive annotations to be stored on remote servers hosted and managed by external content providers.

[1423] To address this, external content providers will provide their content on a KIS metadata cache, which will be hosted and managed by the company. For instance, Reuters™ will provide their content to a customer like Intel™ but Intel™ will host and manage the KIS. The Intel™ KIS would crawl the Reuters™ KIS (thereby chaining KIS servers) or the Reuters™ DSA. This way, sensitive Intel™ annotations can be published as ‘Post-Its’ using Reuters™ content as context while Intel™ will still maintain control over its sensitive data.

[1424] Federated Annotations

[1425] Federated annotations is a very powerful feature that allows the user to annotate an object that comes from one agency/server (KIS) and annotate the object with comments (and/or attachment(s))—like ‘Post-Its’ on another server. For example, a server (call it Server A) might not support annotations (this is configurable by the administrator and might be the common case for Internet-based servers that don’t have a domain of trust and verifiable identity). A user might get a document (or any other semantic result) from Server B but might want to annotate that object on one or more agencies (KISes) that do support annotations (more typically Intranet or Extranet-based agencies that do have a domain of trust and verifiable identity). In such a case, the annotation email message would include the URI of the object to be annotated (the email message and its attachment(s) would contain the annotation itself). When the server crawls its System Inbox and picks up the email annotation, it scans the annotation’s encoded To or Subject field and extracts the URI for the object to be annotated. If the URI refers to a different server, the server then invokes an XML Web Service call (if it has access) to that server to get the SRML metadata for the object. The server then adds the SRML metadata to its Semantic Metadata Store (SMS) and adds the appropriate semantic links from the email annotation to the SRML object. This is very powerful because it implies that users of the agency would then view the annotation and also be able to semantically navigate to the annotated object even though that object came from a different server.

[1426] If the destination server (for the annotation) does not have access to the server on which the object to be annotated resides, the destination server informs the client of this and the client then has to get the SRML from the server (on which the object resides) and send the complete SRML back to the destination server (for the annotation). This embodiment essentially implies that the client must first “de-reference” the URI and send the SRML to the destination server, rather than having the destination server attempt to “de-reference” the URI itself. This approach might also be superior for performance reasons as it spreads the CPU and I/O load across its clients (since they have to do the downloading and “de-referencing” of the URI to SRML).”

[1427] Semantic Alerts for Federated Annotations

[1428] In the same manner that semantic browser would poll each KIS in the currently viewed user profile for “Breaking News” relevant to each currently viewed object on a regular basis (e.g., every minute), the same will be performed for annotations. Essentially, this resembles polling whether each object that is currently displayed “was just annotated.” For annotations that are not federated (i.e., annotations that have strong semantic links to the objects they annotate), this
is a straightforward SQML call back to the KIS from whence the annotated object came. However, for federated annotations, the process is a bit more complicated because it is possible that a copy of object has been annotated on a different KIS even though the KIS from whence the object came doesn’t support annotations or contain an annotation for the specific object.

[1429] In this case, for each object being displayed, the semantic browser would poll each KIS in the selected profile and pass the URI of the object to “ask” the KIS whether that object has been annotated on it. This way, semantic alerts will be generated even for federated annotations.

[1430] Annotation Hints

[1431] This refers to a feature where the KIS returns a context attribute indicating that an object has been annotated. This can be cached when the KIS detects an annotation (typically from the System Inbox) and is updating the semantic network. This context attribute then becomes a performance optimizer because for those objects with the attribute set, the client wouldn’t have to query the KIS again to check if the object has been annotated. This amounts to caching the state of the object to avoid an extra (and unnecessary) roundtrip call to the KIS.

[1432] Another Perspective on Annotations

[1433] An interesting way to think of the Simple and Semantic Annotations feature of the Information Nervous System is that now every object/item/result in a user’s knowledge universe will have its own contextual inbox. That way, if a user views the object, the inbox that is associated with the object’s context is always available for viewing. In other words,

[1434] Category Naming and Identification (URLs) for Federated Knowledge Communities

[1435] This refers to how categories will be named on federated knowledge communities. For instance, a Reuters™ knowledge community (agency) deployed at Intel™ will be named Reuters@Intel with categories named like “Reuters@Intel/Information Technology/Wireless/802.11”.

[1436] The preferred embodiment, every category will be qualified with at least the following properties:

[1437] Knowledge Domain ID—this is a globally unique identifier that uniquely identifies the knowledge domain from whence the category came

[1438] Name—this is the name of the category

[1439] Path—this is the full taxonomy path of the category

[1440] In this example, the knowledge domain id is c9554be-aeaf-4564-8117-48842b8e5a07 type=c-category & path=InformationTechnology/Wireless/802.11


[1442] The semantic browser will also allow users to anonymously annotate and publish to an anonymous community (agency). In this mode, the metadata is completely stored (with the user identity) but is flagged indicating that the publisher wishes to remain anonymous. This way, the Inference Engine can infer using the complete metadata but requests for the publisher will not reveal his/her identity. Alternately, the administrator will also be able to configure the knowledge community (agency) such that the inference engine cannot infer using anonymous annotations or publications.


[1444] The semantic browser will also have offline support. The browser will have a cache for every remote call. The cache will contain entries to XML data. This could be SRML or could be any other data that gets returned from a call to the XML Web Service. Each call is given a unique signature by the semantic browser and this signature is used to hash into the XML data. For instance, a semantic query is hashed by its SQML. Other remote calls are hashed using a combination of the method name, the argument names and types, and the argument data.

[1445] For every call to the XML Web Service, the semantic runtime client will extract the signature of the call and then map this to an entry in the local cache. If the browser (or the system) is currently offline, the client will return the XML data in the cache (if it exists). If it does not exist, the client will return an error to the caller (likely the Presenter). If the browser is online, the client will retrieve the XML data from the XML Web Service and update the cache by overwriting the previous contents of the file entry with a file path indicated by the signature hash. This assumes that the remote call actually goes through—it might not even if the system/browser is online, due to network traffic and other conditions. In such a case, the cache does not get overwritten (it only gets overwritten when there is new data; it does not get cleared first).


[1447] Overview

[1448] As discussed in my parent application (Ser. No. 10/179,651), the Information Nervous System can be implemented in a cross-platform manner. Standard protocols are preferably employed where possible and the Web service layer should use interoperable Service standards and avoid proprietary implementations. Essentially, the test is that the semantic browser does not have to “know” whether the Knowledge community (or agency) Web service it is talking to is running on a particular platform over another. For example, the semantic browser need not know whether the Web service it is talking to is running on Microsoft’s .NET™ platform or Sun’s J2EE™ platform (to take 2 examples of proprietary application servers), a Linux or any other “open source” server. The Knowledge community Web service and the client-server protocol should employ Web service standards that are commonly supported by different Web service implementations like .NET™ and J2EE™.

[1449] In an ideal world, there will be a common set of standards that would be endorsed and properly implemented across Web service vendor implementations. However, this might not be the case in the real world, at least not yet. To handle a case where the semantic browser must handle unique functionality in different Web service implementations, the Knowledge community schema is preferably extended to include a field that indicates the Web service platform implementation. For instance, a .NET™ implementation of the Knowledge community is preferably published with a field
that indicates that the platform is .NET™. The same applies to J2EE™. The semantic browser will then have access to this field when it retrieves the metadata for the Knowledge community (either directly via the WSDL URL to the Knowledge community, or by receiving announcements via multicast, the enterprise directory (e.g., LDAP), the Global Knowledge community Directory, etc.).

[1450] The semantic browser can then issue platform-specific calls depending on the platform that the Knowledge community is running on. This is not a recommended approach but if it is absolutely necessary to make platform-specific calls, this model is preferably employed in the preferred embodiment.

[1451] 20. Knowledge Modeling

[1452] Knowledge Modeling refers to the recommended way enterprises will deploy an Information Nervous System. This involves deploying several KIS servers (per high-level knowledge domain) and one (or at most few) KDS (formerly KIS) servers that host the relevant ontology and taxonomy. KIS servers are preferably deployed per domain to strike a balance between too narrow such that there is not enough knowledge sharing possibility of navigation and inference in the network and being too high that scalability (in storage and CPU horsepower needed by the database and/or the inference engine) becomes a problem. Of course, the specific point of balance will shift over time as the hardware and software technologies evolve, and the preferred embodiment does not depend on the particular balance struck. In addition, KIS servers are preferably deployed where access control becomes necessary at the server level (for higher-level security) as opposed to imposing access control at the group level with multiple groups sharing the same KIS. For instance, a large pharmaceutical company could have a knowledge community KIS for oncology for the entire company and another KIS for researchers working on cutting-edge R&D and applying for strategic patents. These two KIS might never share the same sources of information but the latter KIS would be more secure because it would provide access only to users from the R&D group. Also, optionally, these researchers’ publications and annotations will not be viewable on the corporate KIS.

[1453] F1G. 48 illustrates an example of a possible knowledge architecture for a pharmaceuticals company. As shown in FIG. 48, the KDS can serve several subsidiary KIS', as follows:

[1454] Client
[1455] Knowledge Integration Server 1 (Oncology)
[1456] Knowledge Integration Server 2 (Pharmacology)
[1457] Knowledge Integration Server 3 (Biotechnology)
[1458] Knowledge Integration Server 4 (Cardiology)
[1459] Knowledge Domain Server (Pharmaceuticals)

[1460] 21. KIS Housekeeping Rules

[1461] The Knowledge Integration Server (KIS) will allow the admin to set up “housekeeping” rules to purge old or stale metadata. This will prevent the SMS on the KIS from growing infinitely large. These rules could be as simple as purging any metadata older than a certain age (between 2-5 years depending on the company’s policies for keeping old data) and which does not have any annotations and that is not marked as a favorite (or rated).

[1462] 22. Client Component Integration & Interaction Workflow

[1463] The client components of the system can be integrated in several different steps or sequences, as can the workflow interaction or usage patterns. In the presently preferred embodiment, the workflow and component integration would be as follows:

1) Shell: User implicitly creates a SQML query (i.e. an agent) via UI navigation or a wizard.

2) Shell: User opens an agent (via tree or folder view).

3) The query buffer is saved as a file, and a registry entry created is created for the agent.

a) Registry entry contains: Agent Name, Creation date, Agent (Request)-GUID, SQML path, Comments, Namespace object type (agency, agent, blender, etc), and attributes

4) Shell: The request is handed off to the presenter:

a) A registry request GUID entry is created containing (namespace path that generated the request, and SQML file URL).

b) Browser is initialized and opened with command line [http://PresenterPage.html/#RequestGUID [http://presenterpage.html/]. The Presenter loads default Chrome contained in the page.

c) Presenter page loads presenter binary behavior and Semantic Runtime OCX.

5) Presenter: Loads SQML and issues requests via the query manager.

6) Presenter: Resolves request GUID to get SQML file path.

b) Loads SQML file into buffer, creates resource handler requests, passes them to resource handlers, waits for and gathers results. Summarization of local resources happens here. All summarization follows one of two paths: Summarize the doc indicated by this file path, or summarize this text (extracted from clipboard, Outlook™, Exchange™, etc.). Both paths produce a summary in the same form, suitable for inclusion in a request to the semantic server XML Web service.

c) Compiles SQML file into individual server request buffers, including any resource summary from above.

d) Initiates Server Requests by calling semantic runtime client Query Manager.

7) Query Manager: Monitors server requests and makes callback on data. It also signals an event on request completion or timeout. The callback is into the Presenter, which mean inter-process messaging to pass the XML.

8) Presenter: receives data and loads appropriate skin:

a) Receives SRML data in buffer; this will happen incrementally.

b) Determines if there is a preferred skin (smart style) associated with this agent, otherwise chooses default skin.

c) Transforms SRML into preferred skin format via XSLT. This is multistage, for the tree of results (root is list, then objects, then DeepLens/BN info) as results come in.

d) Display results in target DIV in page. The target is an argument to the behavior itself and is defined by the root page.

8) Presenter: Calls Semantic Runtime to fill context panels (per context template), deep info, smart copy and paste, and other semantic commands. The Presenter also loads the smart style, which then loads semantic images, motion, etc. consistent with the semantics of the request.

F1G. 49 illustrates the presently preferred client component integration and interaction workflow described above:
23. Categories Dialog Box User Interface Specification

24. Overview

The Categories Dialog Box allows the user to select one or more categories from a category folder (or taxonomy) belonging to a knowledge domain. While more or fewer can be deployed in certain situations, in the preferred embodiment, the dialog box has all of the following user interface controls:

1. Profile—this allows the user to select a profile with which to filter the category folders (or taxonomies) based on configured areas of interest. For instance, if a profile has areas of interest set to “Health and Medicine,” selecting that profile will display only those category folders that belong to the “Health and Medicine” area of interest (for instance, Pharmaceuticals, Healthcare, and Genes). This control allows the user to focus on the taxonomies that are relevant to his/her knowledge domain, without having to see taxonomies from other domains.

2. Area of Interest—this allows the user to select a specific area of interest. By default, this combo box is set to “My Areas of Interest” and the profile combo box is set to “All Profiles.” This way, the dialog box will display category folders for all areas of interest for all profiles. However, by using the “Area of Interest” combo box, the user can directly specify an area of interest with which to filter the category folders, regardless of the areas of interest in his/her profile(s).

3. Publisher Domain Zone/Name—this allows the user to select the domain zone and name of the taxonomy publisher. This is advantageous to distinguish publishers that might have name collisions. In the preferred embodiment, the Publisher Domain Name uses the DNS naming scheme (for instance, IEEE.org, Reuters.com™). The domain zone allows the user to select the scope of the domain name. In the preferred embodiment, the options are Internet, Intranet, and Extranet. The zone selection further distinguishes the published category folder (or taxonomy). A fairly common case would be where a department in a large enterprise has its own internal taxonomy. In this case, the department will be assigned the Intranet domain zone and will have its own domain name—for instance, Intranet/Marketing or Intranet/Sales.

4. Category Folder—this allows the user to select a category folder or taxonomy. When this selection is made, the categories for the selected category folder are displayed in the categories tree view.

5. Search categories—this allows the user to enter one or more keywords with which to filter the currently displayed categories. For instance, a Pharmaceuticals researcher could select the Pharmaceuticals taxonomy but then enter the keyword “anatomy” to display only the entries in the taxonomy that contain the keyword “anatomy.”

6. “Remember” check box—this allows the user to specify whether the dialog box should “remember” the last search when it exits. This is very helpful in cases where the user might want to perform many similar category-based searches/requests from the same category folder and with the same keyword filter(s).

7. Search Options—these controls allow the user to specify how the dialog box should interpret the keywords. The options allow the user to select whether the keywords should apply to the entire hierarchy of each entry in the taxonomy tree, or whether the keywords should apply to only the [end] names of the entries. For instance, the taxonomy entry “Anatomy\Cells\Chromaffin Cells” will be included in a hierarchy filter because the hierarchy includes the word “Anatomy.” However, it will be excluded from a names filter because the end-name (“Chromaffin Cells”) does not include the word “Anatomy.”

Also, the search options allow the user to select whether the dialog box should check for all keywords, for any keyword, or for the exact phrase.

8. Categories Tree View—the tree view displays the taxonomy hierarchy and allows the user to select one or more items to add to the Create Request Wizard or to open as a new Dossier (Guide) request/agent. The user interface breaks the category hierarchy into “category pages”—for performance reasons. The UI allows the user to navigate the pages via buttons and a slide control. There is also a “Deselect All” button that deselects all the currently selected taxonomy items.

9. Explore Button—this is the main invocation button of the dialog box. When the dialog box is launched from the Create Request Wizard, this button is renamed to “Add” and adds the selected items to the wizard “filters” property page. When the dialog box is launched directly from the application, the button is titled “Explore” and when clicked launches a Dossier request on the selected categories. If the user has multiple profiles or if multiple taxonomy categories are selected, the dialog box launches another dialog box, the “Explore Categories Options” dialog box that prompts the user to select the profile with which to launch the Dossier and/or the operator to use in applying the categories as filters to the Dossier (AND or OR).

24. Client-Assisted Server Data Consistency Checking

As the server (KIS) crawls knowledge sources, there will be times when the server’s metadata cache is out of sync with the sources themselves. For instance, a web crawler on the KIS that periodically crawls the Web might add entries into the semantic metadata store (SMS) that become out of date. In this case, the client would get a 404 error and it tries to invoke the source URI. For data source adapters (DSAs) that have monitoring capabilities (for instance, for file-shares that can be monitored for changes), this wouldn’t be much of an issue because the KIS is likely to be in sync with the knowledge source(s). However, for sources such as Web sites that don’t have monitoring/change-notification services, this may present an issue of concern.

My parent application (Ser. No. 10/179,651) described how the KIS can use a consistency checker (CC) to periodically purge stale entries from the SMS. However, in some situations this approach might impair performance because the CC would have to periodically scan the entire SMS and confirm whether the indexed objects still exist. An alternative embodiment of this feature of the invention is to have the client (the semantic browser) notify the server if it gets a 404 error. To do this, the semantic browser would have to track when it gets a 404 error for each result that the user “opens.” For Web documents, the client can poll for the HTTP headers when it displays the results, even before the user opens the results. In this case, if the source web server reports a 404 error (object not found), the client should report this to the KIS.

When the KIS gets a “404 report” from the client, it then intelligently decides whether this means the object is no longer available. The KIS cannot arbitrarily delete the object because it is possible that the 404 error was due to an inter-
mittent Web server failure (for instance, the directory on the Web server could have been temporarily disabled). The KIS should itself then attempt to asynchronously download the object (or at the very least, the HTTP headers in the case of a Web object) several times (e.g., 5 times). If each attempt fails, the KIS can then conclude that the object is no longer available and remove it from the SMS. If another client reports the 404 error for the same object while the KIS is processing the download, the KIS should ignore that report (since it is redundant).

This alternate technique could be roughly characterized as lazy consistency checking. In some situations, it may be advantageous and preferred.

25. Client-Side Duplicate Detection

The server (KIS) performs duplicate detection by checking the source URLS before adding new objects into the semantic metadata store (SMS). However, for performance reasons, it is sometimes advantageous if the server does not perform strict duplicate-detection. In such cases, duplicate detection is best performed at the client. Furthermore, because the client federates results from several KISes, it is possible for the client to get duplicates from different KISes.

As such, it is advantageous if the client also performs duplicate detection.

In the preferred embodiment, the client removes objects that are definitely duplicates and flags objects that are likely duplicates. Definite duplicates are objects that have the same URI, last modified time stamp, summary/concepts, and size. Likely duplicates are objects that have the same summary/concepts, but different URLs, last modified times, or sizes. For objects for which summary extraction is difficult, it is recommended that the title also be used to check for likely duplicates (i.e., objects that have the same summary but different titles are not considered likely duplicates because the summary might not be a reliable indicator of the contents of the object). Also, if summary/concept extraction is difficult (in order to detect semantic overlap/redundancy), the semantic browser can limit the file-size check to plus or minus N% (e.g., 5%)—for instance, an object with the same summary/concepts and different URLs, last-modified times, and sizes might be disqualified as a likely duplicate if the file-size is within 5% of the file-size of the object it is being compared to for redundancy checking.

26. Client-Side Virtual Results Cursor

The client (semantic browser) also provides the user with a seamless user experience when there are multiple knowledge communities (agencies) subscribed to a user profile. The semantic browser preferably presents the results as though they came from one source. Similarly, the browser preferably presents the user with one navigation cursor—as the user scrolls, the semantic browser re-queries the KISes to get more results. In the preferred embodiment, the semantic browser keeps a results cache big enough to prevent frequent re-querying—for instance, the cache can be initialized to handle enough results for between 5-10 scrolls (pages). The cache size is preferably capped based on memory considerations. As the cursor is advanced (or retreated), the browser checks if the current page generates a cache hit or miss. If it generates a cache hit, the browser presents the results from the cache, else if re-queries the KISes for additional results which it then adds to the cache.

The cache can be implemented to grow indefinitely or to be a sliding window. The former option has the advantage of simplicity of implementation with the disadvantage of potentially high memory consumption. The latter option, which is the preferred embodiment, has the advantage of lower memory consumption and higher cache consistency but with the cost of a more complex implementation. With the sliding window, the semantic browser will purge results from pages that do not fall within the window (e.g., the last N—e.g., 5-10—pages as opposed to all pages as with the other embodiment).

27. Virtual Single Sign-On

The client (semantic browser) also provides the user with a seamless user experience when authenticating the user to his/her subscribed knowledge communities (agencies). It does this via what the inventor calls a "virtual single sign-on." This model involves the semantic browser authenticating the user to knowledge communities without the user having to enter his/her username and password per knowledge community. Typically, the user will have a few usernames and passwords but might have many knowledge communities of which he/she is a member (especially within a company based on departmental or group access, and on Internet-based knowledge communities).

As such, the ratio of the number of knowledge communities to the number of authentication credentials (per user) is likely to be very high.

With virtual single sign-on, the user specifies his/her logon credentials to the semantic browser in a server (knowledge community)-independent fashion. The semantic browser stores the credentials in a Credential Cache Table (CCT). The CCT has columns as illustrated below:

<table>
<thead>
<tr>
<th>Account Name</th>
<th>User Name</th>
<th>Password</th>
<th>Knowledge Community Entry List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account Name - this is a friendly name for the account</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Name - this is the logon user name (e.g., an email address)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Password - this is the password, stored encrypted with a secure private key</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Knowledge Community Entry List (KCEL) - this is a list of knowledge communities that authenticate the user using the credentials for this account</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When the user first attempts to subscribe to a knowledge community (or access the knowledge community in some other way—for instance, to get the properties of the community), the semantic browser prompts the user for his/her password and then tries to logon to the server using the supplied credentials. If a logon is successful, the semantic browser creates a new CCT entry (CCTE) with the supplied credentials and adds the KC to the Knowledge Community Entry List (KCEL) for the new CCT entry.

For each subsequent subscription attempt, the semantic browser checks the CCT to see if the KC the user is about to subscribe to is in the KCEL for any CCTE. If it is, the semantic browser retrieves the credentials for the CCTE and logs the user on with those credentials. This way, the user does not have to redundantly enter his/her logon credentials.

Note that the semantic browser also supports pass-through authentication when the operating system is already logged on to a domain. For instance, if a Windows™ machine is already logged on to an NT (or Active Directory™) domain, the client-side Web service proxy also includes the default credentials to attempt to logon to a KC. In the preferred embodiment, the additional credentials supplied by the user are preferably passed via SOAP security headers (via Web Services Security (WS-Security) or a similar scheme). For details of WS-Security and passing authentication-infor-

[1515] The semantic browser exposes a property to allow the user to indicate whether the credentials for a CCTL are preferably purged when the KCEL for the CCTL is empty or whether the credentials should be saved. In the preferred embodiment, the credentials are preferably saved by default unless the user indicates otherwise. If the user wants the credentials purged, the semantic browser should remove a KC from a CCTL in which it exists when that KC is no longer subscribed to any profile in the browser. If after removing the KC from the CCTL's KCEL, the CCTL becomes empty, the CCTL is preferably deleted from the CCL.

[1516] The virtual single sign-on feature, like many of the features in this application, could be used in applications other than with my Information Nervous System or the Virtual Librarian. For example, it could be adapted for use by any computer user who must log into more than one domain.


[1518] The table below shows the actions that the semantic browser invokes when namespace objects are copied and pasted onto other namespace objects.

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Portfolio (Entity Collection)</th>
<th>Object (Result)</th>
<th>Profile</th>
<th>Default Profile</th>
<th>Request</th>
<th>Dossier (Guide)</th>
<th>Knowledge Community (Agency)</th>
<th>Application (Root login)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td></td>
<td>Object Lens (Dossier)</td>
<td>Copy</td>
<td>Copy</td>
<td>Copy</td>
<td>Query</td>
<td>Dossier</td>
<td>Dossier</td>
<td>N/A (Open as bookmark in default profile in alternative embodiment)</td>
</tr>
<tr>
<td>Portfolio</td>
<td></td>
<td>Object Lens (Dossier)</td>
<td>Copy (contents)</td>
<td>Copy</td>
<td>Copy</td>
<td>Query</td>
<td>Dossier</td>
<td>Dossier</td>
<td>N/A (Open as bookmark in default profile in alternative embodiment)</td>
</tr>
<tr>
<td>Object (Result)</td>
<td></td>
<td>Object Lens (Dossier)</td>
<td>Copy (Bookmark)</td>
<td>Copy</td>
<td>Copy</td>
<td>Query</td>
<td>Dossier</td>
<td>Dossier</td>
<td>(Open as bookmark in default profile)</td>
</tr>
<tr>
<td>Profile</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Default</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Request</td>
<td></td>
<td>Smart Lens</td>
<td>Smart Lens</td>
<td>Copy</td>
<td>Copy</td>
<td>Agent Lens</td>
<td>Dossier</td>
<td>Dossier</td>
<td>Copy (to default profile)</td>
</tr>
<tr>
<td>Dossier (Guide)</td>
<td></td>
<td>Smart Lens</td>
<td>Smart Lens</td>
<td>Copy</td>
<td>Copy</td>
<td>Agent Lens</td>
<td>Dossier</td>
<td>Dossier</td>
<td>Copy (to default profile)</td>
</tr>
<tr>
<td>Knowledge Community (Agency)</td>
<td></td>
<td>Smart Lens (from KC)</td>
<td>Smart Lens</td>
<td>Copy (subscribe)</td>
<td>Copy (subscribe)</td>
<td>Agent Lens</td>
<td>Dossier</td>
<td>Dossier</td>
<td>Copy (to default profile)</td>
</tr>
</tbody>
</table>

[1519] 29. Dynamic End-to-End Ontology/Taxonomy Updating and Synchronization

[1520] The Information Nervous System™ will support dynamic updates of ontologies and taxonomies. Knowledge domain plug-ins that are published by Nervana (or that are provided to Nervana by third-party ontology publishers) will be hosted on a central Web service (an ontology depot) on the Nervana Web domain (Nervana.com). Each KDS will then periodically poll the central Web service via a Web service call (for each of its knowledge domain plug-ins, referenced by the URI or a globally unique identifier of the plug-in) and will “ask” the Web service if the plug-in has been updated. The Web service will use the last-modified timestamp of the ontology file to determine whether the plug-in has been updated. If the plug-in has been updated, the Web service will return the new ontology file to the calling KDS. The KDS then replaces its ontology file.

[1521] If the KDS is running during the update, it will ordinarily temporarily stop the service before replacing the file, unless it supports file-change notifications and reloads the ontology (which is the recommended implementation).

[1522] Each KIS also has to poll each KDS it is connected to in order to “ask” the KDS if its ontology has changed. In the preferred embodiment, the KIS should poll the KDS and not the central Web service in case the KDS has a different version of the ontology. The KDS also uses the last modified time stamp of the knowledge domain plug-in (the ontology) to determine if the ontology has changed. It then indicates this to the KIS. If the ontology has changed, the KIS needs to update the semantic network accordingly. In the preferred embodiment, it does this by removing semantic links that refer to categories that are not in the new version of the ontology and adding/modifying semantic links based on the new version of the ontology. In an alternative embodiment, it purges the semantic network and re-indexes it.

[1523] The client then polls each KIS it is subscribed to in order to determine if the taxonomies it is subscribed to (di-
directly via the central Web service or via the KIS(es) have changed. The KIS exposes a method via the XML Web service via which the client determines if the taxonomy has changed (via the last modified time stamp of the taxonomy/ontology plug-in file). If the taxonomy has changed, the client needs to update the Categories Dialog user interface (and other U1-based taxonomy dependents) to show the new taxonomy.

[1524] For taxonomies that are centrally published (e.g., via Nervana), the client should poll the central Web service to update the taxonomies.

[1525] With this model, the client, KIS, KDS, and central taxonomy/ontology depot will be kept synchronized.

[1526] 30. Invoking Dossier (Guide) Queries

[1527] Dossier Semantic Query Processing

[1528] Dossier (Guide) queries are preferably invoked by the client-side semantic query processor by parsing the SQML of the request/agent and replacing the Dossier context predicate with each special agent (context template) context predicate—e.g., All Bets, Best Bets, Breaking News, Headlines, Random Bets, Newsmakers, etc. Each query (per context template) is then invoked via the query processor—just like an individual query. This way, the user operates at the level of the Dossier but the semantic browser maps the dossier to individual queries behind the scenes.

[1529] For example, the SQML for “Dossier on Category C” is parsed and new SQML queries are generated as follows:

[1530] All Bets on Category C

[1531] Best Bets on Category C

[1532] Breaking News on Category C

[1533] Headlines on Category C

[1534] Random Bets on Category C

[1535] Newsmakers on Category C

[1536] Etc.

[1537] The client-side semantic query processor retains every other predicate except the context predicate. This way, the filters remain consistent as illustrated by the example above.

[1538] Dossier Smart Lens

[1539] Like other requests/agents in the Information Nervous System™, dossiers (guides) can be used as a Smart Lens (just like how they can be targets for drag and drop, smart copy and paste, etc.). In this case, the smart lens displays a “Dossier Preview Window” with sections/tabs/frames for each context template (special agent).

[1540] Dossier Screenshots


[1542] The following describe the semantics of a knowledge community (agency) within the context of the semantic namespace/environment in the semantic browser.

[1543] 1. Selecting a knowledge community—this opens a dossier request from that KC. Essentially, the Dossier becomes the equivalent of the KC’s “home page.”

[1544] 2. Drag and drop (document, text, entity, keywords, etc.) to a KC—this opens a Dossier request/agent on the object (using the default predicate) from the KC

[1545] 3. Copy KC to the clipboard—this selects KC as the Smart Lens. When the user hovers over a result or entity, the semantic browser displays the Smart Lens by showing the KC name and the KC’s profile name under the cursor and then opens a Dossier from the KC on the object underneath the lens in the lens preview pane

[1546] 4. Subscribing to a KC—when a KC is subscribed for the first time, the semantic browser adds the KC’s email address to the local email contacts (e.g., in Microsoft Outlook™ or Outlook Express™). This makes it easy for the user to publish knowledge to the KC by sending it email (via the integrated contacts list). Similarly, when the KC is unsubscribed from all profiles, the semantic browser prompts the user whether it should remove the KC from the local email contacts list.

[1547] 32. Dynamic Ontology and Taxonomy Mapping

[1548] One of the challenges of using taxonomies and ontologies is how to map the semantics of one taxonomy/ontology onto another. The Information Nervous System™ accomplishes this by the following algorithm:

[1549] Each KDS will be responsible for ontology mapping (via an Ontology Mapper (OM)) and will periodically update the central Web service (the ontology depot) with an Ontology Mapping Table (OMT). The updates are bi-directional: the KDS will periodically update its ontologies and taxonomies from the central Web service and send updates of the OMT to the central Web service. Each OMT will be different but the central ontology depot will consolidate all OMTs into a Master OMT. The ontology mapper will create a consistent user experience because the user wouldn’t have to select all items in the umbrella taxonomy that are relevant but overlapping. The semantic browser will automatically handle this. The KIS wouldn’t have any concept of the mapper but will get mapped results from the KDS which it will then use to update the semantic network.

[1550] The KDS and KIS administrators would still be responsible for selecting the right KDS ontology plug-ins, however—based on the quality of each ontology/taxonomy (the ontology mapping doesn’t improve ontologies; it merely maps them).

[1551] 33. Semantic Alerts Optimizations

[1552] Semantic Alerts in the semantic browser can be optimized by employing the following rules (in order):

[1553] For a given filter (e.g., result, document, text, keywords, entity):


[1555] 2. If there are Headlines, check for Breaking News and Newsmakers.

[1556] This is because in the preferred embodiment, Headlines are implemented similar to Breaking News except with a larger time window. As a consequence, if there are no Headlines (in the preferred embodiment), there is no Breaking News. Also, in the preferred embodiment, Newsmakers are implemented by returning the authors of Headlines. As such, if there are no Headlines, there are no Newsmakers.

[1557] 34. Semantic “News” Images

[1558] Both Corbis™ (http://www.corbis.com) and Getty Images™ (http://www.gettyimages.com) have “News” images that are constantly kept fresh. The Information Nervous System™ can use these kinds of images for semantic images that are not only context-sensitive but also “fresh.” This can be advantageous in terms of keeping the user interface interesting and “new.” For instance, “Breaking News on SARS” can show not only pharmaceutical images but images showing doctors responding to recent SARS outbreaks, etc.

[1559] 35. Dynamically Choosing Semantic Images

[1560] Semantic images can be dynamically and intelligently selected using the following rules:

[1561] 1. If the currently displayed namespace object is a request, parse the SQML of the object for categories. If there are categories, send the categories to the central Web service
(that hosts the semantic image cache) to get images that are relevant to the categories. Also, send the request type (e.g., knowledge types like All Bets and Headlines, or information types like Presentations) to the central Web service to return images consistent with the request type.

[1562] 2. If the namespace object is not a request, send the areas of interest for the current profile (if available) to the central Web service. The Web service then returns semantic images consistent with the profile's areas of interest. If the profile does not have configured areas of interest, send the areas of interest for the application (the semantic browser). If the application does not have configured areas of interest, send an empty string to the central Web service—in this case, the central Web service returns generic images (e.g., branded images).


[1564] Knowledge communities (agencies) have members (users that have read, write, or read-write access to the community) and contacts. Contacts are users that are relevant to the community but are not necessarily members. For example, a departmental knowledge community (KC) in a large enterprise would likely have the members of the department as members of the KC but would likely have all the employees of the company as contacts. Contacts are advantageous because they allow members of the KC to navigate users that are semantically relevant to the KC but might not be members. The KC might semantically index sent by contacts—the index in this case would include the contacts even though the contacts are not members of the KC.

[1565] Another way to think of this is that communities of knowledge in the real world tend to have core members and peripheral members. Core members are users that are very active in the community while peripheral members include “other” users such as knowledge hobbyists, occasional contributors, potential recruits, and even members of other relevant communities.

[1566] With dynamic KC contacts membership in the Information Nervous System™, the KIS will add users to its Contacts table in the semantic metadata store (SMS) and to the semantic network “when and as it sees them” (in other words, as it indexes email messages that have new users that are not members). This allows the community to dynamically expand its contacts, but in a way that distinguishes between Members and mere Contacts, and “understands” the importance of the distinction semantically when operating the system (e.g., executing searches and the like).

[1567] 37. Integrated Full-Text Keyword and Phrase Indexing

[1568] The KIS also indexes concepts (key phrases) and keywords as first-class members of the semantic network. This can be done in a domain-independent fashion as follows:

[1569] For each new object (e.g., documents) to be added to the semantic network:

[1570] 1. Extract concepts (key phrases) from the body of the object.

[1571] 2. For each content, add the concept to the semantic network with the object type id OBJECTTYPEID_CONCEPT. Add a semantic link with the predicate PREDICATETYPEID_CONTAINS_CONCEPT to the "Semantic Links" table with the new object as subject and the new concept object as the subject.

[1572] 3. For the current concept, extract the keywords from the concept key phrase and add each keyword to the semantic network with the object type id OBJECTTYPEID_KEYWORD. Also, add a semantic link with the predicate PREDICATETYPEID_CONTAINS_KEYWORD to the “Semantic Links” table with the new object as subject and the new keyword object as the subject.

[1573] Repeat the steps above for the title and other meta-tags as appropriate for the schema of the object.

[1574] While some embodiments do not require integrated full-text indexing, it is included in the presently preferred embodiment because it provides several useful advantages:

[1575] 1. It allows a consistent model for implementing semantic filters (in SQML). The user can add categories, documents, entities, and keywords as filters and the filters are applied consistently to the semantic network (as sub-queries).

[1576] 2. In particular, it supports the semantic query processing of entities. Entities can be defined with categories and can be further narrowed with keywords (to disambiguate the keywords in the case where the keywords could mean different things in different contexts). Integrated full-text indexing allows the KIS semantic query processor (SQP) to interpret entities seamlessly—by applying the necessary sub-queries with categories and keywords/concepts to the semantic network.

[1577] 3. In general, integrated full-text indexing results in a seamless and consistent data and query model.

[1578] 38. Semantic “Mark Object as Read”

[1579] In some cases, the KIS might not have the resources to store semantic links between People and objects on a per-object basis. In addition, semantic-based redundancy is not the same as per-object redundancy—as in email. To take an example, email clients allow users to select an email message as read or unread—this is typically implemented as a flag stored on the mail server with the email message. However, because email is not a semantic system, a semantically similar or identical message on the server would not be flagged as such—the user has to flag each message separately regardless of semantic redundancy.

[1580] In the Information Nervous System™, the user is able to flag an object as read not unlike in email. However, in this case, the semantic browser extracts the concepts from the object and informs all the KISes in the request profile that the “concepts” have been read. The KIS then dynamically maps the concepts to categories via the KDSes it is configured with and adds a flag to the objects belonging to those categories (in the preferred embodiment) and/or adds a flag to the semantic network with a semantic link with the predicate PREDICATETYPEID_VIEWEDCATEGORY between the categories corresponding to the concepts and all the objects that are linked to the categories. In the preferred embodiment, the KIS should only flag those categories over a link-strength threshold (for the source concepts). This ensures that only those objects (in the preferred embodiment) and/or categories that are semantically close to the original object will be flagged.

[1581] When the semantic browser flags the object via the KISses, the KISses should return a flag indicating whether the network was updated (it is possible that no changes would be made in the event that the object does not have any “strong” categories or if there are no other objects that share the same “strong” categories). If at least one KIS in the request profile indicates that the network was updated, the semantic browser should refresh the request/agent. The semantic browser can expose a property to allow the user to indicate whether he/she wants the KISses to return only unread objects or all objects (read or unread), in which case the browser should display...
unread objects differently (like how email clients display unread messages in a bold font). The presentation layer in the
semantic browser should then display the read and unread objects with an appropriate font and/or color to provide a
clear visual distinction.

[1583] Multi-select object lens is an alternative implementation of the object lens that was described in my parent
application. In that embodiment, the object lens was invoked via smart copy and paste—pasting an object over another
object would invoke the object lens with the appropriate
default predicate. This has the benefit of allowing the user to
copy objects across instances of the semantic browser, across
profiles, and from other environments (like the file-system,
word processors, email clients, etc.).

[1584] In the currently preferred embodiment, the object
lens is a Dossier Lens (the context predicate is a Dossier, the
filters are the source and target objects, and the profile is the
profile in which the source object was displayed).

[1585] Multi-selection can also be used instead of copy and
paste to invoke an object lens. The semantic browser will
allow the user to select multiple objects (results). The user can
then hit a button (or alternative user-interface object) to
invoke the object lens on the selected objects. In this case, a
Dossier Lens will be displayed (in a preview pane) with a
Dossier context predicate, with the filters as the selected
objects, and the current profile as the request profile.

[1586] 40. Ontology-Based Filtering and Spam
Management

[1587] The KIS (in the preferred embodiment) would only
add objects to the Semantic Metadata Store (SMS) if those
objects belong to at least one category from at least one of the
knowledge domains the KIS is configured with (via one or
more KDSSes). This essentially means the KIS will not index
objects it “does not understand.” The exception to this is that
the KIS will index all objects from its System Inbox—
because this contains at-times personal community-specific
publications and annotations that might be relevant but not
always semantically relevant.

[1588] A side-effect of this ontology-based filtering model
is spam management—ontology-based indexing would be
effective in preventing spam from being indexed and stored.
If users use the semantic browser to access email, as opposed
to their inboxes, only email that has been semantically filtered
will get through.

[1589] 41. Results Refinement

[1590] The results of a request/agent can be further refined
via additional filters and predicates. For example, the request/
agent Headlines on Bioinformatics could be further refined
with keywords specific to certain areas of Bioinformatics.
This way, the end-user can further narrow the result set using
the request/agent as a base. In addition, for time-sensitive
requests, the user can specify a time-window to override the
default time-window. For example, the default Breaking
News time-request could be set to 3 hours. The user should be
able to override this for a specific request/agent (in addition to
changing the defaults on a per-profile or application-wide basis)
with an appropriate UI mechanism (e.g., a slider control
that ranges from 1 hour to 24 hours). The same applies to
Headlines and Newsmakers (e.g., a slider control that ranges
from 1 day to 1 week).

[1591] When the user specifies a filter-override, the semantic
browser invokes the XMI Web Service call for each of the
KISes in the request profile and passes the override arguments
as part of the call. If override arguments are present, the Web
service uses those values instead of the default filter values.
The same applies to additional filters (e.g., keywords)—these
will be passed as additional arguments to the Web service and
the Web service will apply additional sub-queries appropriately
to further filter the query that is specified in the agent/
request SQML (in other words, the SQML is passed as
always, but in addition, the filter overrides and additional
filters are also passed).

[1592] A good case for filter-overrides will be for Best Bets.
The default semantic relevance strength for Best Bets could be
set to 90% (in the preferred embodiment). However, for a
given request/agent, the user might want to see “bets” across
a semantic relevance range. Exposing a relevance UI control
(e.g., a slider control that ranges from 0% to 100%) will allow
this. This essentially allows the user to control the Best Bets
on the fly from “All Bets” (0%) all the way to “Perfect Bets”
(100%).

[1593] A hybrid model should also be employed for
embodiments of context template (special agent) implementa-
tions that involve multiple axes of filtering. For instance,
Breaking News could also impose a relevance filter of 25%
and Headlines and Newsmakers could impose a relevance
filter of 50% (Breaking News has a lower relevance threshold
because it has a higher time-sensitivity threshold; as such, the
relevance threshold can be relaxed). In this case, the semantic
browser should expose UI controls to allow the user to refine
the special agents across both axes (a slider control for time-
sensitivity and another slider control for relevance).

[1594] With dossiers, the semantic browser can display UI
controls for each special agent displayed in the Dossier—the
main Dossier pane can show all the UI controls (changing any
UI control would then refresh the Dossier sub-request for that
special agent). Also, if the Dossier has tabs for each special
agent, each tab can have a UI control specific to the special
agent for the tab.

[1595] 42. Semantic Management of Information Stores

[1596] The Information Nervous System™ can also be
used to manage information stores such as personal email
inboxes, personal contact lists, personal event calendars, a
desktop file-system (e.g., the Microsoft Windows Explorer™
file-management system for local and network-based files),
and also other stores like file-shares, content management
systems, and web sites.

[1597] For client-based stores (such as email inboxes and
file-systems), the client runtime of the semantic browser
should periodically poll the store via a programmatic inter-
face to check for items that have become redundant, stale, or
meaningless. This would address the problem today where
email inboxes keep growing and growing with stale messages
that might have “lost their meaning and relevance.” However,
due to the sheer volume of information users are having to
cope with, many computer users are losing the ability to
manage their email inboxes themselves, resulting in a junk-
heap of old and perhaps irrelevant messages that take up
storage space and make it more difficult to find relevant
messages and items.

[1598] The client runtime should enumerate the items in the
user’s information stores, extract the concepts from the items
(e.g., from the body of email messages and from local docu-
ments) and send the concepts to the KISes in the user’s
profiles. In an alternative embodiment, only the default profile
should be used. The client then essentially “asks” the user’s
subscribed KIses whether the items mean anything to them. In the preferred embodiment, the client should employ the following heuristics:

1599 1. First, check for redundancy—by flagging (or deleting) duplicate email items, duplicate documents that share concepts and summaries (but perhaps with different titles or file-sizes). The client should either delete the duplicate items (user-configurable) or flag the items by moving them into a special folder (user-configurable) in the email client or desktop.

1600 2. Next, for non-duplicate items, the client should check for meaninglessness or irrelevance. First, the client should only check items that are “older” than N days (e.g., 30 days) by examining the last-modified time of the email item, document, or other object. For items that qualify, extract the concepts and call the XML Web Service for each KIS in all the user’s profiles (or the default profile in an alternative embodiment).

1601 3. For very old items (e.g., older than 180 days), the client should specify a very low threshold of meaning to the XML Web Service (e.g., 25%) for preservation. Essentially, this is akin to deleting (or flagging) those items that are very old and weak in meaning.

1602 4. For fairly old items (e.g., older than 90 days old but younger than 180 days old), the client should specify a very low threshold (e.g., 10%) for preservation. This is akin to deleting (or flagging) those items that are fairly old and very weak in meaning.

1603 5. For old items (but not too old—e.g., older than 1 day old but younger than 30 days old), the client should specify a very low threshold (e.g., 0%) for preservation. This is akin to deleting (or flagging) those items that are old (but not too old) but are meaningless, based on the user’s profile.

1604 Essentially, the model for this aspect or feature of the preferred embodiment balances semantic sensitivity with time-sensitivity by imposing a higher semantic threshold on younger items (thereby preserving items that might be largely—albeit not totally—meaningless if they are fairly young. For example, fairly recent email threads might be very weak in meaning—the client should preserve them anyway because their “youth” is also a sign of relevance. As they “age,” however, the client can safely delete them (or flag them for deletion).

1605 This model can also be applied to manage documents on local file-systems. The model can be extended to content-management systems, document repositories, etc. by configuring an Information Store Monitor (ISM) to monitor these systems (via calls to the Information Nervous System™ XML Web Services) and configuring the ISM with KIses that are configured with KDses that have ontologies consistent with the domain of the repositories to be semantically managed. This feature will save storage space and storage/maintenance costs by semantically managing content management systems and ensuring that only relevent items get preserved on those systems over time.

1606 43. Slide-Rule Filter User Interface

1607 The refinement pane in the semantic browser allows the user to “search within results.” The user will be able to add additional keywords, specify date ranges, etc. The date-range control can be implemented like a slide-rule. Shifting one panel in the slide-rule would shift the lower date boundary while moving the other panel will shift the upper date boundary. Other panels can then be added for time boundaries—shifting both time and date panels will impose both date and time constraints. Panels can also be added for other filter axes.

C. Server-Side Semantic Query Processor Specification

1608 1. Overview

1609 This section describes a currently preferred embodiment of how the server-side semantic query processor (SQP) resolves SQML queries. On a given server, queries can be broken into several components:

1610 a. Context (documents, keywords, entities, portfolios (or entity collections)).

1611 b. Context/Knowledge Template (or Special Agent) or Information Template—this describes whether the request if for a knowledge type (e.g., Breaking News, Conversations, Newsmakers, or Popular Items) or for a particular information type (e.g., Documents, Email).

1612 On the client, a semantic query is made up of the triangulation of context, request (or Agent) type, and the knowledge communities (or Agencies). The client sends the SQML that represents the semantic query to all the knowledge communities in the profile in which the request lives. The client asks for a few results at a time and then aggregates the results from one or more servers.

1613 The server-side semantic query processor subdivides semantic queries into several sub-queries, which it then applies (via SQL inner joins or sub-queries in the preferred embodiment). These sub-queries are:

1614 1. Request type sub-query—this represents a sub-query (semantic or non-semantic) depending on the request type. Examples are context (knowledge) types (e.g., All Bets, Best Bets, Headlines, Experts, etc.) and information types (like General Documents, Presentations, Web Pages, Spreadsheets, etc.).

1615 2. Semantic context sub-query—this represents a semantic sub-query derived from the context (filter) passed from the client (an example of this is categories sent from the client or mapped from keywords/text via semantic stemming).

1616 3. Non-semantic context sub-query—this represents a non-semantic sub-query derived from the context (filter) passed from the client (examples are keywords without semantic stemming—mapping to ontology-based categories).

1617 4. Access-control sub-query—this represents a sub-query that filters out those items in the semantic metadata store (SMS) that the calling user does not have access to. For details, see the “Security” specification.

1618 The foregoing steps are illustrated in FIG. 50 (Server-Side Semantic Query Processor Components). FIG. 14 shows how the server-side semantic query processor processes incoming semantic queries (represented as SQML).

1619 2. Semantic Relevance Score

1620 The semantic relevance score defines the normalized score that the concept extraction engine returns. It maps a given term of “blob” of text to one or more categories for a given ontology. The score is added to the semantic network (in the “LinkStrength” field of the “SemanticInks” table) when items are added to the Semantic Network.

1621 3. Semantic Relevance Filter

1622 The relevance filter is different from the relevance score (indeed, both will typically be combined). The relevance filter indicates how the SQP will semantically interpret context (note: in the currently preferred embodiment, the filtering is always semantic in this case). There are two relevance filters: High and Low. With the High relevance filter, the SQP will include a sub-query that is the intersection of categories and terms. For instance, context for the keyword
“XML” will be interpreted as: Items that share the same categories as XML and also include the keyword “XML.” This is the highest level of ontology-based semantic filtering that can occur. However, it could lead to information loss in cases where there are objects in the Semantic Network (or Semantic Metadata Store (SMS)) that are semantically equivalent to the context but that do not share its keywords or terms. For instance, the query described above would miss items that share the same categories as XML but which include the term “Extensible Markup Language” instead. A low relevance filter will only include objects that share the same categories as the context but unlike the high relevance filter, would not include the additional constraint of keyword equivalence.

[1623] For this reason, the relevance filter is preferably used only to create sub-query “buckets” that are then used for ordering results. For instance, the SQP might decide to prioritize a high relevance filter ahead of a low relevance filter when filtering the semantic network but would still return both (with duplicates removed) in order to help guarantee that synonyms don’t get rejected during the final semantic filtering process.

[1624] 4. Time-Sensitivity Filter
[1625] The time-sensitivity filter determines how time-critical the semantic sub-query is. There are two levels: High and Low. A high filter is meant to be extremely time-critical. Default is 3 hours (this accounts for lunch breaks, time away from the office/desk, etc.). A Low filter is meant to be moderately time-critical. The default is 12 hours.

[1626] 5. Knowledge Type Semantic Query Implementations
[1627] Throughout this application certain specific knowledge types are referred to by apt shorthand names, some of which the applicant uses or may use as trademarks. This section explains the nature and function of some of these in greater detail.

[1628] a. All Bets
[1629] For “All Bets” queries, the server simply returns all the items in the semantic metadata store. If the SQML has filters, the filters are imposed via an inner sub-query with no semantic link strength threshold. For instance, All Bets on Topic A will return all items that have anything (strongly or barely) to do with Topic A.

[1630] b. Random Bets
[1631] In the preferred embodiment, for “Random Bets” queries, the server simply returns all the items in the semantic metadata store (like in the case of “All Bets” queries) but orders the results randomly. If the SQML has filters, the filters are imposed via an inner sub-query with no semantic link strength threshold. For instance, Random Bets on Topic A will return all items (ordered randomly) that have anything (strongly or barely) to do with Topic A.

[1632] c. Breaking News
[1633] If the server has user-state, Breaking News can be implemented in a very intelligent way. The table below illustrates the currently preferred ranking and prioritization for Breaking News when the server tracks what items (and/or categories) the user has read:

<table>
<thead>
<tr>
<th>Sub-Query Name</th>
<th>Time-Sensitivity Filter</th>
<th>Semantic Relevance Filter</th>
<th>Primary Ordering Axis</th>
<th>Secondary Ordering Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Breaking</td>
<td>Low</td>
<td>High</td>
<td>Creation Time</td>
</tr>
<tr>
<td></td>
<td>Unread</td>
<td></td>
<td></td>
<td>Semantic Relevance Score</td>
</tr>
<tr>
<td></td>
<td>Semantic News</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[1634] In the preferred embodiment, the server processes SQML for Breaking News (via the Breaking News context predicate) as follows:

[1635] 1. All breaking news is filtered with a sub-query that the returned news must be “younger” than N hours (or days, or months, configurable)—this imposes the key time-sensitivity constraint.

[1636] 2. Breaking News is always semantic.

[1637] 3. In the preferred embodiment, the Semantic Network Manager (SNM) should update the semantic network to indicate the “last read time” for each user to each category. This is then used in the sub-query to check whether news has been “read” or not (per category or per object—per category is the preferred embodiment because the latter will not scale).

[1638] 4. Priority is given to news items that the user has not “read” (this is implemented by comparing the last read time in the SemanticLinks table with the semantic link type that links “User” to “Category”).

[1639] 5. The implication of the semantic prioritization scheme is that the user could get “older” breaking news first because the news is more semantically relevant and “younger” breaking news “later” because the news is less semantically relevant. This results in a hybrid relevance-time sensitivity prioritization scheme.

[1640] 6. The primary ordering axis (Creation Time) guarantees that results are filtered by freshness. The secondary ordering axis (Relevance Score) acts as a tiebreaker and guarantees that equally fresh results are distinguished primary based on relevance.

[1641] 7. Breaking News Intrinsic Alerts can be implemented on the client by limiting the Breaking News priority to Priority 2 and by changing the Priority 1 and Priority time-sensitivity filters to high. This way, only very fresh Breaking Unread Semantic News (of both high and low semantic relevance filters) will be returned. This is advantageous because the alert should have a higher disruption threshold than the Breaking News Request (or agent)—since it is implicit rather than explicit.

[1642] 8. Unread Breaking News is higher priority than Read Breaking News because users are likely to be more interested in stuff they haven’t seen yet.

[1643] 9. Unread Breaking News has a lower time-sensitivity filter than Read Breaking News because users are likely to be more tolerant of older news that is new to them than younger news that is not.
[1644] In some cases, the server might not have user-state (and "read" information). In this case, a simple implementation of Breaking News is shown below:

[1645] 1. By default (no filter), Breaking News should return only items younger than N hours (default is 3 hours).

[1646] 2. If there is at least one filter in the SQML, Breaking News should apply the time-sensitivity filter (3 hours) to the outer sub-query and also apply a moderately strong relevance filter to the inner sub-query (off the Semantic.links table). In the preferred embodiment, this should correspond to a relevance score (and link strength) of 50%. For instance, Breaking News on Topic A should return those items that have been posted in the last 3 hours and which belong to the category (or categories) represented by Topic A with at least a relevance score of 50%. This will avoid false positives like Breaking News items which are barely relevant to Topic A.

[1647] d. Headlines

[1648] Ditto with Breaking News (except that time-sensitivity constraints are more relaxed—e.g., the High filter is 12 hours instead of 3 hours and the low filter is 1 day instead of 12 hours). In the simple implementation, the time-sensitivity constraint is 1 day. This can also be made 3-days on Mondays to dynamically handle weekends (making the number of days the “number of working days”).

[1649] e. Newsmakers

[1650] Newsmakers are handled the same way as Headlines, except that the SQP returns the authors of the Headline items rather than the items themselves.

[1651] f. Best Bets

[1652] As described in my parent application (Ser. No. 10/179,551), Best Bets are implemented by imposing a filter on the strength of the semantic link with the “Belongs to Category” predicate. The preferred default is 90%, although the client (at the option of the user) can change this on the fly via an argument passed via the XML Web Service. Best Bets are implemented with a SQL inner join between the Objects table and the Semantic.links table and joining only those rows in the Semantic.links table that have the “Belongs to Category” predicate and a LinkStrength greater than 90% (default). When the SQML that is being processed contains filters (e.g., keywords, text, entities, etc.), the server-side semantic query processor must also invoke a sub-query, which is an SQL inner join that maps to the desired filters. In the preferred embodiment, this sub-query should also include a “Best Bets” filter.

[1653] In the preferred embodiment, it is advantageous and probably preferable for most users for the outer sub-query to be a Best Bet, and for the inner sub-query. To illustrate this, “Best Bets on Topic A” is semantically different from “Best Bets that are also relevant to Topic A.” In the first example, only Best Bets, which are Best Bets “ON” Topic A, will be returned (via applying the “Best Bets” semantic filter on the inner sub-query). In contrast, the second example will return Best Bets on anything that might have anything to do with Topic A. As such, the second example might return false positives because for example, a document, which is a Best Bet on Topic B but a “weak bet” on Topic B, will be returned and that is not consistent with the semantics of the query or the presumably desired results. Extending the “Best Bets” filter to not only the outer sub-query but also all inner sub-queries will prevent this from happening. Other query implementations can also follow this rule (with the right sub-queries applied based on the semantics of the main query) if the SQML contains filters.

[1654] g. Query Implementation for Other Knowledge Types

[1655] Other knowledge types are implemented in a similar fashion as above (via the right predicates). Several examples are described below.

[1656] Information Type Semantic Query Implementations

[1657] All information type semantic query implementations can follow, and preferably (but not necessarily) follow, the same pattern: the SQP returns only those objects that have the object type id that corresponds to the requested information type. An example is “Information Type:Presentations.” When the SQP parses the SQML received from the client, it extracts this attribute from the SQML and maps it to an object type id. It then invokes a SQL query with an added filter for the object type id. For special information types that could span several individual information types (such as “Information Type:All Documents”), the SQP maps the request to a set of object type ids and invokes a SQL query with this added filter.

[1658] Context Semantic Query Implementations

[1659] When the client sends SQML that contains concepts (extracted on the client from text or documents), the server-side SQP has to first semantically interpret the context before generating sub-queries that correspond to it. To do this, the server sends the concepts to all KDS’es (KBS’es) it is configured with (for the desired knowledge community or agency) for semantic categorization. When the server gets the categories back, it preferably determines which of those categories are “strong” enough to be used as filters before generating the appropriate sub-queries.

[1660] This “filter-strength” determination is advantageous because if the context is, for example, a fairly long document, that document could contain thousands of concepts and categories. As a result, the “representative semantics” of the document might be contained in only a subset of all the concepts/categories in the document. Mapping all the categories to sub-queries will return results that might be confusing to the user—the user would likely have a “sense” of what the document contains and if he/she sees results that are relevant to some weak concepts in the document, the user might not be able to reconcile the results with the document context. Therefore, in the preferred embodiment, the server-side SQP preferably chooses only “strong categories” to apply to the sub-queries. It is recommended that these be categories with a semantic strength of at least 50%. That way, only those categories that register strongly in the semantic context would be applied to the sub-query. The implementation of the sub-query would then follow the rules described above depending on whether the query contains a context predicate, is based on a knowledge type, information type, etc.

[1661] Semantic Stemming Implementation

[1662] As described in my parent application, the server-side semantic query processor performs semantic stemming to map keywords, text, and concepts to categories based on one or more domain ontologies. One way it does this by invoking an XML Web Service call to the KDS/KBS (or KDSes/KBSes) it is configured with in order to obtain the categories. It then maps the categories to its semantic network. This form of stemming is superior to regular stemming that is based on keyword variations (such as singular and plural variations, tense variations, etc.) because it also involves domain-specific semantic mapping that stems based on meaning rather than merely stemming based on keyword forms.
In the currently preferred embodiment, the KIS calls the KDS/KBS each time it receives SQML that requires further semantic interpretation. However, this could result in delays if the KDS/KBS resides on a different server, if the network connection is not fast, or if the KDS/KBS is busy processing many requests. In this case, the KIS can also implement a Semantic Stacking Cache. This cache maps keywords and concepts to categories that are fully qualified with URLs (making them globally unique). When the server-side semantic query processor receives SQML that contains keywords, text, or concepts (extracted from, say, documents on the client or the client-side semantic query processor), it first checks the cache to see if the keywords have already been semantically stemmed. If there is a cache hit, the SQP simply retrieves the categories from the cache and maps those categories to the semantic network via SQL queries. If there is a cache miss (i.e., if the context is not in the cache), it then calls the KDSes/KBSes to perform semantic categorization. It then takes the results, maps them to unique category URLs, and adds the entry to the cache (with the context as the hash code). Note that even if the context does not map to any category, the “lack of a category” is preferably cached. In other words, the context is added as a cache entry with no categories. This way, the server can also quickly determine that a given context does not have any categories, without having to call the KDSes/KBSes each time to find out.

Cache Management

The SQP can also manage the semantic stemming cache. It has to do this for two reasons: first, to keep the cache from growing uncontrollably and consuming too much system resources (particularly memory with a heap-based hash table); and, second, if the KIS configuration is changed (e.g., if knowledge domains are added/removed), the cache is preferably purged because the entries might now be stale. The first scenario can be handled by assigning a maximum number of entries to the cache. In the preferred embodiment, the SQP caches the current amount of memory consumed by the cache and the cache limit is dictated by memory usage. For example, the administrator might set the maximum cache size to 64 MB. To simplify the implementation, this can be mapped to an approximate count of items (e.g., by dividing the maximum memory usage by an estimate of the size of each cache entry).

For each new entry, if the cache limit has not been reached, the SQP simply adds the entry to the cache. However, if the cache limit has been reached, the SQP (in the preferred embodiment) should purge the least recently added items from the cache. In the preferred embodiment, this can be implemented by keeping a queue of items that is kept in sync with a hash table that implements the cache itself (for quick lookups using the context as a key). When the SQP needs to purge items from the cache to free up space, it dequeues an item from the least-recently-added queue and also removes the corresponding item from the hash table (using the context as key). This way, fresh items are more likely to result in a cache hit than older items. This will result in a faster user experience on the client because context for saved agents/requests/queries will end up being cached with quick-lookups each time the user opens the agent/request/query. The same goes for Dossier (Guide) queries which will have the same context (but with different knowledge types)—the client will request for each knowledge type for the same context and since the context will be cached, each sub-query will execute faster.

D. Extensible Client-Side User Profiles Specification for the Information Nervous System

Overview

Extensible client-side user profiles allow the user of a semantic browser to have a different state for different job roles, knowledge sources, identities, personas, work styles, etc. This essentially allows the user to create different “knowledge worlds” for different scenarios. For instance, a Pharmaceuticals researcher might have a default profile that includes all sources of knowledge that are relevant to his/her work. As described in my parent application Ser. No. 10/179, 651, the SRML from each of these sources will be merged on the client thereby allowing the user to seamlessly go through results as though they were coming from one source. However, the researcher might want to track patents separate from everything else. In such a case, the researcher would be able to create a separate “Patents” profile and also include those knowledge communities (agencies) that have to do with patents (e.g., the US Patent Office Database, the EU Patent Database, etc.)

To take another example, for instance, the user might create a profile for ‘Work’ and one for ‘Home’. Many investment analysts track companies across a variety of industries. With the semantic browser, they would create profiles for each industry they track. Consultants move from project to project (and from industry to industry) and might want to save requests and entities created with each project. Profiles will be used to handle this scenario as well.

Profiles contain the following user state:

Name/Description—the descriptive name of the profile.

One or more knowledge communities (agencies) that indicate the source of knowledge (running on a KIS) at which requests (agents) will be invoked.

Identity Information—the user name (currently tagged with the user’s email address) and password.

Areas of Interest or Favorite Categories—this is used to suggest information communities (agencies) to the user (by comparing against information communities with identical or similar categories) and as a default query filter for requests created with the profile.

Smart styles—the smart styles to be used by default for requests and entities created with the profile.

Default Flag—this indicates whether the profile is the default profile. The default profile is initiated by default when the user wishes to create requests and entities, browse information communities, etc. Unless the user explicitly selects a different profile, the default profile gets used.

Profiles can be created, deleted, modified, and renamed. However, in the preferred embodiment the default profile cannot be deleted because there has to be at least one profile in the system at all times. In alternate embodiments, a minimum profile would not be required.

Preferably, all objects in the semantic browser are opened within the context of a profile. For instance, a smart request is created in a profile and at runtime, the client semantic query processor will use the properties of the profile (specifically the subscribed knowledge communities (agencies) in that profile) to invoke the request. This allows a user to correlate or scope a request to a specific profile based on the knowledge characteristics of the request (more typically the sources of knowledge the user wants to use for the request).

In a preferred embodiment, the user is able to navigate his/her knowledge worlds via both profiles without interference.
E. Smart Styles Specification for the Information Nervous System

1. Smart Styles Overview
A color theme and animation theme applied to a style theme yields a “smart style”. “Smart” in this context means the style is adaptive or responsive to the mood of its request, context panes, preview mode, handheld mode, live mode, slideshow mode, screensaver mode, blender/collection mode, accessibility, user settings recognition, and possibly other variables within the system (see below). There is an infinite number and kind or “Classes” of possible styles. The preferred embodiment comprises at least the following style Classes:

1. Subtle—for task-oriented productivity.
2. Moderate—for task-oriented productivity with some presentation effects.
3. Exciting—exciting effects (good for both primary and secondary machines, and for inactive Nervana Windows™ e.g., Nervana client Windows™ in the background or docked on the taskbar).
4. Super-exciting (great for smart screensavers with productivity e.g., secondary machines—when the user is using his/her primary machine).
5. Sci-Fi (for Matrix fans, great for smart screensavers without specific need for productivity—e.g., when the user is away from his/her desk).
6. Style, Color & Animation Themes—Variable, unlimited—created by Nervana, and perhaps users and/or third party skin authors

2. Implicit and Dynamic Smart Style Properties
Preferably, each smart style is responsible, consistent with the semantics of the request, for recognizing (or discerning or perceiving) and then Visualizing (or presenting or depicting or illustrating, consistent with what should deserve the user’s attention):

1. The Mood of the Current Request (including semantic images, motion, chrome, etc.
2. A change in the number of items in the Current Request
3. The Mood of each object (intrinsically)
4. The Mood of each object’s context (headlines, breaking news, experts, etc.)
5. Binary/Absolute issues or characteristics (e.g., is there breaking news, OR NOT? how many experts are there? how many headlines?) distinct from issues that are matters of degree, or on a gradient or continuum
6. If the characteristic is on a gradient or continuum, perceiving the relative placement along it (e.g., how breaking is breaking news?, how critical are the headlines? what is the level of expertise for the experts?, etc.)

A change in each object’s context (there is new breaking news, there are new annotations, etc.)

The RELATIVE criticality of each object being displayed (different sized view ports, different fonts, different chrome, etc.)

A request navigation and “loading” status (interstitials that INTRODUCE the mood of the new request being loaded)

All properties of any individual PIP Windows (animated with an animation control)

All of the addition of a new PIP window (to a PIP window palette)

Any Resizing/Moving/Docking PIP Windows

Any preview windows (for context palettes, “Visualization UP” on each object, timelines, etc.)

Sounds consistent with all of the foregoing Visualizations of mood and notifications (across the board)

In a preferred embodiment, a Dialog Box can allow the user to browse smart styles by pivoting across style classes, style themes, color themes, and animation themes. A preview window shows the user a preview of the currently selected smart style.

F. Smart Request Watch Specification for the Information Nervous System

1. Overview
Smart Request Watch refers to a feature of the Information Nervous System that allows users of the semantic browser (the Information Agent or the Librarian) to monitor (or “watch”) smart requests in parallel. This is a very advantageous feature in that it enhances productivity by allowing users to track several requests at the same time.

The feature is implemented in the client-side semantic runtime, the semantic browser, and skins that allow a configurable way of watching smart requests (via a mechanism similar to “Picture-In-Picture” (PIP) functionality in television sets). Preferably, one or more of the following software components are used:

1. The Request Watch List (RWL)
2. Request Watch Groups
3. The Notification Manager (NM)
4. Watch Group Monitors (WLM)
5. The Watch Pane
6. The Watch Window
2. Request Watch Lists (RWLs) and Groups (RWGs)

The Request Watch List is a list of smart requests (or smart agents) that the client runtime manages. This list essentially comprises the smart requests the user wishes to monitor. The Request Watch List comprises a list of entries, the Request Watch List Entry (RWLE) with the following data structure:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequestID</td>
<td>GUID</td>
<td>The unique identifier of the smart request</td>
</tr>
<tr>
<td>Notification Reference Count</td>
<td>DWORD</td>
<td>The reference count indicating whether the Notification Manager should track whether there are “new” objects for this smart request</td>
</tr>
<tr>
<td>RequestViewInstanceId</td>
<td>GUID</td>
<td>The unique identifier of the smart request view instance that “owns” the RWLE. This is used for dynamically added and browser-instance-specific</td>
</tr>
</tbody>
</table>
[1717] The Request Watch List (RWL) contains an array or vector of RWLE structures. The Request Watch List Manager manages the RWL. The semantic browser provides a user interface that allows the user to add smart requests to the RWL—the UI talks to the RWLM to add and remove RWLEs to/from the RWL. The RWL is stored (and persisted) centrally by the client-side semantic runtime (either as an XML file-based representation or in a store like the Windows™ registry).

[1718] The RWL can also be populated by means of Request Watch Groups (RWGs). A Request Watch Group provides a means for the user to monitor a collection of smart requests. It also provides a simple way for users to have the semantic browser automatically populate the RWL based on configurable criteria. There are at least two types of RWGs: Auto Request Watch Group and the Manual Request Watch Group. Auto Request Watch Groups are groups that are dynamically populated by the semantic browser depending on the selected profile, the profile of the currently displayed request, etc. The Manual Request Watch Group allows the user to manually populate a group of smart requests (regular smart requests or blends) to monitor as a collection. The Manual Request Watch Group also allows the user to add support context types (e.g., documents, categories, text, keywords, entities, etc.)—in this case, the system will dynamically generate the semantic query (SQLML) from the filter(s) and add the resulting query to the Manual Request Watch Group. This saves the user from having to first create a time-sensitive request based on one or more filters before adding the filters to the Watch Group—the user can simply focus on the filters and the system will do the rest.

[1719] Users will be able to add the following types of Auto-RWGs:

[1720] 1. Breaking News—this tells the semantic browser to automatically add a Breaking News smart request to the RWL (for the selected profile(s)).

[1721] 2. Headlines—this tells the semantic browser to automatically add a Headlines smart request to the RWL (for the selected profile(s)).

[1722] 3. Newsmakers—this tells the semantic browser to automatically add a Newsmakers smart request to the RWL (for the selected profile(s)).

[1723] 4. Categorized Breaking News—this tells the semantic browser to automatically add Categorized Breaking News smart requests to the RWL (for the contextual profile). The semantic browser will dynamically add smart requests with category filters corresponding to each subcategory of the currently displayed smart request (and for the contextual or current profile)—if the currently displayed smart request has categories. For example, if the smart request “Breaking News” about Technology” is currently being displayed in a semantic browser instance and if the category “Technology” has 5 sub-categories (e.g., Wireless, Semiconductors, Nanotechnology, Software, and Electronics), the following smart requests will be dynamically added to the RWL when the current smart request is loaded:


[1725] Breaking News about Technology:Semiconductors

[1726] Breaking News about Technology:Nanotechnology

[1727] Breaking News about Technology:Software

[1728] Breaking News about Technology:Electronics

[1729] Also, the RWLEs for these entries will be initialized with the RequestViewInstanceId of the current semantic browser instance. If the user navigates to a new smart request, the categorized Breaking News for the previously loaded smart request will be removed from the RWL and a new list of categorized Breaking News will be added for the new smart request (if it has any categories) and initialized with a new RequestViewInstanceId corresponding to the new smart request view. This creates a smart user experience wherein relevant categorized breaking news (for subcategories) will be dynamically displayed based on the currently displayed request. The user will then be able to monitor Categorized Breaking News smart requests as a watch group or collection.

[1730] 5. Categorized Headlines—this tells the semantic browser to automatically add Categorized Headlines smart requests to the RWL (for the contextual profile). This is similar to Categorized Breaking News, except that Headlines are used in this case. The user will then be able to monitor Categorized Headlines smart requests as a watch group or collection.

[1731] 6. Categorized Newsmakers—this tells the semantic browser to automatically add Categorized Newsmakers smart requests to the RWL (for the contextual profile). This is similar to Categorized Breaking News, except that Newsmakers are used in this case. The user will then be able to monitor Categorized Newsmakers smart requests as a watch group or collection.

[1732] 7. My Favorite Requests—this tells the semantic browser to automatically add all favorite smart requests to the RWL (for the selected profile(s)). This allows the user to watch or monitor all his/her favorite smart requests as a group.
8. My Favorite Breaking News—this tells the semantic browser to automatically add all favorite breaking news smart requests to the RWL (for the selected profile(s)). This allows the user to watch or monitor all his/her favorite breaking news smart requests as a group.

9. My Favorite Headlines—this tells the semantic browser to automatically add all favorite headlines smart requests to the RWL (for the selected profile(s)). This allows the user to watch or monitor all his/her favorite headlines smart requests as a group.

10. My Favorite Newsmakers—this tells the semantic browser to automatically add all favorite newsmakers smart requests to the RWL (for the selected profile(s)). This allows the user to watch or monitor all his/her favorite newsmakers smart requests as a group.

Request Watch Group Manager User Interface

FIG. 19 illustrates the “Smart Request Watch” Dialog Box in the semantic browser of the preferred embodiment. The top half of the dialog box is used to add auto-watch groups. The user can select auto-watch group types and profile types (“All Profiles,” “Contextual Profile,” and the actual profile names) and add them to the auto-watch-group list. The user can also remove auto-watch-groups. The bottom half of the dialog box is used to add/remove smart requests to/from the manual watch group.

3. The Notification Manager (NM)

In the preferred embodiment the Notification Manager (NM) is a component of the semantic runtime client that monitors smart requests in the RWL. The NM has a thread that periodically invokes each smart request in the RWL (via the client semantic query processor) and updates the RWLE with the “results count” and the “last update time.” In the preferred embodiment the NM preferably invokes the smart requests every 5-30 seconds. The NM can intelligently adjust the periodicity or frequency of request checks depending on the size of the RWL (in order to minimize bandwidth usage and the scalability impact on the Web service).

For time-sensitive smart requests (like Breaking News, Headlines, and Newsmakers), the NM preferably invokes the smart request without any additional time filter. However, for non time-sensitive requests (like for information as opposed to contexti types or for non time-sensitive context templates like Favorites and Recommendations), the NM preferably invokes the query for the smart request with a time filter (e.g., the last 10 minutes).

4. Watch Group Monitors

In the preferred embodiment, the semantic runtime client manages what the inventors calls Watch Group Monitors (WGM). For each watch group the user has added to the watch group list, the client creates a watch group monitor. A watch group monitor tracks the number of new results in each request in its watch group. The watch group monitor creates a queue for the RWLEs in the watch group that have new results. The WGM manages the queue in order to maximize the freshness of the results. The WGM periodically polls the NM to see whether there are new results for each request in its watch group. If there are, it adds the request to the queue depending on the ‘last result time’ of the request. It does this in order to prioritize requests with the freshest results first. The currently displayed visual style (skin) running in the Presenter would then call the semantic runtime OCX to dequeue the requests in the WGM queue. This way, the request watch user interface will be consistent with the existence of new results and the freshness of the results. Once there are no more new results in the currently displayed request, the smart style will dequeue the next request from the WGM queue.

5. The Watch Pane

The Watch Pane (WP) refers to a panel that gets displayed in the Presenter (alongside the main results pane) and which holds visual representations of the user’s watch groups. The WP allows the user to glance at each watch group to see whether there are new results in its requests. The WP also allows the user to change the current view with which each watch group’s real-time status gets displayed. The following views are currently defined:

Tiled View—this displays the title of the watch group along with the total number of new results in all its smart requests.

Ticker View—this displays the total number of new results in all the watch group’s smart requests but also shows an animation that sequentially displays the number of new results in each smart request (as a ticker).

Preview View—this is similar to the ticker view except that the most recent result per smart request is also displayed alongside the number of new results in the ticker.

Deep View—in this view, the WP displays the total number of new results in all the watch group’s smart requests along with a ticker that shows the number of new results in each smart request and a slide-show of all the new results per smart request.

6. The Watch Window

The WP also allows the user to watch a watch group. The user will do this by selecting one of the watch groups in the WP and dragging it into the main results pane (or by a similar technique). This forms a Watch Window (WW). This WW resembles or can be analogized to TV’s picture-in-picture functionality in appearance or layout, but differs in several ways, most noticeably in that in this case the displayed content is comprised of semantic requests and results as opposed to television channels are being “watched.” Of course, the underlying technology generating the content is also quite different. The WW can be displayed in any of the aforementioned views. When the WW is in Deep View however, the WW’s view controls are displayed. The following controls are currently defined:

Pinning Requests—this allows the user to pin a particular request in the watch group. The WW will keep displaying the new results for only the pinned requests (in a cycle) and will not advance to other requests in the watch group for as long as the current request remains pinned.

Swapping Requests—this allows the user to swap the currently displayed request with the main request being shown in the semantic browser. The smart style will invoke a method on the OCX to create a temporary request with the swapped request (hushed by its SQML buffer) and then navigate to that request while also informing the Presenter to now display the main request in its place (in the WW).

Stop, Play, Seek, FF, RW, Speedup—these allow the user to stop, play, seek, fast-forward, rewind or speed up the “watch group request stream.” For instance, a fast-forward will advance to several requests ahead of the currently displayed one.

Results controls—this allows the user to control the results in each request in the watch group. Essentially, the results are a stream within a stream and this will also allow the user to control the results in the current request in the current watch group.
Auto-Display Mode—this will automatically hide the WW when there are no results to display and fade it in when there are new results. This way, the user can maximize the utility of his/her real estate on the screen knowing that watch windows will fade in when there are new semantic results. This feature also allows the user to manage his/her attention during information interaction in a personal and semantic way.

Docking, Closing, Minimizing, Maximizing—these features, as the names imply, allow the user to dock, close, minimize or maximize watch windows. FIG. 53 illustrates a Watch Window displaying Filtered Smart Requests (e.g., Headlines on Wireless). FIG. 53 is an Illustration of the Watch Window with a Current Smart Request Title (e.g., “Breaking News”).

In the User Interface, the Watch List can be named “News Watch.” The user will be asked to add/remove requests, objects, keywords, text, entities, etc. to/from the “News Watch.” The “News Watch” can be viewed with a Newsstand watch pane. This will provide a spatially-oriented view of the user’s requests and dynamically-created requests (via objects added to the Watch List, and created dynamically by the runtime using those objects as filters)—not unlike the view of a news-magazine rack when one walks into a Library or Bookstore.

G. Entities Specification for the Information Nervous System

1. Introduction

Entities are a very powerful feature of the preferred embodiment of the Information Nervous System. Entities allow the user to create a contextual definition that maps to how they work on a regular basis. Examples of entities include:

<table>
<thead>
<tr>
<th></th>
<th>People</th>
<th></th>
<th>Mammals</th>
<th>Animals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Teams</td>
<td>3</td>
<td>Action Items</td>
<td>Dogs</td>
</tr>
<tr>
<td>2</td>
<td>Companies</td>
<td>5</td>
<td>Competitors</td>
<td>Boxers</td>
</tr>
<tr>
<td>7</td>
<td>Customers</td>
<td>9</td>
<td>Meetings</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Organizations</td>
<td>10</td>
<td>Partners</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Products</td>
<td>12</td>
<td>Topics</td>
<td></td>
</tr>
</tbody>
</table>

There are also industry-specific entities. For instance, in pharmaceuticals, entities could include drugs, drug interaction issues, patents, FDA clinical trials, etc. Essentially, an entity is a semantic envelope that represents a contextual object. An entity can be dragged and dropped like any other smart object. However, an entity is represented by SQML, and not SRML (i.e., it is a query-object because it has much richer semantics). An entity can be included as a parameter to a smart request.

The user creates entities based on his/her tasks. Entities in the preferred embodiment contain at least the following information (in alternate embodiments they could contain more or less information):

1. Name/Description—a friendly descriptive name for the entity.

2. The categories of the entity—based on standard cross-industry taxonomies or vertical/company-specific taxonomies.

3. Contextual resources—these could include keywords, local documents, Internet documents, or smart objects (such as people).

An entity can be opened in the semantic browser, can be used as a pivot for navigation, as a parameter for a smart request (e.g., Headlines on My Project), can be dropped and dragged, can be copied and pasted, can be used with the smart lens, can be visualized with a smart style, can be used as the basis for an alert, can be saved as a .ENT document, can be emailed, shared, etc. In other words, an entity is a first-class smart object.

The semantic runtime client dynamically creates SQML by appending the rich metadata of the entity to the subject of the relational request to create a new rich SQML that refers to the entity.

Entities preferably also have other powerful characteristics:

1. Regarding topics, entities allow the user to create his/her private taxonomy (without being at the mercy of or restricted exclusively to a public taxonomy that is strictly defined and as such, might not map exactly to the user’s specific context for a request). The problem with taxonomies is that no taxonomy can ever fit everybody’s needs—even in the same organization. Context is very personal and entities allow the user to create a personal taxonomy. For instance, take the example of a dog (of the boxer breed) named Kashmir owned by a dog-owner Steve. To everyone else (but Steve), Kashmir can be expressed (taxonomically) as:

<table>
<thead>
<tr>
<th></th>
<th>Living Things</th>
<th></th>
<th>Animals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>My Loved Ones</td>
<td>2</td>
<td>My Pets</td>
</tr>
<tr>
<td></td>
<td>My Clients</td>
<td>3</td>
<td>Kashmir</td>
</tr>
<tr>
<td></td>
<td>My Dogs</td>
<td></td>
<td>Kashmir</td>
</tr>
</tbody>
</table>

To Steve’s veterinary doctor, however, Kashmir is:

<table>
<thead>
<tr>
<th></th>
<th>My Clients</th>
<th></th>
<th>My Dogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>My Clients</td>
<td>2</td>
<td>My Dogs</td>
</tr>
</tbody>
</table>

If taxonomies (standalone) were used to “define” Kashmir, none of the three taxonomies would satisfy the general public, Steve, and Steve’s veterinary doctor. With entities on the other hand, Steve could create a “Kashmir” entity based on “what Kashmir means to him.” Everyone else could then do the same. And so can Steve’s veterinary doctor. Entities therefore empower the user with the ability to create private topics that might be extensions of broad taxonomies.

To take another example, a Pharmaceuticals researcher in a large Pharmaceutical company might be working on a new top-secret project (named “Gene Project”) on
Genomics. Because “Gene Project” is an internal project, it would likely not exist in a public taxonomy which could be used with the semantic browser of this the preferred embodiment of my invention. However, the researcher could create an entity named “Gene Project”, typed as a Project, and could then initialize the entity by scoping it to Genomics (which exists in broad taxonomies) and then also qualifying it with the keyword-phrase “Gene Project” (using the AND operator). Essentially, this is akin to defining “Gene Project” as anything on Genomics that has the phrase “Gene Project.” This will impose much stricter context than merely using the keywords “Gene Project” (which might return results that contain the word “Project” but have nothing to do with Genomics). By defining a personal topic, “Gene Project” that is scoped to Genomics but also extends “Gene Project” with a specific qualifier, the researcher now has much more precise and personal context. The entity can then be dragged and dropped, copied and pasted, etc. to create requests (e.g., “Experts on Gene Project.”) At runtime, the server-side semantic query processor will interpret this (by mapping the SQML to the semantic network) as “Experts on any information that belongs to the category Genomics AND which also includes the phrase “Gene Project.”

[1774] 2. Entities also allow the user to create a dynamic taxonomy—public taxonomies are very static and are not updated regularly. With entities, the user can “extend” his/her private taxonomy dynamically and at the speed of thought. Knowledge is transferred at the speed of thought. Entities allow the user to create context with the same speed and dynamism as his/her mind or thought flow. This is very significant. For instance, the user can create an entity for a newly scheduled meeting, a just-discovered conference, a new customer, a newly discovered competitor, etc.—ALL AT THE SPEED OF THOUGHT. Taxonomies don’t allow this.

[1775] 3. Taxonomies assume that topics are the only source of context. With entities, a user can create abstract contextual definitions that include—but are not limited to—topics. Examples include people, teams, events, companies, etc. Entities might eventually “evolve” into topics in a taxonomy (over time and as those entities gain “fame” or “notoriety”) but in the “short-term,” entities allow the user to create context that has not yet evolved (or might never evolve) into a full-blown taxonomic entry. For instance, Nervana (our company) was initially an entity (known only to itself and its few employees) but as we have grown and attracted public attention, as an entity we are evolving into a topic in a public taxonomy. With entities, users don’t have to wait for context (like Nervana) to “eventually become” topics.

[1776] 4. Entities allow the user to create what the inventor calls “compound context.” An example of this is a meeting. A meeting typically involves several participants with documents, presentation slides, and/or handouts relevant to the topic of discussion. With entities in the Information Nervous System, a user can create a “meeting” context that captures the semantics of the meeting. Using the Create Entity Wizard, the user can specify that the entity is a meeting, and then specify the semantic filters. Consider an example of a project meeting with five participants and 2 handed out documents, and one presentation slide. The Presenter of the meeting might want to create an entity in order to track knowledge specifically relevant to the meeting. For instance, he/she might want to do this to determine when to schedule a follow-up meeting or to track specific action items relating to the meeting. To create the entity, the user would add the email addresses of the participants, the handed out documents, and also the presentation to the entity filter definition. The user then saves the entity which is then created in the semantic namespace/environment. The user can then edit the entity with new or removed filters (and/or a new name/description) at a later date/time—for instance, if he/she has discovered new documents that would have been relevant to the meeting. When the user drops and drops the entity or includes it in a request/agent, the semantic browser will compile the entity and includes it in its master SQML with the sub-queries also passed to the XML Web Service for interpretation. The server-side semantic query processor then processes the compound SQML by constructing a series of SQL sub-queries (or an equivalent) and by joining these queries with the entity sub-queries which in turn are generated using SQL sub-queries.

[1777] The user can use an AND or OR (or other) operator to indicate how the entity filters should be applied. For instance, the user can indicate that the meeting (semantically) is the participants of the meeting AND the documents/slides handed out during the meeting. When the entity is compiled at the client and the server, the SQML equivalent is used to interpret the entity (with the desired operator). This is very powerful. It means that the user can define an entity named “Project Meeting” and drag and drop that entity to the special agent named “Breaking News.” This then creates a request named “Breaking News on Project Meeting” (with the appropriate SQML referring to the identifier of the entity—which will then be compiled into sub-SQML before it is passed to the server(s) for interpretation. The server then applies default predicates to the entries in the entity (based on what “makes sense” for the object). In this particular example, because of the definition of the entity, the server will then only return:

[1778] Breaking News BY ALL the participants AND which is ALSO semantically relevant TO ALL the documents/slides

[1779] For instance, this will only return conversations/threads that involve all the participants of the meeting and which are semantically relevant to all the handouts given out during the meeting. This is precisely what the user desired (in this case) and the semantic browser would have empowered the user to essentially construct a rather complex query.

[1780] Even more complex queries are possible. Entities can include other entities to allow for compound entities. For instance, if an entire team of people were involved in the meeting, the Presenter might want to create an entity that includes an email distribution list of those people. In this case, the user might search the Information Nervous System for the distribution list and then save the result as an entity. The browser will allow the user to save results as entities and based on the result type, it will automatically create an entity with a default entity type that “makes sense.” For instance, if the user saves a document result as an entity, the semantic browser will create a “Topic” entity. If the user saves a Person result as an entity, the semantic browser will create a “Person” entity. If the user saves an email distribution list as an entity, the semantic browser will create a “Team” entity.

[1781] In this example, the user can save a Person result as a Person entity and then drag and drop that entity into the Project Meeting entity. The Team entity that maps to the email distribution list of the meeting participants can be dragged and dropped to the Project Meeting entity. The user can then create a request called “Headlines on Project Meeting” that includes the entity. The semantic query processor will then
return Headlines BY anyone in the email distribution list (using the right default predicate) and which is semantically relevant to ALL the handouts given out during the meeting. Similarly, a Dossier (Guide) on the Project Meeting will return All Bets on the meeting, Best Bets on meeting, Experts on the meeting, etc.

[1782] Note that such a compound entity that includes other entities gets checked by the client-side semantic consistency checker for referential integrity. In other words, if Entity A refers to Entity B and the user attempts to delete Entity B, the semantic browser will detect this and flag the user that Entity B has an outstanding reference. If the user deletes Entity B anyway, the reference in Entity A (and any other references to Entity B) will get removed. Generally, in some embodiments, the user could be prohibited (whether informed or not) from deleting Entity B in the same situation, based on permissions of others within an organization associated with the entity. For example, employers could monitor activities of employees for risk management purposes, like as is done with email in some companies, only much potentially much more powerfully (Of course, appropriate policies and privacy considerations would have to be addressed). The same process applies to Request Collections (Blenders), Portfolios (Entity Collections—see below), and other compound items in the semantic namespace/environment (items that could refer to other items in the namespace/environment).

[1783] Popular entities can also be shared amongst members of a knowledge community. Like other items in the semantic browser (like requests or knowledge communities (agencies), entities can be saved as files (so the user can later open them or email them to colleagues, or save them on a central file share, etc.). A common scenario would be that the corporate Librarians at businesses would create entities that map to internal projects, meetings, seminars, tasks, and other important corporate knowledge items of interest. These entities would then be saved on a file-share or other sharing mechanism (like a portal or web-site) or on a knowledge community (agency). The knowledge workers in the organization would then be able to use the entities. As the entities get updated, in the preferred embodiment the Librarians can and will automatically edit their context and users will be able refresh or synchronize to the new entities. Entities could also and alternately be shared on a peer-to-peer basis by individual users. This is akin to a legal peer-to-peer file sharing for music, but instead of music, what is shared is context to facilitate meaning, or more meaningful communication.

[1784] Portfolios (or Entity Collections)

Portfolios are a special type of entity that contains a collection of entities. In the preferred embodiment, to minimize complexity and confusion (at least of nomenclature or terminology), while an entity can be of any size or composition, and portfolio can contain any kind or number of entities, a portfolio would not contain other portfolios. A portfolio allows the user to manage a group of entities as one unit. A portfolio is a first-class entity and as such has all the aforementioned features of an entity. When a portfolio is used as a parameter in a smart request, the OR qualifier is applied (by default) to its containing entities. In other words, if Portfolio P contains entities E1 and E2, a smart request titled ‘Headlines on P’ will be processed as ‘Headlines on E1 or E2.’ The user can change this setting on individual smart requests (to an AND qualifier).

[1786] Sample Scenarios
[1787] Again, in reviewing the scenarios below, it is helpful to recall that, conceptually, the system can gather more relevant information in part because it “knows” who is asking for it, and “understands” who that person or group is, and the kinds of information they are probably interested in. Of course, strictly speaking, the system is not cognitive or self-aware in the full human sense, and the operative verbs in the preceding sentence are conceptual metaphors or similes. Still, in operation and results, it mimics understanding and knowledge to an unprecedented degree in part because of its underlying semantically-informed architecture and execution.

[1788] This point can be illustrated by a simplistic contrast: If two very different people entered the exact same search at the exact same time into a search engine such as Google™, they would get the exact same results. In contrast, with the preferred embodiment of the present system, if those same two people entered the same request via an Entity, each would get different results tailored to be relevant to each.

[1789] To appreciate some of the potential power of this feature, it is useful to note that while the system or Entities “know” who is posing the query, the Entities do not depend for that knowledge on the user informing them and keeping them constantly updated and informed (although user information can be supplied and considered at any time). If that were the case, the system could be too labor intensive to be efficient and useful in many situations; it would just be too much work. Instead, the Entities “know” who the requester is by inference and from semantics from characteristics sometimes supplied by others, sometimes derived or deduced, sometimes collected from other requests and the like, as explained throughout this application and its parent application.

[1790] Some example scenarios of Entities in operation:

[1791] A pharmaceuticals ‘patent’ entity could include the categories of the patent, relevant keywords, and relevant documents.

[1792] A CIA agent could create a ‘terrorist’ entity to track terrorists. This could include categories on terrorism, suspicious wire transfers, suspicious arms sales, classified documents, keywords, and terrorism experts in the information community.


[1794] 4. Handle Lines on all of my competitors (this is done by creating the competitor entities, and then creating a smart request with the entities as parameters using the OR qualifier with each predicate).

[1795] 5. Find Experts on my investment portfolio companies (create the individual entities, create a portfolio containing these entities and then create a smart request that has the ‘Experts’ context template and that uses the portfolio as an argument).

[1796] 6. Open a Dossier (Guide) on my competitors (create the individual competitor entities, create a portfolio containing these entities and then create a smart request that has the ‘Dossier’ or ‘Guide’ context template and that uses the portfolio as an argument).

H. Knowledge Community Browsing and Subscription Specification for the Information Nervous System

[1797] Overview

[1798] The Nervana semantic browser will allow the user to subscribe and unsubscribe to/from knowledge communities (agencies) for a given profile. These knowledge communities
will be readily available to the user underneath the profile entry in the semantic environment. In addition, these knowledge communities will be queried by default for intrinsic alerts, context panels, and etc. whenever results are displayed for any request created using the same profile.

The semantic environment includes state indicating the subscribed knowledge communities for each profile. The client-side semantic query processor (SQP) uses this information for dynamic requests that start from results for requests of a given profile (the SQP will ask the semantic runtime client for the knowledge communities for the profile and then issue XML Web Service calls to those knowledge communities as appropriate).

In a preferred embodiment, a user interface for the knowledge community subscription and un-subscription can be based wherever the dialog box has combo boxes allowing the user to filter by profile, to view all, new, subscribed, suggested, and unsubscribed communities, by industry and area of interest, by keywords, by publishing point (all publishing points, the local area network, the enterprise directory, and the global knowledge community directory), and by creation time (anytime, today, yesterday, this week, and last week). The semantic runtime client queries the publishing point endpoints (for each publishing point) using the filters. It then gathers the results and displays them in the results pane. The user is also able to view the categories of each knowledge community in the results pane via a combo box. FIG. 20B illustrates the bottom portion of the Knowledge Communities Dialog Box.


In the preferred embodiment, the Nervana Semantic DHTML Behavior is an Internet Explorer DHTML Behavior that, from the client's perspective, everything it understands as a query document. The client opens 'query documents,' in a manner resembling how a word processor opens 'textual and compound documents.' The Nervana client is primarily responsible for processing a Nervana semantic query document and rendering the results. A Nervana semantic query document is expressed and stored in form of the Nervana Semantic Query Markup Language (SQML). This is akin to a "semantic file format."

In the preferred embodiment, the SQML semantic file format comprises of the following:

- **Head**—The 'head' tag, like in the case of HTML, includes tags that describe the document.
- **Title**—The title of the document.
- **Comments**—The comments of the document.
- **UserName**—The username of the document creator.
- **SystemName**—The system name of the device on which the document was created.
- **Subject**—The subject of the document.
- **Creator**—The creator of the document.
- **Company**—The company in which the document was created.
- **RequestType**—This indicates the type of request. It can be "smart request" (indicating requests to one or more information community web services) or "dumb request" (indicating requests to one or more local or network resources).

- **ObjectType**—This fully qualifies the type of objects returned by the query.
- **URI**—The location of the document.
- **CreationTime**—The creation time of the document.
- **LastModifiedDate**—The last modified time of the document.
- **LastAccessedTime**—The last accessed time of the document.
- **Attributes**—The attributes of the document, if any.
- **RevisionNumber**—The revision number of the document.
- **Language**—The language of the document.
- **Version**—This indicates the version of the query. This allows the web service's semantic query processor to return results that are versioned. For instance, one version of the browser can use V1 of a query, and another version can use V2. This allows the web service to provide backwards compatibility both at the resource level (e.g., for agents) and at the link level.
- **Targets**—This indicates the names and the URLs of the information community web services that the query document targets.

Type—This indicates the type of targets. This can be "targetentries," in which case the tag includes subtags indicating the actual web service targets, or "allsubscribedtargets," in which case the query processor uses all subscribed information communities.

Categories—This indicates the list of category URLs that the query document refers to. Each category entry contains a name attribute and a URI attribute that indicates the URL of the Knowledge Domain Server (KDS) from which the category came.

Type—This indicates the type of categories. This can be either "categoryentries," in which case the subtag refers to the list of category entries, "allcategories," in which case all categories are requested from the information community web services, or "myfavoritecategories," in which case the query processor gets the user's favorite categories and then gathers compiled SQML that contains these categories (this compiled SQML is then sent to the server(s)).

Query—This is the parent tag for all the main query entries of the query document.

Resource—The reference to the 'dumb' resource being queried. Examples include file paths, URLs, cache entry identifiers, etc. These will be mapped to actual resource managers components by the interpreter.

Type—The type of resource reference, qualified with the namespace. Examples of defined resource reference types are: nervana:url (this indicates that the resource reference is a well-formed standard Internet URL, or a custom Nervana URL like 'agent://. . .'), nervana:filepath (this indicates that the resource reference is a path to a file or directory on the file system), and nervana:namespacedef (this indicates that the resource comes from the client semantic namespace).

URI—This indicates the universal resource identifier of the resource. In the case of paths and Internet URLs, this indicates the URL itself. In the case of namespace entries, this indicates the GUID identifier of the entry.
Mid—This indicates the metadata identifier, which is used by the SQML interpreter to map the resource to the metadata section of the document. The metadata id is mapped to the same identifier within the metadata section.

Arg—This indicates the arguments of the resource identifier.

Links—this indicates the reference to the semantic links (for “targets” only).

Type—this indicates the type of links. This can be “linkentries,” indicating the links are explicit entries.

LinkEntries—this indicates the details of a link entry.

Predicate—this indicates the type of predicate for the link. For instance, the predicate “nervana:relevantto” indicates that the query is “return all objects from the resource R that are relevant to the object O,” where R and O and the specified resource and object, respectively. Other examples of predicates include nervana:reportsto, nervana:teammateOf, nervana:from, nervana:to, nervana:cc, nervana:bcc, nervana:attatchto, nervana:sentby, nervana:sentto, nervana:postedon, nervana:containsText, etc.

Type—this indicates the type of object reference indicates in the “Link” tag. Examples include standard XML data types like xml:string, xml:integer. Nervana equivalents of same, custom Nervana types like nervana:datetimeref (which could refer to object references like “today” and “tomorrow”), and any standard Internet URL (HTTP, FTP, etc.) or Nervana URL (objects://, etc.) that refers to an object that Nervanta can process as a semantic XML object.

Metadata—this contains the references to the metadata entries.

MetadataEntry—this indicates the details of a metadata entry.

Mid—this indicates the metadata identifier (GUID).

Value—this indicates the metadata itself.

Example

Documents (Information or Context-Based)
J. Semantic Client-Side Runtime Control API Specification for the Information Nervous System

[1851] The client SQML parser performs "lazy" updating of metadata in the SQML. When the request is invoked, it attempts to update the metadata of all parameters (resources, etc.) in the SQML to handle the case where the objects might have changed since they were used to create the relational request. If the object does not exist, the client uses the metadata it already has. Otherwise, it updates it and uses the updated metadata. That way, even if the object has been deleted, the user experience is not interrupted until the user actually tries to open the object from whence the metadata came.

2. SQML Generation

Preferably, SQML is generated in any one or more of several possible ways:

By creating a smart request

By creating a local request

By creating an entity

By opening one or more local documents in the semantic browser

By the client (dynamically)—in response to a drag and drop, smart copy and paste, intrinsic alert, context panel/link invocation, etc.

3. SQML Parsing

In some embodiments in some situations, SQML that gets created on the client might not be ready (in real-time) for remote consumption—by the server's XML web service or at another machine site. This is especially likely to be the case when the SQML refers to local context such as documents, Entities, or Smart Requests (that are identified by unique identifiers in the semantic environment). In the preferred embodiment, the client is able to create the SQML that is ready for remote consumption. Preferably, it does this by caching the metadata for all references in the metadata section of the document. This is preferable because in some cases, the resource or object to which the reference points might no longer exist when the query is invoked. For instance, a user might drag and drop a document from the Internet to a smart request in order to generate a new relational request. The client extracts the metadata (including the summary) from the link and inserts the metadata into the SQML. Because the resolution of the query uses only the metadata, the query is ready for consumption once the metadata is inserted into the SQML document. However, the link that the object refers to might not exist the day after the user found it. In such a case, even if the user invokes the relational request after the link might have ceased to exist, the request will still work because the metadata would already have been cached in the SQML.

Usage Scenario

A Nervana client application (for instance, the semantic browser) or a Nervana skin will call this method to open a namespace path in order for the user to navigate the namespace from within the semantic browser.

```c
PROTOTYPE
SCODE
EnumObjectsInNamespacePath(
    [in] BSTR Path,
    [in] LONG QueryMask,
    [out] BSTR *QueryRequestGuid );
```

b. CompileSemanticQueryFromBuffer

Introduction

The CompileSemanticQueryFromBuffer method opens an SQML buffer and compiles it into one or more
execution-ready SQML buffers. For instance, an SQML file containing a blender will be compiled into SQML buffers representing each blender entry. If the blender contains blend- ers, the blend will be unwrapped and an SQML buffer will be returned for each contained blender. A compiled or "execution-ready" SQML buffer is one that can be semantically processed by an agency. The implication is that a blender that has agents from multiple agencies will have its SQML compiled to buffers with the appropriate SQML from each agency.

[1860] Note: If the buffer is already compiled, the method returns S_FALSE and the return arguments are ignored.

Usage Scenario

[1861] A Nervana client application (for instance, the semantic browser) or a Nervana skin will call this method to compile an SQML buffer and retrieve generated "compiled code" that is ready for execution. In typical scenarios, the application or skin will compile an SQML buffer and then prepare frame windows where it wants each individual SQML query to sit. It can then issue individual SQML semantic calls by calling OpenSemanticQueryFromBuffer and then have the results displayed in the individual frames.

PROTOTYPE
SCODE
CompileSemanticQueryFromBuffer(
in] BSTR SQMLBuffer,
[in] DWORD Flags,
[out] DWORD *pcbNumCompiledBuffers,
[out] BSTR *pstrCompiledBuffers);

[1862] c. GetSemanticQueryBufferFromNamespace

Introduction

[1863] The GetSemanticQueryBufferFromNamespace method opens an SQML buffer and asynchronously fires the XML results (in SRML) onto the DOM, from whence a Nervana skin can sink the event. Note that in this embodiment the SQML has to be "compiled" and ready for execution. If the SQML is not ready for execution, the call will fail. To compile an SQML buffer, call CompileSemanticQueryFromBuffer.

Usage Scenario

[1864] A Nervana client application (for instance, the semantic browser) or a Nervana skin will call this method to open a compiled SQML buffer.

PROTOTYPE
SCODE
GetSemanticQueryBufferFromNamespace(
in] GUID ObjectID,
[in] BSTR Paths,
[out] BSTR *pstrSQMLBuffer);

[1871] f. GetSemanticQueryBufferFromURL

Introduction


Usage Scenario

[1873] A Nervana client application (for instance, the semantic browser) or a Nervana skin will call this method to convert an URL of any type to SQML. This can include file paths, HTTP URLs, FTP URLs, Nervana agency object URIs (prefixed by "wsobject://") or Nervana agency URIs (prefixed by "wsagency://").

PROTOTYPE
SCODE
GetSemanticQueryBufferFromURL(
in] BSTR URL;
[out] BSTR *pBuffer);

[1874] g. GetSemanticQueryBufferFromClipboard

Introduction

[1875] The GetSemanticQueryBufferFromClipboard method converts the clipboard contents to SQML, and returns the buffer.

Usage Scenario

[1876] A Nervana client application (for instance, the semantic browser) or a Nervana skin will call this method to get a semantic query from the clipboard. The application can then load the query buffer.
h. Stop

Introduction

The Stop method stops current open request.

Usage Scenario

A Nervana client application (for instance, the semantic browser) or a Nervana skin will call this method to stop a load request is just issued.

i. Refresh

Introduction

The Refresh method refreshes the current open request.

Usage Scenario

A Nervana client application (for instance, the semantic browser) or a Nervana skin will call this method to refresh the currently loaded request.

j. CreateNamespaceObject

Introduction

The CreateNamespaceObject method creates a namespace object and returns its GUID.

Usage Scenario

A Nervana client application (for instance, the semantic browser) or a Nervana skin will typically call this method to create a temporary namespace object when a new query document has been opened.

k. DeleteNamespaceObject

Usage Scenario

The DeleteNamespaceObject method deletes a namespace object.

l. CopyObject

Introduction

The CopyObject method copies the semantic object to the clipboard as an SXML buffer using a proprietary SXML clipboard format. The object can then be “pasted” onto agents for relational semantic queries, or used as a lens over other objects or agents.

Usage Scenario

A Nervana skin will typically call the CopyObject method when the user clicks on the “Copy” menu option—off a popup menu on the object.

m. CanObjectBeAnnotated

Introduction

The CanObjectBeAnnotated method checks whether the given object can be annotated.

Usage Scenario

A Nervana skin will typically call the CanObject-BeAnnotated method to determine whether to show UI indicating the “Annotate” command.

n. AnnotateObject

Introduction

The AnnotateObject method invokes the currently installed email client and initializes it to send an email annotation of the object to the email agent of the agency from whence the object came.

Usage Scenario

A Nervana skin will typically call the AnnotateObject method when the user clicks on the “Annotate” menu option—off a popup menu on the object.
Introduction

[1898] The SendEmailToObject method is called to send email to a person or customer object. The method opens the email client and initializes it with the email address of the person or customer object.

Usage Scenario

[1899] A Nervana skin will typically call the SendEmailToObject method when the user clicks on the “Send Email” menu option—off a popup menu on a person or customer object.

Introduction

[1900] A Nervana skin will typically call the CanObjectBePublished method to determine whether to show UI indicating the “Publish” command.

Usage Scenario

[1901] The CanObjectBePublished method checks whether the given object can be published.

p. PublishObject

Introduction

[1902] The PublishObject method invokes the currently installed email client and initializes it to send an email publication of the object to the email agent of the agency from whence the object came.

Usage Scenario

[1903] A Nervana skin will typically call the PublishObject method when the user clicks on the “Publish” menu option—off a popup menu on the object.

Introduction

[1904] The OpenObjectContents method opens the object using an appropriate viewer. For instance, an email object will be opened in the email client, a document will be opened in the browser, etc..

Usage Scenario

[1905] A Nervana skin will typically call the OpenObjectContents method when the user clicks on the “Open” menu option—off a popup menu on the object.

Introduction

[1906] A Nervana skin will typically call the CanObjectBePublished method to determine whether to show UI indicating the “Publish” command.

q. OpenObjectContents

Introduction

[1907] A Nervana skin will typically call the OpenObjectContents method when the user clicks on the “Open” menu option—off a popup menu on the object.

t. IsObjectMarkedAsFavorite

Introduction

[1908] The IsObjectMarkedAsFavorite method is called to check whether an object is marked as a favorite on the agency from whence it came.

Usage Scenario

[1909] A Nervana skin will typically call the IsObjectMarkedAsFavorite method when it wants to display the titles of the annotations an object has—for instance, in a popup menu or when it wants to display the annotations metadata in a window.

Introduction

[1910] The IsObjectMarkedAsFavorite method is called to get the annotations an object has on the agency from whence it came.

Usage Scenario

[1911] A Nervana skin will typically call the GetObjectAnnotations method when it wants to display the titles of the annotations an object has—for instance, in a popup menu or when it wants to display the annotations metadata in a window.

Introduction

[1912] A Nervana skin will typically call the GetObjectAnnotations method when it wants to display the titles of the annotations an object has—for instance, in a popup menu or when it wants to display the annotations metadata in a window.

r. SendEmailToPersonObject

Introduction

[1913] The SendEmailToPersonObject method is called to send email to a person or customer object. The method opens the email client and initializes it with the email address of the person or customer object.

Usage Scenario

[1914] The SendEmailToPersonObject method is called to check whether an object is marked as a favorite on the agency from whence it came.

Usage Scenario

[1915] A Nervana skin will typically call the SendEmailToPersonObject method when it wants to display the “Mark as Favorite” or the “Unmark as Favorite” command. If the object cannot be marked as a favorite (for instance, if it did not originate on an agency), the error code E_INVALIDARG is returned.

Introduction

[1916] A Nervana skin will typically call the SendEmailToPersonObject method when the user clicks on the “Send Email” menu option—off a popup menu on a person or customer object.
Introduction

[1917] The MarkObjectAsFavorite method is called to mark the object as a favorite on the agency from whence it came.

Usage Scenario

[1918] A Nervana skin will typically call the MarkObjectAsFavorite method when the user clicks on the “Mark as Favorite” command.

PROTOTYPE
SCODE
MarkAsFavorite( in ] BSTR ObjectSRML );

[1919] v. UnmarkObjectAsFavorite

Introduction

[1920] The UnmarkObjectAsFavorite method is called to unmark the object as a favorite on the agency from whence it came.

Usage Scenario

[1921] A Nervana skin will typically call the UnmarkObjectAsFavorite method when the user clicks on the “Unmark as Favorite” command.

PROTOTYPE
SCODE
UnmarkAsFavorite( in ] BSTR ObjectSRML );

w. IsSmartAgentOnClipboard

Introduction

[1922] The IsSmartAgentOnClipboard method is called to check whether a smart agent has been copied to the clipboard.

Usage Scenario

[1923] A Nervana skin will typically call the IsSmartAgentOnClipboard method when it wants to toggle the user interface to display the “Paste” icon or when the “Paste” command is invoked.

PROTOTYPE
SCODE
IsSmartAgentOnClipboard( );

x. GetSmartLensQueryBuffer

Introduction

[1924] The GetSmartLensQueryBuffer method is called to get the query buffer of the smart lens. This returns the SQML of the query that represents the objects on the smart agent that is on the clipboard, and which are semantically relevant to a given object.

Usage Scenario

[1927] A Nervana skin will typically call the GetSmartLensQueryBuffer method when the user hits “Paste as Smart Lens” to invoke the smart lens of the smart agent that is on the clipboard.

PROTOTYPE
SCODE
GetSmartLensQueryBuffer( 
[ in ] BSTR ObjectSRML, 
[ in ] LONG QueryMark, 
[ out ] BSTR *pQueryRequestGuid );

Y. OpenObjectContents

Introduction

[1928] The OpenObjectContents method opens the object using an appropriate viewer. For instance, an email object will be opened in the email client, a document will be opened in the browser, etc.

Usage Scenario

[1929] A Nervana skin will typically call the OpenObjectContents method when the user clicks on the “Open” menu option—or a popup menu on the object.

PROTOTYPE
SCODE
OpenObjectContents( [ in ] BSTR ObjectSRML );

3. Email Control APIs

a. Email_GetFromLinkObjects

Introduction

[1930] The Email_GetFromLinkObjects method is called to get the metadata for the “From” links on an email object from the agency from whence it came.

Usage Scenario

[1931] A Nervana skin will typically call the Email_GetFromLinkObjects method when it wants to navigate to the “From” list from an email object, or to display a popup menu with the name of the person in the “From” list.

PROTOTYPE
SCODE
Email_GetFromLinkObjects( 
[ in ] BSTR EmailObjectSRML, 
[ in ] LONG QueryMark, 
[ out ] BSTR *pQueryRequestGuid );

b. Email_GetToLinkObjects

Introduction

[1932] The Email_GetToLinkObjects method is called to get the metadata for the “To” links on an email object from the agency from whence it came.

Usage Scenario

[1933] A Nervana skin will typically call the Email_GetToLinkObjects method when it wants to navigate to the “To”
list from an email object, or to display a popup menu with the name of the person in the “To” list.

```
[1937] c. Email_GetCcListObjects

Introduction

[1938] The Email_GetCcListObjects method is called to get the metadata for the “CC” links on an email object from the agency from whence it came.

Usage Scenario

[1939] A Nervana skin will typically call the Email_GetCcListObjects method when it wants to navigate to the “CC” list from an email object, or to display a popup menu with the name of the person in the “CC” list.

```

```
[1940] d. Email_GetBccListObjects

Introduction

[1941] The Email_GetBccListObjects method is called to get the metadata for the “BCC” links on an email object from the agency from whence it came.

Usage Scenario

[1942] A Nervana skin will typically call the Email_GetBccListObjects method when it wants to navigate to the “BCC” list from an email object, or to display a popup menu with the name of the person in the “BCC” list.

```

```
[1943] e. Email_GetAttachmentListObjects

Introduction

[1944] The Email_GetAttachmentListObjects method is called to get the metadata for the “Attachment” links on an email object from the agency from whence it came.

Usage Scenario

[1945] A Nervana skin will typically call the Email_GetAttachmentListObjects method when it wants to navigate to the “Attachments” link from an email object, or to display a popup menu with the titles of the attachments in the “Attachments” list.

```

```
[1946] a. Person_GetDirectReports

Introduction

[1947] The Person_GetDirectReports method is called to get the metadata for the “Direct Reports” links on a person object from the agency from whence it came.

Usage Scenario

[1948] A Nervana skin will typically call the Person_GetDirectReports method when it wants to navigate to the “Direct Reports” link from a person object, or to display a popup menu with the names of the direct reports in the “Direct Reports” list.

```

```
[1949] b. Person_GetDistributionLists

Introduction

[1950] The Person_GetDistributionLists method is called to get the metadata for the “Member of Distribution Lists” links on a person object from the agency from whence it came.

Usage Scenario

[1951] A Nervana skin will typically call the Person_GetDistributionLists method when it wants to navigate to the “Member of Distribution Lists” link from a person object, or to display a popup menu with the names of the distribution lists of which the person is a member.

```

```
[1952] c. Person_GetInfoAuthored
The Person_GetInfoAnnotated method is called to get the metadata for the "Info Annotated by Person" links on a person object from the agency from whence it came.

Usage Scenario

A Nervana skin will typically call the Person_GetInfoAnnotated method when it wants to navigate to the "Info Annotated by Person" link from a person object, or to display a preview window with time-critical or recent information that the person annotated.

---

The Person_GetAnnotationsPosted method is called to get the metadata for the "Annotations Posted by Person" links on a person object from the agency from whence it came.

Usage Scenario

A Nervana skin will typically call the Person_GetAnnotationsPosted method when it wants to navigate to the "Annotations Posted by Person" link from a person object, or to display a preview window with time-critical or recent annotations that the person posted.

---

The Person_SendEmailTo method is called to send email to a person or customer object. The method opens the email client and initializes it with the email address of the person or customer object.

Usage Scenario

A Nervana skin will typically call the Person_SendEmailTo method when the user clicks on the "Send Email" menu option—off a popup menu on a person or customer object.

---

The OnBeforeQuery event is fired before the control issues a query to resources consistent with the current semantic request.

Usage Scenario

A Nervana client application (for instance, the semantic browser) or a Nervana skin will sink this event if it wants to cancel a query or cache state before the query is issued.

---

The OnQueryBegin event is fired when the control issues the first query to a resource consistent with the current semantic request.

Usage Scenario

A Nervana client application (for instance, the semantic browser) or a Nervana skin will sink this event if it wants to cache state or display status information when the query is in progress.
c. Event: OnQueryComplete

Introduction

The OnQueryComplete event is fired before the control issues a query to resources consistent with the current semantic request.

Usage Scenario

A Nervana client application (for instance, the semantic browser) or a Nervana skin will sink this event if it wants to cancel a query or cache state before the query is issued.

K. Security Specification for the Information Nervous System

1. Authorization

Introduction

The ‘People’ DSA will be initialized with an LDAP Directory URL and Group Name. The ‘Users’ DSA will also be initialized with an LDAP Directory URL and Group Name. Typically, the ‘Users’ will be a subset of ‘People.’ For instance, a pharmaceuticals corporation might install a KIS for different large pharmaceutical categories (e.g., Biotechnology, Life Sciences, Pharmaco, etc.). Each of these will have a group of users that are knowledgeable or interested in that category. However, the KIS will also have the ‘People’ group populated with all employees of the corporation. This will enable users of the KIS to navigate to members of the entire employee population even though those members are not users of the KIS. In addition, the inference engine will be able to infer expertise with semantic links off people that are in the corporation, not necessarily just users of the KIS.

This is also advantageous for access control at the KIS level—this complements or supplements access control provided by the application server at the Web service layer. The Users group will contain people that have access to the KIS knowledge. However, the People group will contain people that are relevant to the KIS knowledge, even though those people don’t have access to the KIS.

Both People and Users DSA populate the People table in the Semantic Metadata Store (SMS) and indicate the object type id appropriately. Note that preferably the passwords are NOT stored in the People table in the SMS.

The Users DSA also populates the User Authentication Table (UAT). This is an in-memory hash table that maps the user names to passwords. The server’s Web service will implement the IPasswordProvider interface as an equivalent. The implementation of the PasswordProvider object will return the password that maps to a particular user name. The C# example below illustrates this:

```csharp
class PasswordProvider : IPasswordProvider
{
    public string GetPassword( string username )
    {
        return "password";
    }
}
```

The following C# code shows how the Web service can retrieve the user information after the user has been authenticated:
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using Microsoft.WSDK.Security;
using Microsoft.WSDK;
namespace WSDK__Security
{
    {
        [WebMethod]
        public string PersonalHello()
        {
            string response = "";
            SoapContext requestContext = HttpContext.Context;
            if (requestContext == null)
            {
                throw new ApplicationException("Non-SOAP request.");
            }
            foreach (SecurityToken tok in requestContext.Security.Tokens)
            {
                if (tok is UsernameToken)
                {
                    response = "Hello " + ((UsernameToken)tok).Username;
                }
            }
            return response;
        }
    }

    [WebMethod]
    public string ProcessSemanticQuery(string Query)
    {
        SoapContext requestContext = HttpContext.Context;
        if (requestContext == null)
        {
            throw new ApplicationException("Non-SOAP request.");
        }
        if (requestContext.Security.Tokens.Count == 1)
        {
            foreach (SecurityToken tok in requestContext.Security.Tokens)
            {
                if (tok is UsernameToken)
                {
                    UsernameToken UserToken = (UsernameToken)tok;
                    if (UserToken.PasswordOption == PasswordOption.SendHashed)
                    {
                        return ProcessSemanticQueryInternal(Query, UserToken.Username);
                    }
                    else
                    {
                        throw new SoapException("Invalid UsernameToken password type.", SoapException.ClientFaultCode);
                    }
                    else
                    {
                        throw new SoapException("UsernameToken security token required.", SoapException.ClientFaultCode);
                    }
                }
            }
        }
    }
}

[1987] b. Validating the UsernameToken on the Server


[1988] Although the WSDK verifies the Security header syntax and checks the password hash against the password from the Password Provider, there is some extra verification that is preferably be performed on the request. For instance, the WSDK will not call the Password Provider if a UsernameToken is received that does not include a password element. If there is no password to check, there is no reason to call the password provider. This means we need to verify the format of the UsernameToken ourselves.

[1989] Another possibility is that there is more than one UsernameToken element included with the request. WS-Security provides support for including any number of tokens with a request that may be used for different purposes.

[1990] The code above can be modified for the Nervana Web method to verify that the UsernameToken includes a hashed password and to only accept incoming requests with a single UsernameToken. The modified code is listed below.

[1986] In order to create a UsernameToken for the request, the Nervana client has to pass the username and password as part of the SOAP request. The Nervana client can pass multiple tokens as part of the request — this is preferable for cases where the user’s identity is federated across multiple authentication providers. The Nervana client will gather all the user account information the user has supplied (including user name and password information), convert these to WS-Security tokens, and then issue the SOAP request. The client code will look like the following (reference: [http://www.microsoft.com]):

```csharp
localhost.Service1 proxy = new localhost.Service1();
UsernameToken clearTextToken
```
else
{
    throw new SoapException(
        "Request must have exactly one security token."
    );
}
return null;
}

[1991] 2. People Groups
[1992] The KIS will include metadata for people groups. These are not unlike user groups in modern operating systems. The People Group will be a Nervana first-class object (i.e., it will inherit from the Object class). In addition, the People Group schema will be as follows:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectID</td>
<td>String</td>
<td>The object id of the people group</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>The name of the people group</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>The description of the people group</td>
</tr>
<tr>
<td>URI</td>
<td>String</td>
<td>The URL of the people group - this uniquely identifies the group and in the preferred embodiment will be an LDAP URI</td>
</tr>
</tbody>
</table>

[1993] In most cases, people groups will map to user groups in directory systems (like LDAP). For instance, the KIS server admin will have the KIS crawl a configurable set of user groups. There will be a People DSA that will crawl the user groups and populate the People Groups and Users tables in the SMS. The People DSA will perform the following actions:

[1994] Create the group (if it doesn’t exist in the SMS) or update the metadata of the Group (if it exists).

[1995] Enumerate all the users in the group (at the source—an LDAP directory in the preferred embodiment).

[1996] For all the users in the group, create People objects (or update the metadata if the objects already exist in the SMS).

[1997] Update the semantic network (via the ’Semantic sublinks’ table in the SMS) by mapping the people objects to the group objects (using the BELONGS_TO_GROUP semantic link type). This ensures that the SMS has semantic links that capture group membership information (in addition to the groups and users themselves).

[1999] Identity Metadata Federation (IMF) refers to a feature wherein an Information Community (agency) is deployed over the Internet but is used to service corporate or personal customers. For instance, Reuters™ could set up an information community for all its corporate customers that depend on its proprietary content. In such a case where multiple customers share an information community (likely in the same industry), Reuters™ will have a group on the SMS for each customer. However, each of these customers would have to have its corporate directory mirrored on Reuters™ in order for people metadata to be available. This would cause problems, particularly from a security and privacy standpoint. Corporations will probably not be comfortable with having external content providers obtaining access to the metadata of their employees. IMF addresses this problem by having the internet-hosted information community (agency) host only enough metadata for authentication of the user. For instance, Reuters™ will store only the logon information for the users of its corporate customers in its SMS. When the semantic browser receives SRML containing such incomplete metadata, the client will then issue another query to the enterprise directory (via LDAP access or via UDDI if the enterprise directory metadata is made available through a Web services directory) to fetch the complete metadata of the user. This is possible because the externally stored metadata will have the identity information with which the remaining metadata can be fetched. Since the client fetches the remaining metadata within the firewall of the enterprise, the sensitive corporate metadata is not shared with the outside world.

[2002] In the preferred embodiment, the KIS will include and enforce access control semantics. The KIS employs a policy of “default access.” Default access here means that the KIS will grant access to the calling user to any metadata in the SMS, except in cases where access is denied. As such, the system can be extended to provide new forms of denial, as opposed to new forms of access. In addition, this implies that if there is no basis for denial, the user is granted access (this leads to a simpler and cleaner access control model).

[2003] The KIS will have an Access Control Manager (ACM). The ACM is primary responsible for generating a Denial Semantic Query (DSQ) which the SQP will append to its query for a given semantic request from the client. The ACM will expose the following method (C# sample):

[2004] String GetDenialSemanticQuery(String CallingUserName)

[2005] Preferably, the method takes in the calling user name and returns a SQL query (or equivalent) that encapsulates exception objects. These are objects that must not be returned to the calling user by the SQP (i.e., objects for which the user does not have access).

[2006] The SQP then builds a final raw query that includes the denial query as follows:

[2007] Aggregate Raw Query AND NOT IN (Denial Query)

[2008] For example, if the aggregate raw query is:

[2009] SELECT OBJECTID FROM OBJECTS WHERE OBJECTTYPEID=5

[2010] and the denial query is:

[2011] SELECT OBJECTID FROM OBJECTS WHERE OWNERUSERNAME<>’JOHNDOE’

[2012] The final raw query (which is that the SQP will finally execute and serialize to SRML to return to the calling user) will be:

[2013] SELECT OBJECTID FROM OBJECTS WHERE OBJECTTYPEID=5 AND NOT IN (SELECT OBJECTID FROM OBJECTS WHERE OWNERUSERNAME<> ’JOHNDOE’)

[2014] Semantically, this is probably equivalent to:

[2015] “Select all objects that have an object type id of 5 but that are not in an object list not owned by John Doe.”

[2016] This in turn is probably semantically equivalent to:

[2017] “Select all objects that have an object type id of 5 that are owned by John Doe.”

[2019] Each semantic query processed by the semantic query processor (SQP) will contain an access control check. This will guarantee that the calling user only receives meta-
data that he/she has access to. The SQP will employ the following access control rules when processing a semantic query:

1. Preferably, if the query is for 'People' objects (people, users, customers, experts, newspaper, etc.), the returned 'People' objects must either:
   - Include the calling user, or
   - Include people that share at least one group with the calling user, and be owned by the calling user or the system.

2. Preferably, the corresponding denial query maps to the following rule: The returned objects must satisfy the following:
   - Is not the calling user
   - Is not owned by the calling user or the system
   - Has people that do not share any group with the calling user
   - Sample Denial Query SQL

The SQL below illustrates the access control denial query that will be generated by the ACM and appended by the SQP to enforce the access control policy. In this example, the name of the calling user is 'JOHINDOE'.

```sql
SELECT objectid FROM objects WHERE ownerusername = 'joehndoe'
```

2. If the query is for non-people objects (documents, email, events, etc.), the returned objects must:
   - Be owned by the calling user or the system, and
   - Be the subject of a semantic link with the calling user as the object, or
   - Be the object of a semantic link with the calling user as the subject, or
   - Be the subject of a semantic link with the object being a person that shares at least one group with the calling user, or
   - Be the object of a semantic link with the subject being a person that shares at least one group with the calling user.

```
SELECT objectid FROM objects WHERE ownerusername = 'joehndoe'
```

2. Preferably, the corresponding denial query maps to the following rule: The returned objects must satisfy the following:
   - Is not owned by the calling user
   - Is not owned by the system
   - Is not the subject of a semantic link with the calling user as the object
   - Is not the object of a semantic link with the calling user as the object
   - Is not the subject of a semantic link with the object being a person that shares at least one group with the calling user
   - Is not the object of a semantic link with the subject being a person that shares at least one group with the calling user.

```
SELECT objectid FROM objects WHERE ownerusername = 'joehndoe'
```

Sample Denial Query SQL

The SQL below illustrates the access control denial query that will be generated by the ACM and appended by the SQP to enforce the access control policy. In this example, the name of the calling user is 'JOHINDOE'.

```
SELECT objectid FROM objects WHERE ownerusername = 'joehndoe'
```

Sample Merged Denial Query SQL

By merging these two rules, the ACM returns the following merged query to the SQP for access denial:

```
SELECT objectid FROM objects WHERE ownerusername = 'joehndoe'
```

Example Scenario

For instance, a Reuters™ agency (KIS) might have people groups for each enterprise customer that Reuters™ serves. The agency will have a common information base (Reuters™ content) but will have people groups per enterprise customer. These groups might include competitors. As such, it is preferable to ensure that the knowledge flow, generation, and inference do not cross competitor boundaries. For instance, an employee of Firm A must not derive knowledge directly from an employee of Firm B that competes with Firm A, not must he or she derive knowledge indirectly (via inference). An employee of Firm A must not be able to get recommendations for items annotated by employees of Firm
B. Or an employee of Firm A must not be able to find experts that work for Firm B. Of course, this assumes that Firm A and Firm B are not partners in some fashion (in which case, they might want to share knowledge). In the case of knowledge partners, Reuters® would create a people group (likely via LDAP) that includes the people groups of Firm A and Firm B. The Reuters® KIS will then have the following people groups: Firm A, Firm B, and Firm A&B. The SMS will also include metadata that indicates that the people in Firms A and Firms B belong to these groups (via the “belongs to group” semantic link type). With this process in place, the aforementioned rules will guarantee that knowledge gets shared between Firms A and B.

[2072] c. Access Control Rules for Annotations

[2073] In the case of annotations, the calling user will be editing the semantic network, as opposed to querying it. In this case, the following rules would apply:

[2074] 1. Preferably, if the object being annotated is a Person object, the object must either be:

[2075] The calling user, or

[2076] A person that shares at least one people group with the calling user, and be owned by the calling user or the system

[2077] 2. Preferably, if the object being annotated is a non-Person object (e.g., a document, email, event, etc.), the object must either be:

[2078] Owned by the calling user

[2079] Owned by the system

[2080] Sample Denial Query SQL

[2081] The SQL below illustrates the access control denial query that will be generated by the ACM (for checking access control for annotations) and appended by the SQP to enforce the access control policy. In this example, the name of the calling user is ‘JOHINDOE’.

[2082] select objectid from objects where
[2083] ownerusername< > ’johindoe’ or
[2084] ownerusername< > ’system’ or
[2085] objectid not in (select objectid from people where name=’johindoe’) or
[2086] objectid not in (select objectid from semanticlinks where objecttypeid=’person’ and predicatedatatypeid=’belongs_to_group’ and objectid in (select objectid from semanticlinks where objectid in (select objectid from people where name=’johindoe’))

[2087] Access Control Enforcement

[2088] The ACM enforces access control for annotations and other write operations on the KIS. The KIS XML Web Service exposes an annotation method as follows (C# sample):

[2089] AnnotateObject(String CallingUserName, String ObjectID);

[2090] This method calls the ACM to get the denial query. Then it creates a final query as follows:

[2091] Annotation Object Query AND NOT IN (Denial Query)

[2092] In the preferred embodiment, the annotation object query is always of the form:

[2093] SELECT OBJECTID FROM OBJECTS WHERE OBJECTID=ObjectID,

[2094] where ObjectID is the argument to the AnnotateObject method.

[2095] The ACM then builds a final access control query SQL and uses this SQL to check for access control. Because the ACM does not have to return the SQL, it merely invokes it directly in order to check for access control. In addition, because it is a binary check (access or no access), the ACM merely checks whether the denial query returns at least one row. For instance, a final query might look like:

[2096] SELECT OBJECTID FROM OBJECTS WHERE OBJECTID=ObjectID AND NOT IN (SELECT OBJECTID FROM OBJECTS WHERE OWNERUSERNAME< >’JOHINDOE’)

[2097] The ACM then runs this query (via the SQL query processor) and asks for the count of the number of rows in the result set. If there is one row, access is granted, else access is denied. This model is implemented this way in order to have consistency with the denial query model (the ACM always builds a denial query and uses this as a basis for all access control checks).

L. Deep Information Specification for the Information Nervous System

[2098] Deep Information Overview

Introduction

[2099] In the preferred embodiment, the Nervana ‘Deep Info’ tool is aimed at providing context-sensitive story-like information for a Nervana information object. Deep Info essentially provides Nervana users with information that otherwise would be lost, given a particular context. By way of rough analogy, Deep Info is like the contextual information that gets displayed on music videos on MTV (showing information on the current artist, the current song, and in some case, the current musical instrument in the song).

[2100] The ‘deep’ in ‘deep info’ refers to the fact that the contextual information will often span multiple “hops” in the semantic network on the agency from whence the object came. ‘Deep Info’ is comprised of ‘deep info nuggets’ which can either be plan textual metadata or metadata with semantic query links (via SQL).

[2101] In the preferred embodiment, there are at least five kinds of Deep Info nuggets:

[2102] 1. Basic Semantic Link Nuggets

[2103] 2. Context Template Nuggets

[2104] 3. Trivia Nuggets

[2105] 4. Matchmaker Nuggets

[2106] 5. Recursive Nuggets

[2107] a. Basic Semantic Link Nuggets

[2108] With basic semantic link truths, deep info nuggets merely convey a semantic link of the current object. These nuggets involve a semantic link distance of 1. In this case, there is overlap with what will be displayed in the ‘Links’ context/task pane. Examples are:

[2109] Patrick Schmitz reports to Nosa Omoigui

[2110] Patrick Schmitz has 5 Direct Reports

[2111] Patrick Schmitz annotated 47 objects

[2112] Patrick Schmitz authored 13 objects

[2113] Patrick Schmitz was copied on 56 email objects

[2114] b. Context Template Nuggets

[2115] Context template nuggets display contextual information for each relevant context template, based on the information at hand. These nuggets are identical to those that will be displayed in the context bar or context panel for each type of context template. For example:

[2116] Patrick Schmitz posted 3 breaking news items

[2117] Patrick Schmitz posted 14 classics

[2118] Patrick Schmitz authored 7 headlines
Patrick Schmitz is involved in 13 discussions
Patrick Schmitz is a newsmaker on 356 objects
c. Trivia Nuggets
For all email objects on an agency:
Steve Judkins appears on the ‘To’ list of all of them
Steve Judkins replied to 23% of them
Patrick Schmitz annotated 50% of them
Only 3 of these have a thread depth greater than 2
For all people objects on an agency:
Patrick Schmitz has sent email to 47% of them
14% of them report to Nosa Omoigui
Sally Smith has had discussions with 85% of them
12% of them are newsmakers on at least one topic
All of them have been involved in at least one discussion this week
33% of them are experts on at least one topic
8% of them are experts on more than three topics
For a given distribution list on an agency:
Steven Judkins has posted the most email to this list
Sarah Trent has replied to the most email on this list
Nosa Omoigui has never posted to this list
Patrick Schmitz has posted 87 messages to this list this month
Richard Novotny has posted 345 messages to this list this year
For all distribution lists on an agency:
Steven Judkins has posted the most email to all lists
Lisa Heibron has replied to email on only 2% of the lists
Nosa Omoigui has never posted to any list
Pat Schnitz has posted at least once every week to all the lists
Richard Novotny has posted messages on 3 lists
For all information objects on an agency:
Steven Judkins has been the most prolific publisher (he published 5% of them)
Sally Smith has been the most prolific annotator (she annotated 2% of them)
Nosa Omoigui has been the most active newsmaker
Patrick Schmitz has the most aggregate expertise
Steve Judkins has the most expertise for information published this year
Gavin Schmitz has been involved in the most discussions (12% of them)
Richard Novotny has been involved in the most discussions this month (18% of them)
d. Matchmaker Nuggets
Person to Person
Semantic Link Based
Patrick Schmitz has sent mail to 13 people
47 people have appeared on same To list as Patrick Schmitz
47 people have appeared on same CC list as Patrick Schmitz
89 people in total have been referenced on email sent by Patrick Schmitz
24 people have annotated the same information as Patrick Schmitz
3 people are on all the same distribution lists as Patrick Schmitz
29 people are on at least one of Patrick Schmitz’s distribution lists
Context-Template Based
12 people have expertise on the same information categories as Patrick Schmitz
14 people and Patrick Schmitz are newsmakers on the same information items
27 people are in discussions with Patrick Schmitz
Information to Person
Semantic Link Based
Patrick Schmitz posted this information item
Steve Judkins authored this information item
This information item was copied to 2 people
3 people annotated this information item
Context Template Based (similar to context template nuggets)
There are 4 experts on this information item
There are 27 newsmakers on this information item
Information to Information
Context Template Based (similar to context template nuggets)
There are 578 relevant ‘all bets’
There are 235 relevant ‘best bets’
There are 4 relevant breaking news items
There are 46 relevant headlines
Semantic Link Based (via people)
There are 21 information items that have the same experts with this one
There are 23 information items that have the same newsmakers with this one
There are 34 information items posted by the same person that posted this one
There are 34 information items authored by the same person that authorized this one
There are 44 information items annotated by people that annotated this one
e. Recursive Nuggets
With recursive nuggets, displaying deep info on the subject of the current information nugget forms a contextual hierarchy. The system then recursively displays the nuggets based on the object type of the subject. With recursive nuggets, the system essentially probes the semantic network starting from the source object and continues to display nuggets along the path of the network. Probing is preferably stopped at a depth that is consistent with resource limitations and based on user feedback.
Another way to think of recursive nuggets is like a contextual version of an business organization chart. However, with Deep Information in the Information Nervous System, users will be able to browse a tree of KNOWLEDGE, as opposed to a tree of INFORMATION. To take an example, if a user selects an object and a tree view will show up like what is displayed below:

Example with document as context:

Newsmakers on ‘Title of document’
Gavin Schmitz
Reports To ->
Steve Judkins
Experts Like Steve Judkins ->
Nosa Omoigui
Patrick Schmitz
Interest Group Like Steve Judkins ->
Patrick Schmitz
Chuck Johnson
In the preferred embodiment, recursive nuggets will most typically be displayed via a drill-down pane beside each result object in the semantic browser. This will allow the user to select a result object and then recursively and semantically “explore” the object (as illustrated above).

Also, each header item in the Deep Info drill down tree view will be a link to a request (e.g., Experts Like Steve Judkins), and each result will be a link to an entity. For example, users will be able to “navigate” to the “person” (semantically) Patrick Schnitz from anywhere in the Deep Info tree view. Users will then be able to view a dossier on Patrick Schnitz, copy Patrick Schnitz, and Paste it on, say, Breaking News—in order to open a request called Breaking News by Patrick Schnitz. Again, notice the use of a default predicate based on the Person subject (“BY”).

The preferred embodiment Presenter Deep Info tree view (with support from the semantic runtime API in the semantic browser) will keep track of those links that are requests and those links that are result objects; that way, it will intelligently interpret the user’s intent when the user clicks on a link the tree view (it will navigate to a request or navigate to an entity).

M. Create Request Wizard Specification for the Information Nervous System

Introducing the Create Request Wizard Overview

The preferred embodiment Create Request (or Smart Agent) Wizard allows the user to easily and intuitively create new requests that represent semantic queries to be issued to one or more knowledge sources (running the Knowledge Integration Service).

Wizard Page 1: Select a Profile and Request Type: This page allows the user to select what profile the request is to be created in. The page also allows the user to select the type of request he/she wants to create. This type could be a Dossier (Guide) which will create a request containing sub-requests for each context template (based on the filters indicated in the request), knowledge types (corresponding to context templates such as Best Bets, Headlines, Experts, Newsmakers, etc.), information types (corresponding to types such as Presentations, General Documents, etc.), and request collections which are Blenders and allow the user to view several requests as a cohesive unit.

Wizard Page 2: Select Knowledge Communities (Agencies): This page allows the user to select which knowledge communities (running on Knowledge Integration Servers (KISes) the request should get its knowledge from. The user can indicate that the request should use the same knowl-
edge communities as those configured in the selected profile. The user can alternatively select specific knowledge communities.

[2205] Wizard Page 3: Select Filters: This page allows the user to select which filters to include in the request. Filters can include one or more of the following: keywords, text, categories, local documents, Web documents, email addresses (for People filters), and Entities. In alternate embodiments, other filter types will be supported. The property page also allows the user to select the predicate with which to apply a specific filter. Preferably, the most common predicate that will be exposed is “Relevant to.” Other predicates can be exposed consistent with the filter type (for instance a filter that refers to a Person via an email address or entity will use the default predicate “BY” if the requested type is not ‘People’—e.g., Headlines BY John Smith and will use the default predicate “LIKE” if the request type is ‘People’—e.g., Experts LIKE John Smith). The property page also allows the user to select the operation with which to apply the filters. The two most common operators are AND (in which case only results that satisfy all the filters are returned) and OR (in which case results that satisfy any of the filters are returned).

[2206] Wizard Page 4: Name and describe this request: This page allows the user to enter a name and description for the request. The wizard automatically suggests a name and description for the request based on the semantics of the request. Examples include:

[2208] 2. Experts from R&D on Encryption Techniques OR on User Interface Design, etc.
[2210] 4. Dossier on Data Mining AND on Web Development.

[2211] The user is allowed to override the suggested name/ description. The suggestions are truncated as needed based on a maximum name and description length.

[2212] The semantic browser also exposes the properties of an existing request via a property sheet. This allows the user to “edit” a request. The property sheet exposes the same user interface as the wizard except that the fields are initialized based on the semantics of the request (by de-serializing the request’s SQML representation).

N. Create Profile Wizard Specification for the Information Nervous System

[2213] Introducing the Create Profile Wizard

Overview

[2214] The Create Profile Wizard allows the user to easily and intuitively create new user profiles.

[2215] Wizard Page 1: Select your areas of interest: This page allows the user to select his/her areas of interest. This allows the semantic browser to get some high-level information about the user’s knowledge interests (such as the industry he/she works in). This information is then used to narrow category selections in the categories dialog, recommend new knowledge communities (agencies) configured with knowledge domains consistent with the user’s area(s) of interests, etc.

[2216] Wizard Page 2: Select your knowledge communities: This page allows the user to subscribe to knowledge communities for the profile. This allows the semantic browser to “know” which knowledge sources to issue requests to, when those requests are created for the profile. The semantic browser also uses the knowledge communities in the profile when it invokes Visualizations, semantic alerts, the smart lens (when the lens is a request/agent for the given profile), the object lens (when the target object is a result from the given profile), when the user drags and drops (or copies and pastes) an object to a request/agent for the given profile, etc. See FIG. 459.

[2217] In a preferred embodiment, the user can enter a name and description for the profile. The page allows the user to indicate whether the profile is preferably made the default profile. The default profile is used when the user does not explicitly indicate a profile in any operation in the semantic browser (for example, dragging and dropping a document from the file system to the icon representing the semantic browser will open a bookmark with that document from the default profile, whereas dragging and dropping a document to an icon representing a specific profile will open a bookmark with that profile).

O. Create Bookmark Wizard Specification for the Information Nervous System

[2218] 1. Introducing the Create Bookmark Wizard

Overview

[2219] The Create Bookmark (or Local/Dumb Request Agent) Wizard allows the user to easily and intuitively create new bookmarks (local/dumb requests) to view local/Web documents, entities, etc. in the semantic browser via which he/she can get access to the toolbox of the system (i.e., drag and drop, smart copy and paste, smart lens, smart alerts, Visualizations, etc.).

[2220] Wizard Page 1: Select a Profile and Request Type: This page allows the user to select what profile the bookmark is to be created in. The page also allows the user to add/remove items to/from the bookmark.

[2221] Wizard Page 2: Name and describe this bookmark: This page allows the user to enter a name and description for the bookmark. The wizard automatically suggests a name and description for the bookmark based on the items in the bookmark. Examples include:

[2222] Document 1, Document 2, and Document 3
[2223] Documents Matching ‘Encryption’
[2224] Documents in the Folder ‘My Documents’ and Subfolders

[2226] The user is allowed to override the suggested name/ description. The suggestions are truncated as needed based on a maximum name and description length. See FIG. 468.

[2227] 2. Scenarios

[2228] Show Me All Presentations on Protein Engineering

[2229] Using the Create Request Wizard, select the Presentations information-type (in Documents/Presentations), and then select the Protein Engineering category as a filter. Hit Next—the wizard intelligently suggests a name for the request (Presentations on Protein Engineering) based on the semantics of the request. The wizard also selects the right default predicates. Hit Finish. The wizard compiles the query, sends the SQML to the KISes in the selected profile, and then displays the results.
3. Intelligent Publishing—Tool Metadata Suggestion and Maintenance

While the Information Nervous System does not rely or depend on metadata that is stored by Publishing Tools (e.g., the author of a document), having such metadata available and reliable can be advantageous. One problem with prior art is that publishing tools (e.g., Microsoft Word™, Adobe Acrobat, etc.) do not intelligently manage the metadata creation and maintenance process. Here are some ways that the preferred embodiment of the present invention can be used to make the metadata creation and maintenance process better:

a. When the user creates a new document, add the author’s email address (this can be programmatically retrieved from the user’s email client and in the event that the user has several addresses, the publishing tool should prompt the user for which address to use) to the metadata header of the document (rather than merely the author’s name). This is because email addresses provide much more uniqueness (for instance, the name ‘John Smith’ could refer to one of millions of people—as such the existence of such data in the metadata of a document is not that useful). Note that one possible email address to use in the metadata header can be retrieved from, say, the logged on user’s single sign-on account (e.g., Microsoft Passport™)

b. When the document is edited and if the current user is different from the author of the document (as is indicated in the metadata header), prompt the user if he/she wants to change the metadata header accordingly. This provides some basis form of intelligent metadata maintenance.

This model can be applied across different object types and metadata fields in cases where the publishing tool can validate the field (e.g., as in the case of the currently logged on user’s name and email address).

P. Semantic Threads Specification for the Information Nervous System™

1. Semantic Threads

Overview

In the preferred embodiment, semantic threads are objects in the KIS semantic network that represent threads of annotations or conversations. They are different from regular email threads in that they are also semantic—they have object identifiers and type identifiers (the OBJECTTYPEID_THREAD identifier) thread-specific semantic links, they convey meaning via one or more ontology-based knowledge domains and they support dynamic linking. Also, because they are first-class objects in the Information Nervous System, they can be queried, copied, pasted, dragged, dropped, and used with the smart and object lenses. FIG. 53 illustrates a semantic thread object and its semantic links

Because a semantic thread object is a first-class member of the semantic network and the entire Information Nervous System, it is subject to manipulation, presentation, and querying like other objects in the system. For example, the semantic browser will allow the user to navigate from a Person object to all threads that that person has participated in (via the “Participant” predicate—with a predicate type id of PREDICATEYPEID_PARTICIPANTOFTHREAD). The user can then navigate from the thread to all the thread’s participants (People) and keep dynamically navigating from then on. To take another example, a thread object can also be a Best Bet in a given context (or none, if none is specified).

In the preferred embodiment, the semantic thread object also conveys meaning. This is advantageous because it means that the thread can be returned via a semantic query in the system. For instance, “Find me all threads on Topic A and Topic B.” The KIS maintains semantic links for semantic threads just like it does with other objects such as documents. However, because semantic threads can refer to multiple objects, the semantics of the thread evolve with the objects the thread contains. For example, a thread can start with one topic and quickly evolve to include other topics. Email threads can end in a very different “semantic domain” from where they started—participants introduce new perspectives, new information is added to the thread, email attachments might be added to the thread, etc., all on the basis of meaning.

The KIS manages the “semantic evolution” of semantic threads. It does this by adding semantic links to the thread to “track” the contents of the thread. For instance, if a thread starts off with one document and an annotation, the KIS adds a semantic link to the thread for each to which the category the document and annotation belong. In other words, the thread is asserted to have the same semantics as the document and annotation it contains. If another annotation is added to the thread (e.g., if a user annotates the first annotation), the KIS computes a new link strength for the categories of the new annotation that are already linked off the thread. It is preferable if it does this because the new annotation can attenuate or strengthen the semantics of the entire thread from a particular perspective. However, this modification of the strength of the semantic link(s) for the categories that are already present off the thread are preferably done on a per-category basis—as with other objects, the thread can belong to multiple categories with different strengths. The new link strength can be computed in at least two ways: in a simple embodiment, the average of all link strengths for the category being linked to the thread is used. However, this has the disadvantage that too many items in the thread of weak strength can erode the “perceived” (as far as the KIS semantic query processor is concerned) semantics of the entire thread. An alternative embodiment is to use the maximum link strength. However, this also has a disadvantage that the semantics of the thread might remain fixed to a domain/category even though the thread “has moved on” to new domains/categories. From a weighted-average perspective, this would likely return confusing results as the thread grows in size.

In the preferred embodiment, the KIS preferably computes a weighted average of all the link strengths for the categories to be linked to the thread. This new weighted average becomes the link strength. The weighted average is preferrably computed using the number of concepts in each object in the thread. This has the benefit of ensuring that “semantically light” objects (such as short postings) do not erode the semantics of the thread relative to “semantically denser” objects in the thread (such as email attachments and long postings). The number of concepts, and not the size, is preferrably used in the preferred embodiment because the size of the object is a less reliable indicator of the conceptual weight of the object. For instance, a document could contain images or could include much information that does not map well to key phrases or concepts.

Preferably, the computed weight could also include the time when the entry was added (thereby “aging” the
semantics of older items relative to newer ones). This weight is then multiplied by the category link strength and the multiples are added and then divided by the number of entries. Other weighting schemes can also be applied.

[2242] The following rules are applied when a new item is added to the semantic network and which is to be added to a semantic thread:

[2243] 1. Categorize the new item to be added to the thread

[2244] 2. For each category in the returned list of categories which are already on the semantic thread

```plaintext
{ Compute new weighted-average link strength
  Update category semantic link off the semantic thread
  object
}
```

[2245] 3. For each category in the returned list of categories which are not already on the semantic thread

```plaintext
{ Assign link strength
  Add category semantic link off the semantic thread object
}
```

[2246] The weighted-average link strength is computed as follows:

\[
\text{New Link Strength} = \frac{\sum \text{ Ci } \times \text{ Li } }{N} \tag{1}
\]

[2247] Where Ci is the normalized number of concepts (from 0 to 1) of object i, Li is the link strength of object i, and N is the number of objects in the thread (including the new object). The normalized number of concepts is computed by dividing the number of concepts in each object (extracted by the Knowledge Domain Manager (KDM)) by the number of concepts in the largest object in the thread (including the new object).

[2248] If a semantic thread comprises of standard, intrinsic (and unedited) email threads, the KIS modifies the semantic network differently. This is because most email clients include all prior email messages that form the thread in the most recent email message. As such, in this case, the KIS preferably simply uses the most recent email message as being representative of the entire thread. To accomplish this, the KIS preferably categorizes the most recent email message, and replace all prior semantic links (relating to categories) from the thread object with new semantic links corresponding with the new categories and link strengths.

[2249] For non-email threads (for example, threads that form based on an annotation of an existing object in the semantic network), the model described above should be employed. Alternatively, the KIS can maintain an Aggregate Thread Document (ATD) which is then categorized. This document should contain the text of the objects in the thread—roughly analogous to how an email message contains the text of prior messages in the same thread.

[2250] When a new object is added to the thread, the KIS preferably updates the last-modified-time of the thread object in the Semantic Metadata Store (SMS).

[2251] 2. Semantic Thread Conversations

[2252] Semantic thread conversations in the Information Nervous System are a special form of semantic threads. Essentially, a conversation is a semantic thread that has more than one participant. Semantic thread conversations have the object type id, OBJECTTYPEID_THREADCONVERSATION.

[2253] The KIS creates a thread based on the number of participants in that thread and could immediately create the thread as a thread conversation. Alternatively, the KIS could "upgrade" a thread to a conversation once additional participants are detected.


[2255] The pseudo-code below illustrates how the KIS adds preferred threads and conversations to the semantic network:

[2256] 1. If an individual email message is detected and is a member of an existing thread object

```plaintext
{ Add the new email object to the thread and update the semantic network
  If the thread has more than one participant, change the thread's object type identifier to OBJECTTYPEID_THREADCONVERSATION
}
```

[2257] 2. If an email thread is detected

```plaintext
{ Create a new thread object and update the semantic network
  If the thread has more than one participant, change the thread's object type identifier to OBJECTTYPEID_THREADCONVERSATION
}
```

[2258] 3. If an email annotation of an existing object is detected

```plaintext
{ Add the annotation to the semantic network
  If the annotated object is not itself an annotation
  { Create a new thread object and update the semantic network
  }
  Else
  { Add the new annotation to the thread containing the annotated object (i.e., the existing annotation) and update the semantic network
    If the updated thread has more than one participant, change the thread's object type identifier to OBJECTTYPEID_THREADCONVERSATION
  }
}
```

Q. Sample Screen Shots

[2259] FIG. 54 is a screen shot further illustrating the functions, options and operations discussed above.
1. Semantic Images & Motion

a. Overview

Semantic images and motion can be an advantageous component of the preferred embodiment in terms of the Nervana semantic user experience. In other words, the user's experience with the system can be enhanced in an embodiment that has semantic image/motion metadata stored on a Nervana agency (information community) and accessed via the Nervana XML web service. In that embodiment, via Nervana, end users will have context and time-sensitive semantic access to their images. Imagine, for example only, using a Getty Image™ (or Corbis™) agent as a smart lens over an email message—when invoked, this will open images that are semantically related to the message. Or, imagine dragging and dropping a document from your hard drive to a Getty agent to view semantically related images. This will involve having image metadata (consistent with an image schema). The Nervana toolbox remains the same—we merely add a new information object type for images. Also, there are semantic skins for semantic images—different views, thumbnails, slide shows, filtering, aggregation, etc. For examples of semantic images, visit:

[bibliographic reference]

Very generally, the properties of the semantic visualizations will vary depending upon different variables. Among these variables will often be the context, including the context of what feature or property of the system is being invoked. In the next several sections some of the contextual variables that influence the semantic determinations will be listed and/or described. In many instances, there will be overlap or commonality of the variables or determinants of the semantic visualizations, but in some cases, the considerations or combination of considerations will be unique to the particular situation.

b. Industry-Specific Semantic Images and Motion

Industry-specific semantic images/motion are images/motion that can be used (and in the preferred embodiment are used) as part of the presentation atmosphere for semantic results for one or more categories (that map to industries). For instance, visit [bibliographic reference] for the keywords listed below (which, in the aggregate, map to target industries, based on industry-standard taxonomies). Such images/motion can also be used as backgrounds, filter effects, transformations, and animations for context and category skins (that map to context templates and categories). In addition, these images/motion can be used for visuals for motion paths extracted from some of these images for superior screensavers. For example, imagine a skin displaying metadata and visualizations along a motion path extracted from one of these semantic images (e.g., metadata rotating inside a light bulb—for the "electric utilities" industry), along with chrome with other surrounding images and animations, etc. Other industries, with industry specific images and motion might include:

- Pharmaceuticals
- Telecommunications
- Airlines
- Medicine
- Telecom Equipment
- Retail
- Healthcare
- Telecom Services
- Fashion
- Life Sciences
- Telecom Technology
- Advertising
- Biotechnology
- Telecom Regulations
- Aerospace
- Oil and gas
- Tobacco
- Defense
- Chemical
- Automotive
- Agriculture
- Energy
- Automobiles
- Agriculture
- Electric Utilities
- Insurance
- Beverages
- Gas Utilities
- Consulting
- Business services
- Water Utilities
- Information
- E-commerce
- Entertainment
- Technology
- Environmental
- Services
- Computer Equipment
- Forest products
- Publishing
- Health Care Providers
- Real Estate
- Computing
- Hospitality
- Financial
- Semiconductors
- Internet
- Brokers
- Nanotechnology
- Law
- Financial Services
- Public Sector
- Legal
- Banking
- Government
- Manufacturing
- Consumer
- Homeland Security
- Marketing
- Consumer Products
- Travel
- Media
- Consumer Services
- Tourism
- Networking
- Communications
- Transportation

For example, if the user launches a request/agent, headlines on Bioinformatics or on Protein Engineering, the semantic browser will map the biotechnology-related categories from the SQML to a set of images in the biotechnology industry. It will then display one or more images as part of the skin for the results of the request/agent (thereby providing a pleasant user experience as well as visually conveying the "mood" of the request/agent).

FIG. 55-1 is a sample semantic image for Pharmaceutical/Biotech industry (artistic DNA helix superimposed over a human face on the left and an organic chemical chart on the right, licensed from the Corbis™ web site).

The same applies to information types and context templates. (See, for example, FIGS. 55-59) Skins will do the smart thing based on the context/information type and the category/ontology and mix and match semantic images/motion across these properties in an intelligent manner. For instance, an agent titled "Headlines on Wireless Technology" can have chrome (and/or a smart hourglass—see below) that shows an image/motion-based animation toggling between a "Headlines" image/motion and a "Wireless" image/motion.

A blender titled "Headlines on Wireless and Breaking News on Semiconductors and Email by anyone in my group related to the product specification" can have chrome (and/or a smart hourglass) that "toggles" between images/motion for "Headlines," "News," "Wireless," "Semiconductors," and "Email."

The Presenter's query processor can enumerate all context template and information types and all categories (from the agent/blender SQML) and set up the chrome animation accordingly.

For information types, enter searches (e.g., on Corbis™ and Getty) for:
Also, for context templates, enter searches for:

<table>
<thead>
<tr>
<th>Headlines</th>
<th>Favorites</th>
<th>News</th>
<th>Places</th>
<th>Discovery</th>
<th>Time (for “timeline” and “upcoming events”)</th>
<th>Conversations</th>
<th>Schedule</th>
<th>Experts</th>
<th>Appointment</th>
</tr>
</thead>
</table>

Also, note that semantic images/motion are preferably not completely random. However, preferably they are not from a bounded set either. Preferably, they are carefully picked and then skins can randomly select from the chosen set. But, preferably they are not random from the entire set, for example, Corbis™ or Getty Images™. Otherwise there may be similar images, cartoons, and some potentially offensive or inappropriate images. Also, some of these guidelines preferably vary depending on whether the skin theme is in subtle, moderate, exciting, or super-exciting mode. In subtle mode, the skin might decide to choose one image/motion per visualization pivot. In other modes, this would likely lead to a boring user experience.

In low-flashiness mode, the skin can use a semantic image/motion as part of the theme—not unlike a PowerPoint slide-deck background (e.g., alpha blended). Semantic images/motion can also be used in the smart hourglass (see below) as well as in part of the visualization (on the context bar, panel, or palette). For visualizing context and information types, semantic images/motion are preferably carefully picked to clearly indicate the information type or context. In addition, the selection mode can also be a skin property.

Also, the number of possible semantic images/motion used per skin would likely need to be capped—depending on where the images/motion are being displayed. However, in some scenarios, this might not be necessary. For instance, a blender skin might cycle between chrome backgrounds as the user navigates the blender results (from page to page or agent to agent)—to be consistent with what is currently being displayed from the blender. This can also be a skin property.

c. The Client-Side Semantic Image & Motion Cache

The Presenter has a smart expandable client-side cache with semantic images and motions that are downloaded and stored on the client (on installation). Skins can then select from these pre-cached images and motions. The images/motions can be pre-cached based on the user’s favorite categories and areas of interest (which he or she selects)—which map to target industries. Skins can then complement the pre-cached semantic images/motions with on-demand image queries to an image server (an XML Web Service that exposes server-side images/motions hosted by Nervana or a third party like Corbis™ or Getty Images™).

The Presenter will also do the smart thing and have a bias function such that recently downloaded images/motions are selected before older ones (as a tiebreaker). A “usage count” is also cached along with each image/motion—the Presenter uses this count in filtering which images/motions to display and when. Such “load balancing” will yield a fresher and non-repetitive user experience.

The cache is preferably populated on demand (based on the user’s semantic queries)—for instance, there is no point in pre-caching pharmaceutical images/motions for a user’s machine at Boeing. Preferably, the cache size is also capped and the image cache manager preferably purges “old” and “unused” images using an LRU algorithm or the equivalent. This way, the cache can be in “semantic sync” with the user’s agent usage pattern and favorite agent’s list.

2. The Smart Hourglass

A majority of the calls that the Nervana Presenter will make to provide the “semantic user experience” probably will be remote calls to the XML Web Service. As such, there will be unpredictable, potentially unbounded delays in the UI. One can expect a fair amount of bandwidth and server horsepower within the enterprise but the Nervana user interface must still “plan” for unknown latency in method invocations.

Operating systems today have this problem with unbounded I/O calls to disk or to the network. Some CPU-bound operations also have substantial delays. In the Windows™ and Mac™ UI, the user is made to perceive delay via a “wait” cursor—usually in the shape of an “hourglass.”

In the preferred embodiment, the Presenter will have semantic hints (via direct access to the SQML “method call”) with which it can display the equivalent of a “smart or semantic hourglass.” This could be in the form of an intermediate page that displays “Loading” or some other effect. Additionally, the Presenter can convey the semantics of the query by reading the SQML to get hints on the categories that the query represents and the information type or context template. The Presenter can then use these hints to display semantic images and text consistent with the query, even though it has not received the results. The more hints the query has, the smarter the hourglass can get. The “Loading” page can then convey the atmosphere of “what is to come”—even before the actual results arrive from the Web service and are merged (if necessary) by the Presenter to yield the final results.

This “smart hourglass” can be displayed not just on the main results pane, but perhaps also on smart lens balloon popup windows and inline preview windows (essentially at every call site to the Web service and where there is “focus”). The Presenter can do the smart thing by timing out on the query (perhaps after several hundred milliseconds—the implementation should use usability tests to arrive at a figure for this) before displaying the “hourglass.”

3. Visualizations—Context Templates

Introduction

Context templates are scenario driven information query templates that map to specific semantic models for information access and retrieval. Essentially, context templates can be thought of as personal, digital semantic information retrieval “channels” that deliver information to a user by employing a predefined semantic template. Context templates preferably aggregate information across one or more Agencies.

The context templates described below have been defined. Additional context templates, directed towards the integration and dissemination of varied types of semantic information, are contemplated (examples include context templates related to emotion, e.g., “Angry,” “Sad,” etc. context templates for location, mobility, ambient conditions, users tasks, etc.).

Breaking News

The Breaking News context template can be analogized to a personal, digital version of CNN’s “Breaking
News” program insert in how it conveys semantic information. The context template allows a user to access information that is extremely time-critical from one or more Agencies, sorted according to the information creation or publishing time and a configurable amount of time that defines information criticality.

**FIG. 55-2** is an illustration of a semantically appropriate image visualization for the Breaking News context template.

Breaking News—Sample Object and Context Bar Visualizations

Below is a list of sample or representative elements of visualizations appropriate to the Breaking News context. As with all Visualizations (or components thereof) in the preferred embodiment, the “mood” or semantic feeling or connotation will be appropriate to the specific context. By way of very rough analogy, the Visualization will be appropriate to the context within the application in the same way that a “set” must be appropriate to the particular scene in a screenplay for a movie. This will be true not only for this particular Object and Context Bar Visualization, but for all Visualizations in the preferred embodiment.

1. Ticking clock showing publication or scheduled time of most recent or pending breaking news item over a background of the total number of upcoming breaking news items
2. Ticking clock showing publication or scheduled time of most recent or pending breaking news item over semantic image(s)
3. Ticking clock showing publication or scheduled time of most recent or pending breaking news item over semantic image(s) and the total number of breaking news items
4. Ticking clock showing publication or scheduled time of most recent or pending breaking news item over a plain background
5. Non-ticking clocks showing publication or scheduled time of all breaking news items (sequentially) over various backgrounds
6. Calendar view showing publication or scheduled time of most recent or pending breaking news item over various backgrounds
7. Calendar view showing publication or scheduled time of all breaking news items (sequentially) over various backgrounds
8. Scaled font size—depending on the publication or scheduled time of the most recent or pending breaking news item
9. Scaled font size—depending on the number of breaking news items
10. Animated font (e.g., flashing text, rotating text, text on motion path, etc.) with animation rate depending on the publication or scheduled time of the most recent or pending breaking news item
11. Animated font (e.g., flashing text, rotating text, text on motion path, etc.) with animation rate depending on the number of breaking news items
12. Varying font color—depending on the publication or scheduled time of the most recent or pending breaking news item
13. Varying font color—depending on the number of breaking news items
14. Animated graphic of breaking news semantic image(s) or an equivalent
15. Number of breaking news items
16. Titles of breaking news items animated in a sequence (list view)
17. Titles and details of breaking news items animated in a sequence (tiled view)
18. Semantic image/motion moving on an orbital motion path around the object
19. Balloon popup showing number of items on semantic image/motion background
20. Balloon popup showing number of items with plain background but animated with semantic image/motion

### Headlines

The Headlines context template can be analogized to a personal, digital version of CNN’s “Headline News” program in how it conveys semantic information. The context template allows a user to access information headlines from one or more Agencies, sorted according to the information creation or publishing time and a configurable amount of time or number of items that defines information “freshness.” For example, CNN’s “Headline News” displays headlines every 30 minutes (around the clock). In a preferred embodiment, the Headlines context template will be implemented as a SQL query on the server with the following sub queries chained in sequence: Recommendations Published Today, Favorites Published Today, Best Bets Published Today, Upcoming Events Occurring Today and Tomorrow, Annotated Items Published Today.

Preferably, all sub queries will be sorted by the publishing date/time and then be chained together. Additional filters will be applied to the query based on the predicate list in the SQML. The foregoing principles are illustrated in FIG. 55-3, which is a Headlines Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background, chrome, etc.

### Conversations Context Template

The Conversations context template can be analogized to a personal, digital version of CNN’s “Crossfire” program in how it conveys semantic information. Like “Crossfire,” which uses Conversations and debates as the context for information dissemination, in the preferred embodiment, the Conversations context template tracks email postings, annotations, and threads for relevant information.

The Conversations context template comprises the following information object types:

1. Email of a thread depth of at least one (An email reply to an email message)
2. Annotations of a thread depth of at least one (The annotation of an annotation of an object)
3. Internet News Postings (A news posting reply to a news posting)

The query will be sorted by thread depth. Additional filters will be applied to the query based on the predicate list in the SQML. In addition, the context skin should display the information items by thread.
FIG. 55-4 is a Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Two People working at a desk)

Conversations Context—Sample Object and Context Bar Visualizations

Below is a list of considerations for, or characteristics of visualization elements semantically appropriate to the corresponding indicated context (in parentheses).

1. Animated graphic of semantic image/motion(s) (icon and context guide view)
2. Maximum thread depth over plain background (icon and context guide view)
3. Maximum thread depth over semantic image/motion (icon and context guide view)
4. Titles of conversations animated in a sequence (list view)
5. Titles and details of conversations animated in a sequence (tiled view)
6. The number of conversations over a plain background (icon and context guide view)
7. The number of conversations over semantic image/motion(s) (icon and context guide view)

Newsmakers Context Template

The Newsmakers context template can be analogized to a personal, digital version of NBC’s “Meet the Press” program in how it conveys semantic information. In this case, the emphasis is on “people in the news,” as opposed to the news itself or Conversations. Users navigate the network using the returned people as Information Object Pivots. The Newsmakers context template can be thought of as the Headlines context template, preferably with the “People” or “Users” object type filters, and the “authored by,” “possibly authored by,” “hosted by,” “annotated by,” “expert on,” etc. predicates (predicates that relate people to information). The “relevant to” default predicate preferably is used to cover all the germane specific predicates. The sort order of the relevant information, e.g., the newsmakers, is sorted based on the order of the “news they make,” e.g. “headlines”.

The query will be sorted by number of headlines. Additional filters will be applied to the query based on the predicate list in the SQML.

FIG. 55-5 illustrates a semantic “Newsmaker” Visualization or Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Football Championship)

Newsmakers—Sample Object and Context Bar Visualizations

1. Animated graphic of 2 talking heads in conversation (icon and context guide view)
2. Animated graphic of semantic image/motion(s) (icon and context guide view)
3. Total number of newsmakers (icon and context guide view)
4. Total number of newsmakers over semantic image/motion (icon and context guide view)
5. Names of newsmakers animated in a sequence (list view)
6. Names and details of newsmakers animated in a sequence (tiled view)

Upcoming Events Context Template

The Upcoming Events context template (and its resulting Special Agent) can be analogized to a personal digital version of special programs that convey information about upcoming events. Examples include specials for events such as “The World Series,” “The NBA Finals,” “The Soccer World Cup Finals,” etc. The equivalent in a knowledge-worker scenario is a user that wants to monitor all upcoming industry events that relate to one or more categories, documents or other Information Object Pivots. The Upcoming Events context template is preferably identical to the Headlines context template except that only upcoming events are filtered and displayed (preferably using a semantically appropriate “context Skin” that connotes events and time criticality). Returned objects are preferably sorted based on time criticality with the most impending events listed first.

FIG. 55-6 illustrates a semantic “Upcoming Events” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Appointment Binder)

Upcoming Events—Sample Object and Context Bar Visualizations

1. Ticking clock showing time till next event over a background of the total number of upcoming events (icon and context guide view)
2. Ticking clock showing time till next event over semantic image/motion(s) (icon and context guide view)
3. Ticking clock showing time till next event over semantic image/motion(s) and the total number of upcoming events (icon and context guide view)
4. Ticking clock showing time till next event over a plain background (icon and context guide view)
5. Non-ticking clocks showing time till all upcoming events (sequentially) over various backgrounds (icon and context guide view)
6. Calendar view showing scheduled time of next upcoming event over various backgrounds (icon and context guide view)
7. Calendar view showing scheduled time of all upcoming events (sequentially) over various backgrounds (icon and context guide view)
8. Animated graphic showing calendar motion (icon and context guide view)
9. Animated graphic of semantic image/motion(s) (e.g., schedule book) (icon and context guide view)
10. The total number of upcoming events over semantic image/motion(s) (icon and context guide view)
11. The total number of upcoming events over a plain background (icon and context guide view)
12. Titles of upcoming events animated in a sequence (list view)
13. Titles and details of upcoming events animated in a sequence (tiled view)

Discovery

The Discovery context template can be analogized to a personal, digital version of the “Discovery Channel.” In this case, the emphasis is on “documentaries” about particular topics. The Discovery context template simulates intelligent aggregation of information by randomly selecting information objects that relate to a given set of categories and which are posted within an optionally predetermined, configurable time period. The semantic weight as opposed to the time is the preferred consideration for determining how the information is to be ordered or presented. The context template can be implemented by filtering all information types by the semantic link strength for the categorization predicate. In this case, the filter should be less selective than the “Best Bets” filter—
the context template lies somewhere between ‘Best Bets’ and ‘All Items’ in terms of filtering.

[2355] FIG. 55-7 is a “Discovery” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Petri Dish).

Discovery—Sample Object and Context Bar Visualizations

[2356] 1. Animated graphic of semantic image/motion(s) (e.g., a telescope, a voyager spacecraft, an old ship at sea) (icon and context guide view)
[2357] 2. Titles of the first N information items in a sequential animation (list view)
[2358] 3. Titles and details of the first N information items in a sequential animation (tile view)
[2359] 4. The total number of items over semantic image/motion(s) (icon and context guide view)
[2360] 5. The total number of items (icon and context guide view)

[2361] History

[2362] The History context template can be analogized to a personal, digital version of the “History Channel.” In this case, the emphasis is on disseminating information not just about particular topics, but also with a historical context. For this template, the preferred axes are category and time. The History context template is similar to the Discovery context template, further in concert with “a minimum age limit.” The parameters are preferably the same as that of the Discovery context template, except that the “maximum age limit” parameter is replaced with “a minimum age limit” parameter (or an optional “history time span” parameter). In addition, returned objects are preferably sorted in reverse or random order based on their age in the system or their age since creation.

[2363] FIG. 55-8 illustrates a semantic “History” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (War Memorial).

History—Sample Object and Context Bar Animations Visualizations

[2364] 1. Animated graphic of semantic image/motion(s) or an equivalent
[2365] 2. Titles of the oldest (or random) N information items in a sequential animation (list view)
[2366] 3. Titles and details of the oldest (or random) N information items in a sequential animation (tile view)
[2367] 4. Total number of items over semantic image/motion(s) (icon and context guide view)
[2368] 5. Total number of items over plain background (icon and context guide view)

[2369] All Items

[2370] The All Items context template represents context that returns any information that is relevant based on either semantics or based on a keyword or text based search. In this case, the emphasis is on disseminating information that may be even remotely relevant to the context. The primary axis for the All Items context template is preferably the more possibility of relevance. In the preferred embodiment, the All Items context template employs both a semantic and text-based query in order to return the broadest possible set or universe of results that may be relevant.

[2371] FIG. 56-1 illustrates a semantic Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Outer Space).

All Items—Visualization & Sample Object and Context Bar Animations

[2372] 1. Animated graphic of semantic image/motion(s) or an equivalent
[2373] 2. Titles of the most recent N information items in a sequential animation (list view)
[2374] 3. Titles and details of the most recent N information items in a sequential animation (tile view)
[2375] 4. Total number of items over semantic image/motion(s) (icon and context guide view)
[2376] 5. Total number of items over plain background (icon and context guide view)

[2377] Best Bets

[2378] The Best Bets context template (and its resulting Special Agent) represents context that returns only highly relevant information. In a preferred embodiment, the emphasis is on disseminating information that is deemed to be highly relevant and semantically significant. For this context template, the primary axis is relevance. In essence, the Best Bets context template employs a semantic query and will not use text based queries since it cannot guarantee the relevance of text-based query results. The Best Bets context template is preferably initialized with a category filter or keywords. If keywords are specified, the server performs categorization dynamically. Results are preferably sorted based on the relevance score, or the strength of the “belongs to category” semantic link from the object to the category filter.

[2379] FIG. 56-2 illustrates a “Best Bets” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Microscope).

Best Bet Visualization—Sample Object and Context Bar Animations

[2380] 1. Animated graphic of semantic image/motion(s) or an equivalent
[2381] 2. Titles of the most recent N information items in a sequential animation (list view)
[2382] 3. Titles and details of the most recent N information items in a sequential animation (tile view)
[2383] 4. Total number of items over semantic image/motion(s) (icon and context guide view)
[2384] 5. Total number of items over plain background (icon and context guide view)

Favorites

[2385] The Favorites context template (and its resulting Special Agent) represents context that returns “favorite” or “popular” information. In this case, the emphasis is on disseminating information that has been endorsed by others and has been favorably accepted. In the preferred embodiment, the axes for the Favorites context template include the level of readership interest, the “reviews” the object received, and the depth of the annotation thread on the object. In one embodiment, the Favorites context template returns only information that has the “favorites” semantic link, and is sorted by counting the number of “votes” for the object (based on this semantic link).
FIG. 56-3 illustrates a semantic Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc. (coffee and pastry).

Favorites Visualization—Sample Object and Context Bar Animations

1. Animated graphic of semantic image/motion(s) or an equivalent
2. Titles of the most recent N information items in a sequential animation (list view)
3. Titles and details of the most recent N information items in a sequential animation (tiled view)
4. Total number of items over semantic image/motion(s) (icon and context guide view)
5. Total number of items over plain background (icon and context guide view)

Classics

The Classics context template (and its resulting Special Agent) represents context that returns “classical” information, or information that is of recognized value. Like the Favorites context template, the emphasis is on disseminating information that has been endorsed by others and has been favorably accepted. For this context template, the preferred axes include a historical context, the level of readership interest, the “reviews” the object received and the depth of the annotation thread on the object. The Classics context template is preferably implemented based on the Favorites context template but with an additional minimum age limit filter and voting score, essentially functioning as an “Old Favorites” context template.

FIG. 56-5 illustrates a semantically appropriate Sample Image for “Classics” for smart hourglass, interstitial page, transition effects, background chrome, etc. (Car)

Classics Visualizations—Sample Object and Context Bar Animations

1. Animated graphic of semantic image/motion(s) or an equivalent
2. Titles of the most recent N information items in a sequential animation (list view)
3. Titles and details of the most recent N information items in a sequential animation (tiled view)
4. Total number of items over semantic image/motion(s) (icon and context guide view)
5. Total number of items over plain background (icon and context guide view)

Recommendations

The Recommendations context template represents context that returns “recommended” information, or information that the Agencies have inferred would be of interest to a user. Recommendations will be inserted by adding “recommendation” semantic links to the “SemanticLinks” table and by mining the favorite semantic links that users indicate. Recommendations are preferably made using techniques such as machine learning and collaborative filtering. The emphasis of this context template is on disseminating information that would likely be of interest to the user but which the user might not have already seen. For this context template, the primary axes preferably include the likelihood of interest and freshness.

FIG. 56-5 illustrates a semantically appropriate “Recommendation” Visualization—Sample Image for the contextual/application elements of smart hourglass, interstitial page, transition effects, background chrome, etc. (Thumbs up).

Recommendation Visualization—Sample Object and Context Bar Animations

1. Animated graphic of semantic image/motion(s) or an equivalent
2. Titles of the most recent N information items in a sequential animation (list view)
3. Titles and details of the most recent N information items in a sequential animation (tiled view)
4. Total number of items over semantic image/motion(s) (icon and context guide view)
5. Total number of items over plain background (icon and context guide view)

Today

The Today context template represents context that returns information posted or holding (in the case of events) “today.” The emphasis with this context template is preferably on disseminating information that is deemed to be current based on “today” being the filter to determine freshness.

FIG. 56-6 illustrates a semantic “Today” Visualization—Sample Image for the elements smart hourglass, interstitial page, transition effects, background chrome, etc.

“Today Visualization”—Sample Object and Context Bar Animations

1. Animated graphic of semantic image/motion(s) or an equivalent
2. Titles of the most recent N information items in a sequential animation (list view)
3. Titles and details of the most recent N information items in a sequential animation (tiled view)
4. Total number of items over semantic image/motion(s) (icon and context guide view)
5. Total number of items over plain background (icon and context guide view)

Annotated Items

The Annotated Items context template represents context that returns annotated information. The emphasis with this context template is on disseminating information that is likely to be important based on the fact that one or more users have annotated the items.

FIG. 56-7 illustrates a semantic “Annotated Items” Visualization—Sample Image for smart hourglass, interstitial page, transition effects, background chrome, etc.

“Annotated Items” Visualization—Sample Object and Context Bar Animations

1. Animated graphic of semantic image/motion(s) or an equivalent
2. Titles of the most recent N information items in a sequential animation (list view)
3. Titles and details of the most recent N information items in a sequential animation (tiled view)
4. Total number of items over semantic image/motion(s) (icon and context guide view)
5. Total number of items over plain background (icon and context guide view)
Annotations

[2420] The Annotations context template represents context that returns annotated information. The emphasis with this context template is on disseminating information that are annotations.

[2421] FIG. 56-8 illustrates a semantic “Visualization—Sample image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Note pinned to Bulletin Board)

“Annotation” Visualization—Sample Object and Context Bar Animations

[2422] 1. Animated graphic of semantic image/motion(s) or an equivalent
[2423] 2. Titles of the most recent N information items in a sequential animation (list view)
[2424] 3. Titles and details of the most recent N information items in a sequential animation (list view)
[2425] 4. Total number of items over semantic image/motion(s) (icon and context guide view)
[2426] 5. Total number of items over plain background (icon and context guide view)

Experts

[2427] FIG. 57-1 illustrates a semantic “Experts” Visualization—Sample image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Professor)

“Experts” Visualization—Sample Object and Context Bar Animations

[2428] 1. Animated graphic of semantic image/motion(s) or an equivalent
[2429] 2. Names of the most recent N experts in a sequential animation (list view)
[2430] 3. Names and details of the most recent N experts in a sequential animation (list view)
[2431] 4. Total number of experts over semantic image/motion(s) (icon and context guide view)
[2432] 5. Total number of experts over plain background (icon and context guide view)

Places

[2433] FIG. 57-2 illustrates a semantic “Places” Visualization—Sample image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Paris)

“Places” Visualization—Sample Object and Context Bar Animations

[2434] 1. Animated graphic of semantic image/motion(s) or an equivalent
[2435] 2. Names of the most recent N places in a sequential animation (list view)
[2436] 3. Names and details of the most recent N places in a sequential animation (list view)
[2437] 4. Total number of places over semantic image/motion(s) (icon and context guide view)
[2438] 5. Total number of places over plain background (icon and context guide view)

Blenders

[2439] FIG. 57-3 illustrates a semantic “Blenders” Visualization—Sample image for smart hourglass, interstitial page, transition effects, background chrome, etc. (Blenders)

“Blenders” Visualization—Sample Iconic Animations

[2440] 1. Animated graphic of semantic image/motion(s) or an equivalent
[2441] 2. Animated graphic of blender or mixer in action
[2442] 3. Titles of the blender items in a sequential animation (list view)
[2443] 4. Titles and details of the blender items in a sequential animation (list view)
[2444] 5. Total number of items over semantic image/motion(s) (icon and context guide view)
[2445] 6. Total number of items over plain background (icon and context guide view)

Information Object Types

[2446] FIGS. 57-3 through 59-6 illustrate semantic Visualizations for the following Information Object Types, respectively: Documents, Books, Magazines, Presentations, Resumes, Spreadsheets, Text, Web pages, White Papers, Email, Email Annotations, Email Distribution Lists, Events, Meetings, Multimedia, Online Courses, People, Customers, and Users.

Presentation Skin Types

Timeline

[2447] FIG. 59-7 illustrates a semantic “Timeline” Visualization—Sample image for smart hourglass, interstitial page, transition effects, background chrome, etc.

“Timeline” Visualization—Sample Object and Context Bar Animations

[2448] 1. Calendar view showing effective time (publication time, scheduled time, etc.) of information item over various backgrounds (icon and context guide view)
[2449] 2. Calendar view showing effective time of all information items (sequentially) over various backgrounds (icon and context guide view)
[2450] 3. Animated graphic showing calendar motion (icon and context guide view)
[2451] 4. Animated graphic of semantic image/motion(s) (e.g., time warp image/motion) (icon and context guide view)
[2452] 5. The total number of information items over semantic image/motion(s) (icon and context guide view)
[2453] 6. The total number of information items over a plain background (icon and context guide view)
[2454] 7. Titles of information items animated in a sequence (list view)
[2455] 8. Titles and details of information items animated in a sequence (tilled view)
[2456] 9. Scrolling, linear timeline control with items populated based on effective date/time
[2457] 10. Animated timeline ticker control sorted by effective date/time
[2459] One final note concerning Visualizations. The preferred embodiment not only searches for information seman-
tically, and not only organizes and stores it semantically, it also presents it semantically. And, the presentation is not semantic only in the sequence, organization and relationships of the information, but also visually, as the foregoing Visualizations are, in part, intended to convey. As a result, the user is aided in understanding the information being presented by the system in roughly the same way that a viewer of a movie is aided in understanding the meaning of dialogue by the surrounding context of the lighting, costume, music and entire set or scene. Put differently, the Visualizations, as with everything else presented or managed by, or located with, the preferred embodiment system, serve the purpose of conveying meaningful information; or, just as aptly, to convey information meaningfully. Meaning is a unifying theme of the preferred embodiment; it permeates the design and operation of the system, and each constituent component part of which the system is comprised.

[2460] There will be debates, questions, etc. amongst users of the Information Nervous System on the appropriate queries to ask given the intent of the users. There might be a tendency to assume that this is a "problem," and that the user should immediately be able to determine the right query given his/her intent. This is not necessarily a problem, but on the contrary can be an advantageous reflection of a natural and/or "Darwinian" process of context selection.

Intent and context are "curvy" and could have an arbitrary number of "geometric forms." Indeed, it is good to see healthy debates and conversations on what the "right query" is, for a given user's intent. Part of this has to do with users having to become more familiar with the system. However, there will always be competing representations of semantic intent. This is natural and healthy.

[2461] In a previously-filed commonly owned application, there was described what were called "entities." Entities can include digital representations of abstract, personalized context. There may be competing entities within a community of knowledge. In one embodiment, users create and share entities INDEPENDENT of knowledge sources. In one scenario, an Entity Market could develop where domain experts could get brokering rights for creating and sharing the best entities in a given context. Human librarians could focus on creating and sharing the best entities for their organizations, based on their knowledge of ongoing projects and researchers' intent. Entities could even be shared across organizational boundaries by independent domain experts. In one embodiment, users can be able to save and email entities to each other. The best entities will win. Again, this is natural.

[2462] In one embodiment, a user can be able to open an entity (sent, say, via email) in the Librarian and then drag and drop that entity to a Knowledge Community like Medline. Again, the entity is INDEPENDENT of the knowledge source. The entity could be applied to ANY knowledge source in ANY profile. With entities, context (and NOT content) is important.

In one embodiment, example of entities that would map to recent "debates on context" are:

[2463] 1. HIV Infection (CRISP) and Immunologic assay and Test (CRISP)

[2464] 2. Plasmodium Falciparum (MeSH) AND Polymerase Chain Reaction (MeSH) AND ("diagnosis of malaria" OR "malaria diagnosis")

Semantic stemming in the Knowledge Integration Service (KIS). In one embodiment, this allows the user to easily specify a qualified keyword that the KIS can interpret semantically. This can significantly aid usability, especially for those users that might not care to browse the ontologies, and for access from the simple Web UI. In one embodiment, the query, "Find all chemicals or chemical leads relevant to bone diseases and available for licensing" can now be specified simply as:

[2465] *chemical AND "*bone diseases" licensing

[2466] Or

[2467] *chemical AND "*bone diseases" AND licensing

[2468] The following rules may be used in various embodiments of the invention to achieve semantic stemming. Each of the rules may be practiced independently of the others or in combination with one or more rules. Furthermore, the rules themselves may be altered, reduced, or augmented with various steps as may be necessary.

[2469] 1. In one embodiment, the KIS preferably maps *: to ALL supported ontologies and intelligently generates a semantic query (alternatively, the user can specify an ontology name to restrict the semantic interpretation to a specific ontology e.g., "MeSH: bone diseases"). This implementation turned out to be non-trivial because the KIS smartly prunes the query in order to guarantee fast performance. In one embodiment, the following pruning rules may be employed.

[2470] A. Map the keyword to categories by calling the Ontology Lookup Manager (OLM). The OLM caches the ontologies that the KIS may be subscribed to (via KDSes). The ontologies may be zipped by the KDS and/or exposed via [HTTP] URLs. The KIS then auto-downloads the ontologies as KDSes may be added to KCS on the KIS. The KIS also periodically checks if the ontologies have been updated. If they have, the KIS re-caches the ontologies. When an ontology has been downloaded, it may be then indexed into a local Ontology Object Model (OOM). The data model may be described in detail in the section titled "Semantic Stemming Processor Data and Index Model" below. The indexing may be transacted. Before an ontology may be indexed, the KIS sets a flag and serializes it to disk. This flag indicates that the ontology may be being indexed. Once the indexing is complete, the flag may be reset to 0 (FALSE). If the KIS is stopped or goes down while the indexing is in progress, the KIS (on restart) can detect that the flag is set (TRUE). The KIS can then re-index the ontology. This ensures that an incompletely indexed ontology isn't left in the system. In one embodiment, indexed ontologies may be left in the KIS and aren't deleted even when KCS are deleted—for performance reasons (since ontology indexing could take a while).

[2471] B. If at least one ontology for a KC is still being indexed into the OOM and a semantic query comes in to the KIS (needing semantic stemming), the KIS uses the KDS for ontology lookup. In such a case, the fuzzy mapping steps below may be employed. Else, the KIS employs the OLM, which invokes a semantic query on the Ontology Table(s) referred to by the semantic query. This first semantic query may get the categories from the semantic keywords (semantic wildcards). If there are multiple ontologies, a batched query can be used to increase performance (across multiple ontology tables in the OOM).

[2472] C. The modified time of ontologies at the KDS may be the modified time of the ontology file itself and not of the ontology metadata file; this way, if only the ontology XML file may be updated, that would be enough to trigger a KIS ontology-cache update.
D. For all returned categories (which could include many irrelevant categories because of poor document set
analysis algorithms using context-less Latent Semantic
Indexing or similar techniques), prune the list by checking for
categories matching the qualified concept name (passed by
the user)—when fuzzy mapping with the KDS may be
employed
E. If there are still no categories, perform a fuzzy
string compare (e.g., bacterium □ bacteria)—when fuzzy
mapping with the KDS may be employed
F. If there are still no categories, add all the returned
categories just to be safe—perhaps only when fuzzy mapping
with the KDS may be employed
G. If there are still no categories, add a non-semantic
corresponding to the passed concept name. The KIS
defaults to a non-semantic filter if the specified filter cannot
be semantically interpreted. This allows the user to be lazy by
specifying the "*:*" with the assurance that Keywords may be
used as a last resort.
H. Add the pruned categories to a local cache for
super-fast lookup. The cache may be guarded by a reader-
writer lock since the cache may be a shared resource. This
ensures cache coherency without imposing a performance
penalty with multiple simultaneous queries.

1. The cache may be pruned after 10,000 entries
using FIFO logic.
2. In one embodiment, the stemmer intelligently
picks candidates on a per ontolgy basis—when fuzzy
mapping with the KDS may be employed. This way,
selecting one good candidate from one ontology does not
preclude the selection of other good candidates from
other ontologies—even with a direct (non-fuzzy) match
with one ontology. Example:

*chemical would map to chemical (CRISP) and/or Drugs
and Chemicals (Cancer). Ditto for *chemicals.
3. When fuzzy mapping is employed, in one
embodiment, more fuzzy logic can be added to map
terms in the semantic stemmer to close equivalents—e.g.,
*Calcium Channel-Calcium Channel Inhibitor Activity.
In one embodiment, this error on the conserva-
tive side (supersets may be favored more than subsets;
subsets may require the same number of terms to qualify
as candidates). In any event, even if the fuzzy logic
results in false positives, the model still handles this and
“balls itself out” (the fuzzy logic, not unlike the ontology
imperfections, may be a form of uncertainty). The event-
ual filters soften the impact of this uncertainty.
4. When fuzzy mapping is employed, add more
predicate logic to correctly interpret complex queries
that have field qualifiers. The KIS can infer the union of
predicates for complex queries that have a combination of
different qualifiers. This may be a semantic approxi-
mation in order to guarantee fast graph traversal. How-
ever, by restricting the predicate set to the union set (as
opposed to all predicates), this significantly increases
precision for these query types.
5. Example: Find all research on Heart or Bone
Diseases published by Merck or published in 2005:
Dossier on ("*Heart Diseases" OR "*:Bone Diseases") AND (affil:Merck OR pubYear:2005)
6. The KIS can add a default concept filter check
for ontology or cross-ontology qualified keywords (e.g.,
"*:bone diseases"). This addition may be only done for
rank bucket 0 and/or for All Bets or Random Bets—for
non-semantic sub-queries. This offers high precision
even with ontology-qualified keywords and/or for
semantic knowledge types like Best Bets or Breaking
News.
7. When fuzzy mapping is employed, added more
smarts to the KIS semantic stemmer. If the stemmer
doesn’t find initial candidates, it preferably carefully
prunes the large (and/or often false-positive laden—due
to context-less document analysis) category list from the
KDS. It does this by eliding parent paths for all paths—
ensuring that no included path also has an ancestor
included. This heuristic works very well, especially
since the KIS does its own semantic and/or context-
sensitive inference (meaning the stemmer doesn’t have
to try to be too clever). Example:
Find all recent press releases or product announce-
ments on infectious polynuereis:
Dossier on “*:infectious polynuereis” this preferably
returns results on polynuereis and on the Guillain-
Barre Syndrome, which is also known as infectious polynuereis.
8. The semantic stemmer preferably recognizes
ontology name aliases.
So you can preferably have Dossier on Go-Bio:
Apoptosis
Alias names for all our current ontologies are avail-
able. However, even if the alias name is not present, the KIS
tries to infer the ontology name by performing a direct or
fuzzy match. So Cancer:Kinase or NCI:Kinase would both
work and both map to Cancer (NCI).
9. The KIS semantic stemmer can dynamically
add a non-semantic concept filter for an ontology qualified
concept if the rank bucket is 0 or if the concept
could not be semantically interpreted. This is beautiful
because it works for all cases: if the concept could not be
interpreted, the non-semantic approximation may be used;
if the concept was interpreted and/or the context is
semantic (e.g., Best Bets or Breaking News), the non-
semantic concept may be not added as not to pollute
the results (since the concept has already been inter-
preted); if, on the other hand, the rank bucket is 0, the
semantics don’t matter so adding the concept is a good
thing anyway (it increases recall without imposing a cost
on precision), even if the concept has already been
semantically interpreted.

In one embodiment, a method to the KIS Web Ser-
vice Interface for the Web UI integration. The KIS may be
passed a text string (including Booleans) which it can then
to a semantic query.
In one embodiment, the KIS can automatically
specify the “since” parameter to the KIS Data Connector (if it
detects this) to optimize the incremental indexing path to
minimize the number of redundant queries during incremen-
tal indexing (since there are much more read-write conten-
tion—since it may be a real-time service).
In one embodiment, the KIS may use the system
thread-pool and/or EACH KC runtime object can have its
own semaphore. This ensures that the KCs don’t overwork
the KDSes yet increases concurrency by allowing multiple KCs
to index as fast as possible simultaneously.
In one embodiment, the central KIS runtime man-
ger holds/increments a work reference count on each docu-
ment sourced from each connector that may be currently
indexing (it releases/reductions it once it is done indexing
the document). This fixes a problem where a KC connector would quickly "find" an RSS file and think it was done, even while the items within the RSS file were still being processed and/or indexed.

[2496] In one embodiment, the KIS supports broad time-sensitivity settings.

[2497] a. Every two months

[2498] b. Every three months

[2499] In one embodiment, the KIS can map extended characters to English-variants. For instance, the Guillain–Barre Syndrome can be mapped to Guillain–Barre Syndrome.

In one embodiment, Semantic Wildcards may be also integrated with Deep Info. The user may be able to specify a request including (but not limited to) semantic wildcards and/or then navigate the virtual knowledge space using the request as context. The KIS returns category paths to the semantic client which can then be visualized in Deep Info (not unlike Category Discovery). The user may then be able to navigate the hierarchies and/or continue to navigate Deep Info from there. The following are examples of various embodiments of the invention. They may be practiced independently or in combination and/or may be limited or augmented with steps as may be necessary.

[2500] The categories may be visualized in the Deep Info console. And then the tree can be directly invoked by the user to launch a semantic query off a related category once the user discovers a category from his/her launch point (returned categories can be visualized differently from parent categories—perhaps in a different font/color). This could be a profile, keywords, document, entity, etc. In this case, it may be the request itself.

[2501] There may be a Request Deep Info, Profile Deep Info, and/or Application Deep Info—corresponding to different default launch points (in all cases, some Deep Info elements—like categories in the News, etc.—can always be available). In other cases, the user can type in keywords in the Deep Info pane to "semantically explore" the keywords without explicitly launching a request.

[2502] Another launch point may be the Clipboard—the Deep Info console can have a Clipboard Launch Point (if there is something on the clipboard) for whatever may be on the clipboard. This is very powerful as it would the user to copy anything to the clipboard (text, chemical images, document, etc.), go to the Deep Info and/or then browse/explore without actually launching a request.

[2503] Some Deep Info metadata (like categories) can be returned as part of the SRML header (they may be request-specific but result-independent).

[2504] The KIS can preferably handle virtually any kind of semantic query that users might want to throw at it (Drag and Drop and entities can provide even more power).

[2505] An exemplary search for example might be to find recent research by Pfizer or Novartis on the impact of cell surface receptors or enzyme inhibitors on heart or kidney diseases.

[2506] We can preferably handle this query as follows:

```
Dossier on (Pfizer or Novartis) AND (**:Cell Surface Receptors**
OR **:Enzyme Inhibitors**) AND (**:Heart Diseases**
OR **:Kidney Diseases**)
```

[2507] An example of the semantically stemmed and/or generated sub-queries is shown below.
[2508] Semantic Client highlights preferred ontology-qualified prefix tags

[2509] In one embodiment, Ontology qualified or multi-ontology qualified search terms and the Librarian can semantically highlight relevant terms. So for example, type in Dossier on ""bone disease"" and the semantic client can do the smart thing. This was non-trivial and has some pieces that need to be noted in the docs:

[2510] In one embodiment, ontology-qualified terms may be dynamically interpreted based on the current profile, the semantic client maps the terms (e.g., ""bone disease") to the ontologies for the request profile. It gets tricky shortly thereafter. For multi-ontology mapping (prefixed with ""~""), the semantic client figures out the ontologies for the request profile and/or add semantic highlight terms for each of these ontologies. However, going through multiple ontologies has an impact on performance. Furthermore, the user could (in the limit) have a profile with tens of KCs each of which have several different ontologies. As such, a more pragmatic, fuzzy algorithm was called for. The following are various embodiments of the invention that may be practiced independently or in combination and/or may be reduced or augmented or altered with steps as may be necessary.

[2511] a) The Librarian first starts a timer to time the mapping process. This may be configurable and/or can be switched off to have no timer.

[2512] b) The Librarian then tries all the ontologies in the request profile in the order of ontology size. This ensures that it filters through smaller ontologies.

[2513] c) If the ontology returns in less than a second, the timer (if available) may be reset. This ensures that many small ontologies don’t preclude the generation of terms from larger ontologies that await downstream in time.

[2514] d) Once the Librarian finds an ontology that has the semantic terms, it stops. This may be a good trade-off because the alternative may be to greedily check all ontologies for the terms. This isn’t practical and/or wouldn’t buy much because there may be a fair chance that the ontologies have good terms for the desired concept (if they have the concept at all). In other words, the likelihood is that an ontology either has good terms for a concept or doesn’t support the concept, period.

[2515] e) The Librarian continues to hunt for semantic terms with the remaining ontologies until the timer expires. Currently, there may be a timeout of 10 seconds.

[2516] f) The mapping process using XPath to find every descendant of every category that has a hook corresponding to the desired concept. This entailed loading the XML document, finding all the hooks with the concept name, cloning the iterator, navigating to the parent category, and/or then selecting all the descendants of the parent category.

[2517] g) When the Presenter attempts to ask for the highlight hit list, the semantic runtime client preferably waits for the hit generation for 10 seconds (if configured to have a timer). This may be enough time for most queries but also prevents the system from locking up in case the user has a query with, say, 20, cross-ontology qualifiers (this could hang the system).

[2518] h) This algorithm may be stable and/or provides the user with a very high probability of always getting most or all the right terms (with "~") or all the right terms with specific categories or keywords, WITHOUT making the system vulnerable to hangs with, say, arbitrary queries with a profile with many arbitrary KCs.

[2519] Support Parenthesized Filters on Categories

[2520] In one embodiment, the entire system (end-to-end) supports parenthesized category filters.

[2521] Semantic client correctly highlights hooks included in "NOT" predicates

[2522] In one embodiment, Dossier on Autoimmune Diseases AND NOT on Multiple Sclerosis excludes Multiple Sclerosis terms from the highlight list.

[2523] Semantic client to stop exploding complex search queries (KIS preferably handles this)

[2524] In one embodiment, the semantic client attempts to explode complex queries. The KIS handles all complex Boolean logic so the Librarian doesn’t have to do this.

[2525] Highlighting with categories that have single or double quotes)
In one embodiment, the XPath query uses double-quotes (consistent with the XPath spec).

Export and/or import speed up with ontology downloads and hit cache included.

In one embodiment, the semantic client excludes ontology and/or highlighting hit cache state from import/export. The Librarian can regenerate the hit cache after an import.

Overview

In one embodiment, the KIS uses the system thread-pool and EACH KC runtime object preferably has its own semaphore. This ensures that the KCs don’t overwork the KDSes yet increases concurrency by allowing multiple KCs to index as fast as possible simultaneously.

In one embodiment, the central KIS runtime manager holds/increments a work reference count on each document sourced from each connector that may be currently indexing (it releases/decrements it once it is done indexing the document).

Ads in news feeds can be problematic because they can affect the ability of the KIS to semantically filter and/or rank properly. For instance, some web pages contain several times (at times more than 5 times) as much ad content as the actual content for the article. Here is an example: [http://www.npr.org/templates/story/story.php?storyid=47383044&sourceCode=RSS]

In one embodiment, this problem may be addressed in the following manner:

1. Assume that all articles contain ads. The news connector can indicate this in the generated RSS. The KIS takes this as a signal not to follow the link (this is what currently happens for Medline). Due to the KIS’ Adaptive Ranking algorithm, the KIS may be able to semantically rank on a relative basis so that the “best” descriptions can still be returned first. From looking at the metadata, the size distribution may be all over the map but is acceptable (there are many meaty descriptions). Optionally, advantageously, the descriptions for the Life Sciences channel tend to be very meaty.

2. Implement a Safe List. The Safe List may be manually maintained initially. This can contain a list of publisher names that don’t include ads. A good example is the Business-Wire which includes press releases. We can manually maintain the Safe List as part of our ASP value proposition. The News Connector can check the Safe List and/or if the publisher is deemed safe, can indicate to the KIS that it can safely index the entire document.

3. Automate the Safe List. A set of algorithms to attempt to automate the population and/or maintenance of the Safe List. This involves populating a Safe Candidate List, which can then be periodically scanned by humans. Humans can ultimately be responsible for what goes into the Safe List. The auto-population may be based on detecting those URLs that have “Printable Page” links. If these are detected, the connector can indicate to the KIS that it is to index the printable pages. These generally don’t contain ads.

4. Content-cleansing uses heuristics, machine learning, and/or layout analysis to automatically detect whether a page has ads. If ads are detected, the service can then attempt to extract the subset of the document that may be the meat of the document (as text) and/or then indicate to the KIS (via RSS signaling) that the KIS is to index that document.

In one embodiment, a combination of all three processes can address the issue.

The following are rules that may be used in various embodiments of the invention. They may be practiced independently or in combination and/or may be altered as may be necessary.

Ad-Removal Rule #1

For every HTML page (I have code for this—a URL not in the HTML exclusion list or a URL that has a query [uri uri=new Uri(url)]; if ((uri.Query != "String.Empty") && (uri.Query != "")); . .

If the web page contains a link (walk the link list using SmlReader, which converts HTML to XHTML)—see last URL, I emailed you; use XPath to walk the list) with any of the following titles (case-insensitive comparison):

1. “Text only”
2. “Text version”
3. “Text format”
4. “Text-only”
5. “Text-only version”
6. “Text-only format”
7. “Format for printing”
8. “Print this page”
9. “Printable Version”
10. “Printer Friendly”
11. “Printer-Friendly”
12. “Print”
13. “Print story”
14. “Print this story”
15. “Printer friendly format”
16. “Printer-friendly format”
17. “Printer friendly version”
18. “Printer-friendly version”
19. “Print this”
20. “Printable format”
21. “Print this article”

And if the link is not JavaScript (which launches the print dialog) . . .

Add the linkToIndexed tag to the generated RSS and/or point it to the printable link.

Alternate embodiments also detect the “print” icon with the “print” tool tip (or any tool tip with text mapping to any of the above), and/or apply the same rule.

Ad-Removal Rule #2

Cache the stats on host names for which rule #1 works. Add the host names to a “safe list candidates” file. We then need to validate those candidates and/or add them to the safe list. You also add items to the safe list based on submissions from trusted people (e.g., within Nervana and/or Beta customers).

Ad-Removal Rule #3

Apply the current rules (per description length, etc.) 

| Since these also save network I/O |
| If the item is recommended for addition: |
| If the hostname for an item is in the safe list, |
| Add it as “follow” with the inserted linkToBeIndexed tag |
| Else |
| Run rule #1 |
| If the item is a safe candidate |
| Add the host name to the “safe candidate list” file |
| (if it’s there already - use a hash table for quick comparison) |
| Add it as “follow” with the inserted linkToBeIndexed tag |
| Else |
| Add it as “noindex” |
| Else |
| Add it as “nofollow” |

As users/testers use the KCs, and/or if they see a pattern of content that don’t contain ads, they can email the URL and/or the Publisher (via the Details Pane) to Nervana to
add to the Safe List. Over time, this can accrete and/or can increase the recall of the system.

These ad removal and/or cleansing rules can also be employed at the semantic client during Dynamic Linking (e.g., Drag and Drop or Smart Copy and Paste). For example, if the user drags and drops a Web page, the cleansing rules can first be invoked to generate text that does not contain ads. This may be done BEFORE the context extraction step. This ensures that ads are not semantically interpreted (unless so desired by the user — this can be a configurable setting).

FIGS. 60 and 61 illustrate sample tables that may be present in various embodiments of the invention.

There may be also a composite index which is the primary key (thereby making it clustered, thereby facilitating fast joins on the SemanticLinks table since the database query processor may be able to fetch the semantic link rows without requiring a bookmark lookup) and which includes the following columns:

1. SubjectID
2. PredicateTypeID
3. ObjectID

FIGS. 62 through 65 illustrate examples of various embodiments of the invention, that are operable, for example, to:

1. Find me Breaking News on Chemical Compounds Relevant to Bone Diseases — Dossier on "*:bone diseases" chemical
2. Find me Breaking News on Cancer — Dossier on *cancer
3. Find me Breaking News on Cancer-Related Clinical Trials — Dossier on "*:clinical trials" :cancer
4. Find me Breaking News on Bacteria — Dossier on *bacteria

In one embodiment, the Life Sciences News KC can periodically ask the General News KC (during its real-time indexing process) for Breaking News on *:Health OR *:Health Care OR *:Medical Personnel OR *:Drugs OR *:Pharmaceutical Industry OR *:Pharmacology OR *:Medical Practice

This way, we can have chained Breaking News.

In one embodiment, a KC was populated based on editorial rules, based on tags provided by our news provider, to determine which sources and/or articles may be Life-Sciences-related.

When there is Life-Sciences-related content in General News (or other combination) that needs to be indexed in Life-Sciences News, this can be accomplished using KIS-Chaining. The Life Sciences (LS) News KC can ALSO point to the General News KIS via the preferred KIS RSS interface. The RSS can include a reference to *:Health OR *:Health Care OR *:Medical Personnel OR *:Drugs OR *:Pharmaceutical Industry OR *:Pharmacology OR *:Medical Practice

These come from the General Reference and Products & Services ontologies, which the General News KC may be indexed with.

The LS News KC can index the Health subset of the General Reference KC. This way, we use our own technology for domain-specific filtering.

Other vertical KCs (e.g., IT, Chemicals, etc.) can also employ the same approach to ensure they have the most relevant yet broad dataset to index. And that way, we don’t rely too much on the tags that come from Moreover to figure out which articles may be Life-Sciences-related.

[2588] In one embodiment the approach described below may be set for the IT News KC and/or ALL Vertical KCs.

[2589] The approach can also be used to funnel (or tunnel, depending on your perspective) traffic from the General Patents KC to the Life Sciences Patents KC (and/or other vertical Patents KCs in the future).

[2590] In one embodiment, we track the traffic for Breaking News for the following categories (ORed) from General News and/or compare that with the traffic on Breaking News on the Life Sciences KC.

[2591] We can then funnel content from the General News KC to the Life Sciences News KC via machine-to-machine KIS Chaining as described.

[2592] It is OK if these categories represent overly broad context. The Life Sciences News KC can still do its job and/or semantically filter and/or rank the articles according to its 6 Life Sciences ontologies. This may be akin to chaining perspectives and/or then performing “perspective switching and/or filtering” downstream.

| Clinical Tests of Medical Procedures | OR | Clinical Tests of Medical Procedures OR Drugs
| Drugs | OR | Forensic Medicine OR Group Medical Practice (all contents) OR Health
| Health | OR | Health Care OR Health Insurance OR Home Medical Tests OR Medical Equipment OR Medical Ethics OR Medical Examiners OR Medical Expense Deduction OR Medical Malpractice OR Medical Personnel OR Medical Records OR Medical Research OR Medical Savings Accounts (all contents) OR Medical Schools OR Medical Screening OR Medical Supplies OR Medical Technology OR Medical Wastes OR Pharmaceutical Industry OR Pharmacology OR Preventive Medicine OR Sports Medicine OR Telemedicine OR Biological Clocks OR Biological Diversity (all contents) OR Biology OR Biologists OR Biological and Chemical Weapons (all contents) OR Biotechnology OR Agricultural Biotechnology OR Genetics OR Anatomy and Physiology OR Animal Care OR Animals OR Aquatic Life OR Births OR Chemicals OR Child Care OR Child Development OR Children and Youth OR Cognition and Reasoning OR Contamination OR Death and Dying OR Environment OR Farming OR Females OR Flowers and Plants OR Food |
(Dielectrics OR Ceramic materials OR Dielectric materials) AND
(Mobile phones OR Telecommunications OR Handy OR Cellular phone OR Portable phone)
(Wireless communication OR Cordless communication OR Radiophone) AND (Microwave
High frequency OR High power OR High pulse OR High waveband)
and other combinations . . . no wonder it’s so expensive and time consuming.

In one embodiment, this may be done with a powerful, natural semantic query:
Check out the Engineering ontology in the semantic client. It has everything needed for this query: “dielectric materials” AND “microwave filters” AND “cellular telephone systems”
The painful keyword search below may be replaced by a simple Nervana semantic search on an Engineering Patents KC indexed with the Engineering ontology for:
“.dielectric materials”AND“.cellular telephone” AND “microwave filters”
In addition, the Information Nervous System adds multi-dimensional semantic ranking which may be currently a manual (and almost impossible) task.
The following are sample queries used in various embodiments of the invention.
Find me News on chemical compounds relevant to the treatment of bone diseases: Dossier on “*.bone diseases” *.chemicals
Find me News on chemical compounds relevant to the treatment of musculoskeletal or heart diseases: Dossier on *.chemicals AND (“*.musculoskeletal diseases”OR “*.heart diseases”)
Find me News on autoimmune, cardiovascular, kidney, or muscular diseases: Dossier on “*.autoimmune diseases” OR “*.cardiovascular diseases” OR “*.kidney diseases” OR “*.muscular diseases”
Find me latest News on work Pfizer, Novartis, or Aventis are doing in cardiovascular diseases: Dossier on “*.cardiovascular diseases” AND (Pfizer or Novartis or Aventis)
Find me latest News on cell surface receptors relevant to all types of Cancer: Dossier on “*.cell surface receptor” *.cancer
Find me latest News on enzyme inhibitors or monoclonal antibodies: Dossier on “*.enzyme inhibitors” OR “*.monoclonal antibodies”
Find me latest News on genes that might cause mental disorders: Dossier on “*.genes” “*.mental disorders”
Find me latest News on ALL protein kinase inhibitors or biomarkers but only in the context of cancer: Dossier on “cancer:protein kinase inhibitors” OR cancer:biomarkers
Find me latest News on Cancer-related clinical trials: Dossier on “*.clinical trials” *.cancer
Find me latest News on clinical trials on heart or muscle diseases: Dossier on “*.clinical trials” AND (”.: heart diseases” OR “*.muscle diseases”)
I want to track news on the Gates Foundation’s Grand Challenge titled “Develop a genetic strategy to deplete or incapacitate a disease-transmitting insect population”. Dossier on *.genetics *.diseases *.insects
[2629] I want to track news on the Gates Foundation’s Grand Challenge titled “Develop a chemical strategy to deplete or incapacitate a disease-transmitting insect population”: Dossier on *chemicals *diseases *insects

[2630] Find me research news highlighting the role of genetic susceptibility in pollution-related illnesses: Dossier on *genetics *pollution *diseases

[2631] Find research by Amgen or Genentech on chemical compounds used to treat autoimmune diseases: Dossier on AutoImmune Diseases (MeSH) AND Chemical (CRISP) AND (Amgen OR Genentech) as this works today (another common example is to filter by year e.g., (2004 or 2005))

[2632] Find research by Roche or Pfizer published in the past three years on the use of protein kinase or cyclooxygenase inhibitors to treat Lung or Breast Cancer: Dossier on (**:Protein Kinase Inhibitor” OR “**:cyclooxygenase inhibitor”) AND (**:lung Cancer” OR “**:Breast Cancer”) AND (Roche or Pfizer) AND (range:2003-2005). Here is an alternative that can work across ALL unstructured data repositories: Dossier on (**:Protein Kinase Inhibitor” OR “**:COX Inhibitor”) AND (**:lung Cancer” OR “**:Breast Cancer”) AND (Roche or Pfizer) AND (range:2003-2005). Here is a more specific alternative: Dossier on (**:Protein Kinase Inhibitor” OR “**:COX Inhibitor”) AND (**:lung Cancer” OR “**:Breast Cancer”) AND (affiliation:Roche or affiliation:Pfizer) AND (pubyear:2003-2005)

[2633] In one embodiment, *: may be a preferred and very powerful way for expressing semantic queries in Nervana and provides as close to natural-language queries as may be computationally possible.

[2634] In one embodiment, *: provides semantic stemming and semantic reasoning to INFER what terms MEAN IN A GIVEN CONTEXT IN A GIVEN PROFILE, NOT synonyms or other word forms of the terms.

[2635] In one embodiment, the Information Nervous System (read: The Nervana System) also semantically ranks results with *: queries IN THE CONTEXT of the desired terms/concepts. In the preferred embodiment, this may be NOT the same as mapping the query to a long Boolean query nor may it be the same as ranking the synonyms of the terms.

[2636] In one embodiment, a Dossier on **:bone diseases” AND **:chemicals may be NOT mathematically equivalent to a Boolean search for every type of bone disease (OREd) AND every type of chemical (OREd) BECAUSE OF CONTEXT-SENSITIVE RANKING.

[2637] In one embodiment, to increase recall, the KIS (on indexing incoming content from news feeds and other sources) adds the following logic:

[2638] 1. If you cannot extract the description and the metada description may be empty, mark it as unsafe for follow. Then add the “safe” column to the composite constraint that includes Title and Accessible.

[2639] 2. If a particle comes in with the same title as something you have already attempted to extract and the preferred one can be extracted, you replace the one that failed with the preferred one.

[2640] 3. Mark [http URLs as unsafe to follow (preferably but optionally requiring subscription)]

[2641] Logging Searches, Privacy, and Smarter Ontology Tools

[2642] In one embodiment, with privacy provisions, the KIS can *anonymously* log semantic searches and use those logs to improve our ontologies.

[2643] In one embodiment, actual searches are a great window to actual REAL-WORLD vocabularies being used—including typos and/or other word-forms that our ontologies might currently lack.

[2644] In one embodiment, this idea relates to an end-to-end ontology improvement service/system (with a Web application and/or Web services) that can allow ontologists to view logs and/or statistics and/or loop that back into the ontology improvement process. This may be tied to an ontology management tool via Web services. An ontology research and/or development team that can own the statistical analysis of search logs, ontology semi-automation, and/or distributed ontology development tools. The ontology tools have collaboration functions and/or to be tied into online communities and/or Wikis. Customers may be able to recommend ontology improvements from the Librarian and/or Web UI and/or have that propagated to the ontology analysis and/or development team in real-time.

[2645] Deny potential Denial-of-Service Attack when range: tag is used

[2646] In one embodiment, the KIS can not go beyond 1000 numbers in the range tag to guard against a DOS attack. This number may be adjusted as may be necessary.

[2647] In one embodiment, Deep Info Hyperlinks may be a visual tool in the Information Nervous System, used to complement the Deep Info pane. Deep Info Hyperlinks allow the user of the semantic client to navigate Deep Info not unlike navigating hyperlinks. This allows the user to be able to continuously navigate the semantic knowledge space, via Dynamic Linking, without any limitations based on the size of the knowledge space (which could exceed the amount of available UI real estate in say, a tree view). There may be a Deep Info stack to track “Back,” “Forward” and/or “Home”. For non-root category nodes in Deep Info, there may be an enabled “Up” button to allow the user to navigate to the parent category in a given ontology.

[2648] In one embodiment, Deep Info results (actual documents, people, etc.) can be restricted to the first major level in the tree (i.e., a result does not have a tree expansion which then shows more results—in the same in-place tree UI). Context templates (special agents or knowledge requests) can be displayed, along with previews of results there from, but thereafter the user can navigate to the template itself (e.g., Breaking News) to get more information—e.g., discovered categories with the template/special-agent as a pivot. Category hierarchies can be reflected in the tree as deep as may be needed. The user can navigate to a result, category, etc. and/or then continue the navigation from there—without overloading the UI.

[2649] Deep Info Hyperlinks may be indicated with the underlined text. Also, notice the Back, Forward, Stop, Refresh, Home, Mail, and/or Print buttons (no different from a hypertext web browser). The user may be able to navigate the Deep Info knowledge space (via Dynamic Linking) by recursively clicking on the Deep Info Hyperlinks and/or by going “Back” and/or “Forward,” as desired. Clicking Home would take the user back to the starting “Deep Info position” (either for application-wide or profile-wide Deep Info or to the context point from where the Deep Info semantic chain was launched). Clicking Refresh would refresh the Deep Info pane, not unlike refreshing a loaded web page in a Web browser. Clicking Stop would stop the pane from loading.
Clicking Mail would email the Deep Info XML contents to a person or group of persons. Clicking Print would print the Deep Info pane.

In one embodiment, the Deep Info Hyperlinks also have a drop-down menu to allow the user launch a new request (or entity) corresponding to the clicked Deep Info node.

Furthermore, in one embodiment, each entry in the Deep Info Hypertext space may be a legitimate launch point for a new request, bookmark, or entity. The user may be able to create a new request, bookmark, or entity (opened in place or “explored”—opened in a new window). The system intelligently maps the current node to a request, bookmark, or entity, based on the semantics of the node. For instance, a category may be mapped to a Dossier on that category (by default and/or exposed in the UI as a verb/command) or a “topic” entity referring to the category (as another option, also exposed in the UI as a verb/command). A context template (special agent or knowledge request) can be mapped to a request with the same semantics and/or with the filter based on the source node (upstream) in the Deep Info pane. Some nodes might not be “mappable” (e.g., a category folder) and/or the UI indicates this by disabling or greying out the request launch commands in such cases.

In one embodiment, the clipboard launch point for Deep Info can be automatically updated when the clipboard changes (via a timer or a notification mechanism for tracking clipboard changes) or can be left as is (until the user refreshes the Deep Info Pane). In one embodiment, the semantic client keeps track of the most recent N clipboard items (via the equivalent of a clipbook) and/or have those exposed in the Deep Info pane. The most recent clipboard item may be displayed first (at the top). The “current” item then may be auto-refreshed in real-time, as the clipboard contents change. Also, if the current item on the clipboard (or any entry in the clipbook) may be a file-folder, the Deep Info pane allows the user to navigate to the contents of that folder (shallowly or deeply, depending on the user’s preference).

In one embodiment, there may be at least two Deep Info Panes with Hypertext Bars—a main pane that would encapsulate the entire semantic namespace and/or which may be displayed everywhere in the namespace (in every namespace item console) and/or a floating pane (the Deep Info Minibar) which may be displayed next to a selected result item. The main pane allows the user to semantically explore all profiles but the current (contextual) profile may be displayed first (highest in the tree, in the case of a tree UI, perhaps after the current request and/or clipboard contents Deep Info launch points). The Deep Info Minibar may be displayed when the user selects an item (perhaps via a small button the user must click first) and/or has only the result item as an initial launch point (so as not to overload the UI). Also, the Deep Info Minibar includes a Deep Info path with “Annotations” off the result item itself (in addition to all the context templates and/or other Deep Info paths). The Minibar also allows the user to explore—off the result item as a launch point—both the current (contextual) profile and/or other profiles in the system. The user be able to semantically explore Deep Info across profile boundaries.

Legend:
Blue: Ontology (Category Folder) for discovered category
Red: Parent category for discovered category
Green: Discovered category

In one embodiment, the Deep Info pane flags each category in the hierarchy as belonging to Best Bets, Recommendations, or All Bets. This allows the user to visually get a sense of the strength of the Deep Info path (in this case a category) IN THE CONTEXT of the strength of the categories IN THE CONTEXT of the query or document (or the Deep Info source). This may become a hint to the user per how much time and/or effort to spend navigating different paths. So in the example below, the user can have a clear sense that Cardiac Failure may be a Best Bet category, Dementia may be a Recommended category, and/or that Immunologic Assays may be an All Bets category. Also, there may be a visual indicator showing if a category is [also] in the news (e.g. Dementia below)—the sample picture shown reads “NEW!” but in practice reads “NEWS.” There may be also an indicator alongside each category folder showing the total category count, and/or the count for Best Bet, Recommended, and/or “In the News” categories. This provides the user with a visual hint as to the richness of the category results within a specific category folder (ontology) before he/she actually explores the category folder.

In one embodiment, in the case where a semantic wildcard query (or a category query) may be the Deep Info source, the hints represent the relevance of the inferred cat-
egories in the corpus itself. Else, in the case of a document, the clipboard, text, etc., the hints represent the INTERSECTION of relevance of the inferred categories in the source AND the corpus (the index). As an illustration, if the Deep Info source may be a document, the Best Bet hint for a Deep Info category may only be set IF the category (or categories) may be Best Bets in BOTH the source document AND the corpus. Ditto for Recommended categories (the category has to be at least a Recommendation in both source and/or destination). Else, the hint may be indicated as All Bets.

[2656] It guides the user to the relevance of the categories ALONG the path, consistent with BOTH source and/or destination. If the category may be weak in the source yet strong in the corpus, the intersection can tell the user same. If the category may be strong in both, this may be clearly the path to navigate first.

[2657] Here is an example, in accordance with an embodiment of the invention (see the legend below):

```
[+] Current Request (Dossier on "Cardiac Failure")
AND **(Cardiac Failure) AND **(Dementia) AND **(Immunologic Assays)
[+] MeSH (15 total, 1 Best Bet, 4 Recommended, 2 in the News)
[+] Cardiovascular Diseases
[+] Cardiac Failure
[+] Mental Disorders
[+] Dementia
[+] Immunologic Techniques
[+] Immunologic Assays
```

Legend:
Blue: Ontology (Category Folder) for discovered category
Red (Bold): Parent category for discovered Best Bets (very strong relevance) category
Green (Bold): Discovered Best Bet category
Red: Parent category for discovered Recommended (strong relevance) category
Green: Discovered Recommended category
Dark Grey: Parent category for discovered All Bets (weak relevance) category
Light Grey: Discovered All Bets (weak relevance) category

[2658] In one embodiment, the model (as described above for flagging categories in context via visual hints) also applies to People. Experts may be to be treated as Best Bets on the People axis, Interest Group may be treated as Recommendations on the People axis, and/or Newsmakers may be treated as Headlines on the People axis.

[2659] In one embodiment, for a Person object in the Deep Info pane, the same model applies. However, the visual hints preferably indicate relevance based on Expertise, Interest, and/or News (per newsmakers). These visual hints for discovered categories may be displayed IN ADDITION to the context templates (special agents or knowledge requests) also displayed for the Person/People in question. In the preferred embodiment, the symmetric (People) visual hints also supplements the Information hints (Best Bets, etc.). The visual hints may be based on direct equivalences in the semantic networks in the KIses in the contextual profile—indeed the Category information returned in the Deep Info query has identical attributes to the BestBetHint, RecommendationHint, BreakingNewsHint, and/or HeadlinesHint in the semantic network. These attributes indicate whether the category is a Best Bet category, a Recommended category, a Breaking News category, or a Headlines category. In one embodiment, the KIS goes further and/or also return a hint to the semantic client indicating whether the Deep Info source (e.g., John Smith) below is a “Best Bet” (expert per semantic symmetry), “Recommendation” (interest group per semantic symmetry), Breaking News (breaking newsmaker per semantic symmetry) and/or Headlines (newsmaker per semantic symmetry). The KIS accomplishes this by querying for these hints from categories in the Objects table (or Categories table in an alternate embodiment) and/or joining this against the People table with the filter indicating whether the person (“John Smith” in this case) has a semantic link to the category.

[2660] An illustration of the People visual hints is shown below, in accordance with an embodiment of the invention. The balloon tool tips show additional Deep Info visual hint qualifiers on the People axis, specifically related to the Person in question (in this case, John Smith).

```
[+] John Smith
[+] MeSH (15 total, 1 Best Bet, 4 Recommended, 2 in the News, 1 Expert, 2 Interest Group, 1 Newsmaker)
[+] Cardiovascular Diseases
[+] Cardiac Failure
[+] Mental Disorders
[+] Dementia
[+] Immunologic Techniques
[+] Immunologic Assays
```

[2661] In one embodiment, In Deep Info, as illustrated in the figure above, the user often starts from a category and/or then navigates from there. However, this can be problematic because the category might not be “understood” (i.e., the category’s ontology might not be supported) in other Knowledge Communities in the contextual profile. Semantic wildcards get around this because the interpretation of the context may be performed on the fly—the categories may be inferred in real-time and/or not explicitly specified.

[2662] In one embodiment, in Deep Info, it may be preferable to preserve the seamlessness of the user experience by supporting intelligent and/or dynamic navigation. With documents and/or text (and in some cases, entities), this happens automatically—Dynamic Linking already involves real-time inference and/or mapping of categories. However, with categories as the source context, things get a bit trickier for the reason described above. To address this, the Information Nervous System supports Intelligent Dynamic Linking If the source category is not understood (as explicitly specified), the KIS can indicate this in the Deep Info result set. However, the KIS can go a step further: it can then attempt to map the explicit category to semantic wildcards simply by adding the “*” prefix to the category name (off the category path). It can then rerun the Deep Info query and/or then return the result set for the new query to the semantic client. The new result set may be tagged as having been dynamically mapped to semantic wildcards. The semantic client can then display a very subtle hint to the user that the Deep Info results were inferred on the fly by the system. Some users might not care, especially if the category name is strong and/or distinct enough to communicate semantics regardless of the contextual path and/or the ontology. Some users, however, might care, especially if the explicit source category is unique and/or distinct from other contexts that might share the same category name.

[2663] In one embodiment, Dynamic Deep Info Seeding allows the user to seek to Deep Info from any piece of text. First, the user may be able to hover over any highlighted text (with semantic highlighting) and/or then dynamically use the highlighted text as context for Deep Info—the semantic client
can detect that the text underneath the cursor is highlighted and/or then use the text as context. The result may be selected (if not already) and/or the Deep Info mini-bar invoked with the highlighted text as context (with semantic wildcards added as a prefix—for intelligent processing). This creates a user experience that feels as though the user seeks (without navigating) from a highlighted term to Deep Info on that term.

In one embodiment, this feature may be also extended to hovering over any piece of selected text. The user can select the text, hover over it, and/or then seek to Deep Info using the text as context.

In one embodiment, anywhere people may be exposed in Deep Info (including in the Deep Info mini-bar), Presence information may be integrated as an additional hint. This indicates whether a displayed user is online, offline, busy, etc. The Presence information may be integrated using an operating system (or otherwise integrated) API. Verbs may be also be integrated in the Deep Info UI to allow the user to see a displayed user and/or then open an IM message, send email, or perform some other Presence-related action either directly within the Deep Info UI or via an externally launched Presence-based or IM application.

In one embodiment, the Geography ontology allows semantic regional scoping/searching. This allows queries like Dossier on American Politics from General News. This may be invoked as Dossier on *American *Politics. Other examples may be:

1. Dossier on Investments in Asia □ Dossier on *Asia *Investments
2. Dossier on Caribbean or African Vacations □ Dossier on *Vacations AND (*:African OR *:Caribbean)

In one embodiment, we have an Institutions ontology that has every company name, school name, etc. We can use the Hoover’s database as an initial reference. This can then be added to all General KCs.

In one embodiment, a combination of the following ontologies: General Reference, Products & Services, Geography, and/or Institutions provide very rich semantic coverage.

1. The “Make me an ontology” Red Button
2. The “Does this ontology suck?” Red Button

In one embodiment, this button can allow a Martian who just landed on Earth to create the first pass for an ontology describing previously unknown knowledge domains on Mars. Coming back to Earth, it would allow Nervana to generate a new ontology for domains or sub-domains, perhaps new industries like nanotech, etc.

In one embodiment, the scientific and/or product development part of this involves creating the Red Button to CONSTANTLY scan through documents on the Web and/or other sources and/or generate the ontology based on high-level taxonomic and/or conceptual inferences that can be made. The generated ontology may only be a first pass; humans may have to then follow up to refine the ontology.

2. The “Does this ontology suck?” Red Button

In one embodiment, this button can allow a user to quickly determine the quality of an ontology. For all our current ontologies, what is the grade? Which gets an A? And which gets an F? Which ontology is so bad that it shouldn’t be used in production, period? And why? What is the basis for determining A, B, C, D, E, or F? What is the scale and/or how are grades determined? These grades can then be used for our ontology certification and/or logo program. This can be employed for ontology comparison analysis (A.) are two ontologies semantically similar and if so, how much? B.) is ontology A better than ontology B for knowledge domain K and if so, by how much, and why?). This button may be tied into a real-time ontology monitor. This monitor can constantly track search logs and/or web logs to determine if an existing ontology may be getting stale or may be otherwise not representative of the domain of knowledge it represents. Search lingo changes and/or the vocabulary around a knowledge domain changes; the real-time ontology monitor can make the “Does this ontology suck?” red button also a “Does this ontology still not suck anymore?” button.

3. The “Fix this ontology” Red Button

In one embodiment, similar to the “Make me an ontology” red button, this button can allow a user to take an existing ontology, integrate it with the real-time ontology monitor, and/or have recommendations made on how to fix or improve the ontology.

1. In one embodiment, the KIS understands the following qualifiers:

author: (this restricts the search to the author field)
publisher: (or pub:) this restricts the search to the publisher field
language: (or lang:) this restricts the search to the language field
host: (or site:)—this restricts the search to the host/site from where the item originated
filetype:—this restricts the search to the file extension (e.g., filetype:pdf)
title:—this restricts the search to the title field
body: this restricts the search to the body field
pubdate:—the publication date
pubyear:—the publication year
range:—a number range (format □ range: <start>-<end>.)
affiliation:—the affiliation of the author(s) (e.g., Merck, Pfizer, Cetek, University of Washington)

In one embodiment, you can combine these filters at will. The model may be also completely extensible—more filters can be added in a backwards compatible way without affecting the system.

For example, Dossier on Heart Diseases AND lang: eng AND “author :long bh”—find all English publications on Heart Diseases authored by Long BH.

In one embodiment, each qualifier has a corresponding predicate which indicates the basis for the semantic link, linking a document (or other information item) to the concept in question. FIG. 66 illustrates the mapping of the qualifiers to predicates (the actual predicate values may be arbitrary but must be unique).

In one embodiment, semantic wildcards (and/or dynamic linking in general) defer semantic interpretation until run-time (when the query is getting executed). In contrast, a category reference (Uri) has a hard-coded expression for semantic interpretation. Hard-coded category references have the problem of brittleness, especially in the context of ontology versioning. A category path or URI might become invalid if an ontology’s hierarchy fundamentally changes. This could become a versioning nightmare. With semantic wildcards (or drag and drop), on the other hand, there may be no hard-coded path or URI (the wildcards refer to concepts/terms that can be interpreted across ontologies and/or ontology versions). This is very powerful because it means that an ontology can evolve without breaking existing queries. It is also powerful in that it more seamlessly allows for ontology
federation—with different ontologies in a virtual network of Knowledge Communities (KCs)—each wildcard term may be interpreted locally with the results then federated broadly.

[2694] In one embodiment, events awareness refers to a feature of the Information Nervous System where the system understands the semantics of events (end-to-end) and/or applies special treatment to provide event-oriented scenarios.

[2695] 1. In one embodiment, there may be Events Knowledge Communities—for instance, Life Sciences Events. This may be similar to Web KC offerings like Life Sciences Market Research and/or Life Sciences Business Web, Life Sciences Academic Web, and/or Life Sciences Government Web.

[2696] Life Sciences Events can allow knowledge-workers semantically keep track of research conferences, marketing conferences, meetings, workshops, seminars, webinars, etc. For instance, questions like: Find me all research conferences on Gastrointestinal Diseases holding in the US or Europe in the next 6 months.

[2697] In one embodiment, the query above can involve the Geography ontology (as described above) to allow location-based filters that may be semantically interpreted.

[2698] In one embodiment, this Knowledge Community (KC) can be seeded manually and/or then filled out with additional business-development (as needed). The seedling would RSS integration (where available) and/or editorial tools (screen-scraper) to generate Event metadata (as RSS) which can then be indexed on a constant basis.

[2699] In one embodiment, a special RSS tag indicates to the KIS that an event “expires” at a certain date/time and/or after a certain time-span. When the event “expires” in the KC, the KIS automatically removes it.

[2700] This idea is also useful with e-Commerce KCs—imagine a semantic index of Sales Events—where a sale might “expire” and/or become unavailable to users of the index.

[2701] In one embodiment, the semantic client may be “aware” of results that may be events and/or can allow users to add events to their Outlook Calendar (or an equivalent). This can be done via a VerbTask on a selected “event result.”

[2702] In one embodiment, the WebUI client allows users to set reminders for events. The WebUI then emails them just before the event occurs (with a configurable window, not unlike Outlook). So for example, a user may be able to register for reminders (semantic reminders, if you will) for the sample query I indicated below.

[2703] In one embodiment, the KIS supports self-aware, expiring events, as described above.

[2704] In one embodiment, the KIS and/or the semantic clients also support a new field qualifier, location:, that allows the user to specify the desired location of an Events semantic search. This maps to a new predicate, PredicateTypeID_LocationContainsConcept. Also, these may be a startdate:, enddate:, and/or duration: (event duration) qualifiers with corresponding predicates.

[2705] In one embodiment, Drag and Drop dynamic query generation applies to entities, semantic wildcards, smart copy and paste and/or other Dynamic Linking invocation models. As noted previously, the query generation rules can result in sequential queries.

[2706] In one embodiment, when there are multiple SQML filter entries that may require dynamic semantic interpretation and/or query generation, the resultant query can be very complicated. For performance reasons, the following query reduction/simplification rules may be employed, in accordance with one embodiment of the invention:

[2707] If there is only one SQML filter entry, the previously described rules may be employed.

[2708] If there are multiple SQML filter entries and/or the operator is an OR, the previously described rules may be employed. The resultant queries may be then concatenated into a master sequential query set. This overall query set may then be invoked, with eventual result duplicates elided.

[2709] If there are multiple SQML filter entries and/or the operator is an AND, the resultant-query generation rules may be a bit more complicated. If there are multiple Best Bet categories generated from the source (the “dragged” object), the categories may be added to a resultant list. Else, if there is one Best Bet category, the category may be added along with Recommendations categories (if available). Else the Recommendations categories may be added to the resultant list (if available). Else, the All Bet categories may be added (if available). If there are non-semantic entries (as previously described)—for instance key concepts in the title or body—these may be also added to the resultant list. This may be repeated for all SQML filter entries. The resultant categories may be then added to one master semantic query, which may be then invoked with an AND operator.

[2710] If there are multiple SQML filter entries and/or the operator is an AND NOT, the rules described for AND (above) may be generated and/or then the resultant query may be modified to have an AND NOT operator rather than an AND operator. These steps may be altered or changed as may be necessary.

[2711] In one embodiment, there are multiple semantic clients that access services exposed by the Information Nervous System. In one embodiment, this may be done via an XML Web services interface. There may be two additional semantic clients: the Nervana WebUI and/or the Nervana RSS interfaces. These have several strategic benefits including, for example: Low Total Cost of Ownership (no client install); No/minimal training for massive deployments (familiar, Web-based interface); Client flexibility (rich (Librarian) vs. reach (WebUI)); shows programmatic flexibility (system can be programmed/accesses with different clients); and Migration path (can start with WebUI; and/or then migrate to Librarian for power-user scenarios).

[2712] In one embodiment, the RSS interface may be also exposed via [HTTP] and/or can be consumed by standard RSS readers. Currently, the RSS interface emits RSS 2.0 data.

[2713] In one embodiment, the figure below shows an illustration of the WebUI. Notice the command-line interface with semantic wildcards—this provides a lot of the semantic power via a text box. Also, notice the integration of the Dossier Knowledge Requests to provide different contextual views of results.

[2714] In one embodiment, any WebUI query can be saved as an RSS query which emits RSS 2.0. This can then be consumed in a standard RSS reader. The RSS interface automatically creates a channel name as follows: Nervana <Knowledge Request> on <Filter>, where <Knowledge Request> is the knowledge request type (Breaking News, Best Bets, etc.), and/or filter is the search filter. FIG. 67 illustrates a WebUI interface, in accordance with an embodiment of the invention.

[2715] In one embodiment, the Infotype semantic search qualifier may be a powerful and/or special qualifier that may be used to specify information types in the Information Ner-
vous System. The user can ask for Breaking News but only those that may be Presentations. This may be specified as Breaking News on InfoType:Presentations.

[2716] In one embodiment, the KIS adds special info predicates corresponding to each information type. This can be a abstraction on top of filetypes—both predicate classes may be added to the semantic network. Furthermore, some infotypes yield other infotypes—e.g., a presentation may also be a document; in such cases, multiple predicate assignments may be issued. Because the infotype predicates may be in the semantic network, they can be mixed and/or matched with other predicate qualifiers, knowledge types, etc. For instance, a user can ask for Best Bets on InfoType:Spreadsheets AND “author:John Smith” (find me best bets that are spreadsheets authored by John Smith). Here is a sample list of InfoType predicates:

[2717] PredicateTypeID_InfoType_Presentation
[2718] PredicateTypeID_InfoType_Spreadsheet
[2719] PredicateTypeID_InfoType_GeneralDocument
[2720] PredicateTypeID_InfoType_Annotation
[2721] PredicateTypeID_InfoType_AnnotatedItem
[2722] PredicateTypeID_InfoType_Event

[2723] In one embodiment, semantic type semantic search qualifiers may be like infotype qualifiers except that the qualifier tags themselves indicate the semantic type. This makes it clear to the KIS that only a specific predicate based on entity-detection is employed. For instance, “person:john smith” indicates to the KIS that only a concept that has been detected to refer to a person may be included in the semantic search. Or place:houstons indicates only a place called Houston and/or not names called Houston. And so on. This information may be added to the semantic network by the KIS via semantic type predicates. Examples may be:

[2724] PredicateTypeID_SemanticType_Person
[2725] PredicateTypeID_SemanticType_Place
[2726] PredicateTypeID_SemanticType_Thing
[2727] PredicateTypeID_SemanticType_Event

[2728] In one embodiment, time search qualifiers are predefined and/or semantically interpreted qualifiers that refer to absolute or relative time. These don’t have to be (nor are they—in the case of relative times) hard-coded into an ontology—they can be interpreted in real-time by the KIS. The KIS then maps these qualifiers to an absolute time (or time range) IN REAL-TIME (resulting in a live computation of the actual time value) and/or then uses the resultant value in the semantic query. Examples can include for example:

[2729] 1. “pubdate: last week”
[2730] 2. “pubdate:today”
[2731] 3. “pubyear: this year”
[2732] 4. “pubyear: last decade” (may be dynamically mapped to a range: query)
[2733] 5. “startdate: next week” (for events)
[2734] 6. “duration: two weeks”

[2735] Examples of queries that may be enabled by time search qualifiers include:

[2736] Find all events on mathematical models for climate change holding in California next week: All Bets on “**mathematical models” AND “**climate change” AND location:California AND “startdate: next three months” (Notice that this query also includes the Geography ontology (for the California filter))

[2737] Find all presentations for request for proposals for communications equipment in the next quarter: All Bets on infoType:presentations AND “**communications equipment” AND ““next quarter”

[2738] In one embodiment, time ontologies allow the semantic interpretation and/or inference of time-related concepts. Examples of time-related concepts may be: “twentieth century,” “the nineties,” “summer,” “winter,” “first quarter,” “weekend” (terms for Saturday and/or Sunday), “weekdays” (leave for Monday through Friday), etc. This can allow queries such as:

[2739] Find all presentations for deals that closed in the third-quarter: All Bets on sales AND infoType: presentations AND “third quarter”

[2740] Find research on quantum physics done by Nobel Prize winners in the second half of the twentieth century: Recommendations on “**quantum physics” AND “**nobel prize” AND “second half of the twentieth century”

[2741] In one embodiment, the triangulation of Time ontologies with Geography ontologies (as described above) covers the space-time continuum, which is part of reality.

[2742] In one embodiment, a similar model may be also applied for numbers—Number Ontologies. This enables queries with concepts like “six-figures,” “in the millions,” etc. This may be also be implemented with number search qualifiers.

[2743] In one embodiment, historical ontologies may be like Time ontologies but rather focus on time in the context of specific historical concepts. Examples:

[2744] Ancient China (concepts that describe all the places and/or other entities in Ancient China)
[2745] Pre-colonial Africa
[2746] Renaissance

[2747] In one embodiment, institutional ontologies may be used as a generic ontologies (like Geography). These have businesses, universities, government institutions, financial institutions, etc. AND their relationships. Sample queries:

[2748] Find Breaking News on cancer research but only that done by Big Pharma
[2749] Find research on bacteria being done by any company affiliated with Merck (research partners, acquired companies, etc.)
[2750] Find Breaking News on job openings in technology companies but only those on the Fortune 500
[2751] Find great papers on Gallium Arsenide based semiconductor research but only by accredited European institutions

[2752] Another example:

[2753] Find great articles on the possible use of semantics to improve research productivity in Life Sciences but only published by Industry Leaders

[2754] This involves the notion of “institutional people” (thought leaders, executives, influencers, key analysts, etc.), in all humility, which may be semantically correlated with an Institutions ontology.

[2755] In one embodiment, this ontology may be also useful to semantically search for companies and/or other institutions referred to by acronyms (e.g., GE). Also, this ontology handles common typos. Example: “Bristol-Myers Squibb” (correct spelling) vs. “Bristol Myers-Squibb” (very common typo).

[2756] In one embodiment, this ontology may be critical for IP searching, for which the ownership of IP is very important.

[2757] In one embodiment, a query like: [Find all patents on manufacturing techniques for polymer-based composites owned by DuPont] brings back patents by DuPont AND
companies that have been "acquired" by DuPont—since DuPont will preferably own the IP.

[2758] In one embodiment, Commentary and/or Conversations may be treated differently in terms of their semantic ranking and/or filtering algorithms. This may be because they may be based on publications, annotations, etc. from people in the Knowledge Communities (KCs). The involvement of people may be a critical axis that determines the basis for relevance. For example, take an email message with the body "Sounds good." or even something as short as "OK." In a typical knowledge community using only ontology-based semantic indexing, ranking, and/or filtering, these messages might be interpreted as being irrelevant or weakly relevant. However, if the author of the email message is the CEO of the company (and/or the knowledge community corresponds to that company) or if the author is a Nobel Prize Winner, all of a sudden the email message "takes on" a different look or feel. It all of a sudden "feels" relevant, independent of the length of the text or the semantic density of the words in the text.

[2759] In one embodiment, another way to think of this may be that in knowledge communities, the author or annotator of an information item might contribute more to its "relevance" than the content of the item itself. As such, it may be dangerous merely to use ontologies as a source of relevance in this context.

[2760] In one embodiment, the Dynamic Linking model of the Information Nervous System partially addresses this because the user can navigate using different semantic paths to reach the eventual item—the paths then become a legitimate basis for relevance, in addition to—or regardless of—the semantic contents of the item itself.

[2761] In one embodiment, several changes may be made to the KIS indexing algorithms when indexing commentary or conversations, for example:

[2762] The semantic threshold may be set to zero—all items may be indexed.

[2763] The ranking may be biased in favor of time and/or not semantic relevance (not unlike email)

[2764] An alternative to a formal Commentary context template (knowledge request) may be to have All Bets ranked by time and/or not semantic relevance—only, perhaps, for a specially defined and/or configured "Discussions" knowledge community (that may be treated differently)

[2765] In one embodiment, a model for comparing and/or mapping ontologies may be present. The model described here will generate a map that shows several (2 or more) ontologies may be similar (or not). Given N ontologies O1 through ON, create N semantic indexes (using the Information Nervous System) of a large number of documents (relevant to a reasonable superset of the knowledge domains that correspond to the ontologies) using each ontology. For every category in each ontology and/or for each document in the corpus, generate a table that with columns for Best Bets and/or Recommendations. These columns will indicate the semantic strength of the category in the given document.

[2766] In one embodiment, once these tables may be generated, a separate set of steps may be invoked to map categories across the ontologies, for example:

[2767] For every source category that may be a Best Bet, find every category in every other ontology that may be a Best Bet. Assign a high score (e.g., 10) for this mapping. For parents of the target categories, assign a high but lesser score (e.g., 8). An additional scalar factor (weakening the score) can be applied for broader categories (moving up the hierarchy chain).

[2768] For every source category that may be a Recommendation but may be not also a Best Bet, find every category in every other ontology that may be either a Recommendation or a Best Bet. Assign a median score (e.g., 6) for the former (Recommendation) mapping and/or a slightly higher score (e.g., 8) for the latter (Best Bet mapping). For parents of the target categories, assign a high but lesser score (e.g., 4 and 6, respectively). An additional scalar factor (weakening the score) can be applied for broader categories (moving up the hierarchy chain).

[2769] For every source category that may be an All Bet but may be neither also a Recommendation nor a Best Bet, find every category in every other ontology that may be an All Bet, a Recommendation, or a Best Bet. Assign a median score (e.g., 2, 4, and 6, respectively) for these mappings. For parents of the target categories, assign a high but lesser score (e.g., 1, 2, and 3, respectively). An additional scalar factor (weakening the score) can be applied for broader categories (moving up the hierarchy chain).

[2770] Categories that don’t qualify based on the above rules may be assigned a score of 0.

[2771] In one embodiment, all the scores may be tallied. For every category, a ranked list of every category in every other ontology may be generated (from highest to lowest scores, greater than 0). This then represents the ontology assignment/comparison map. The larger and/or more relevant the corpus to the entire ontology set, the better. This map may then be used to map categories across ontology boundaries—during indexing.

[2772] In one embodiment, federated and/or merged semantic notifications refers to a feature of the Information Nervous System that allows users to have rich semantic notifications from a federation of knowledge communities, organized by profile, and/or across a distributed set of servers.

[2773] In one embodiment, every KIS can be configured with a master notification server that it then communicates notifications too (based on a polling frequency and/or on registered user semantic-requests). Federated identity and/or authentication may be used to integrate user identities. The master notification servers then merge all the notification results, eliminate duplicates, and/or then notify the registered user.

[2774] Alternatively, the user can register for notifications from specific KISes (or KCs) which can then notify the users (via email, SMS, etc.).

[2775] Alternatively yet, these notifications can be sent to a Notification Merge Agent which lives centrally on a special KIS. This merge agent can then mark all the source profiles (by GUID), merge and/or organize the notification results by profile, and/or then forward the merged and/or organized results to the registered user.

[2776] In one embodiment, this refers to a feature to allow the user to get semantic wildcard equivalents from the semantic client categories dialog. The categories dialog can have a "Copy to Clipboard" button—enabled only, perhaps, when there may be selected categories. When this button is clicked, the selected categories may be copied to the clipboard as text. For example:

[2777] If "Heart Diseases" and/or "Muscular Diseases" are selected as categories, the following may be copied to the clipboard as text:
In one embodiment, the user can then go back to the edit control in the standard request or the command line on the Home Page and/or click Paste. The user can then change the text to AND, add parentheses, change the wildcard to a specific ontology alias qualifier (e.g., Cancer or MeSH), etc.

In one embodiment, this may be the semantic client namespace item serialization model and/or file formats—for Request, Results, and/or Profiles (and/or other non-container namespace items) Saving and/or Sharing (e.g., email):

In one embodiment, a request may be saved (or emailed) as a Zipped folder (read: an easily sharable file). When we have critical mass, we can have our own extension (.req) which we actually reserved a couple of years ago.

In one embodiment, the Zipped folder can contain the following files and/or folders:

In one embodiment, results (this folder can contain the results as they were when they were saved):

[Request Name].XML (the results as RSS)

If the request is a Dossier, there may be one XML file for each request type:

[Request Name].HTM (the results saved as an HTML file)

If the request is a Dossier, there may be one HTML file for each request type:

The HTML file may be a report generated from the results XML. It can have lists and/or a table showing each result and/or its metadata. Also (from a usability standpoint), it can have hyperlinks to the result pages, which a TXT file would not have.

In one embodiment, request (Original Profile) (this folder can contain the XML (SQML) that represents the semantic query/request AS IT WAS WHEN IT WAS SAVED)

[Request Name].XML

The request XML can contain all the state in the original request, including the KCS for the request profile. This allows other users to view the identical request, since their profile information might be different.

Request Info.HTM (this file can describe the request, its filters and/or the original profile, including the names of its KCS and/or category folders)

This file can also contain the metadata for the request—e.g., the creation date/time, the last modified date/time, the request type, the profile name, etc.

In one embodiment, request (Any Profile) (this folder can contain the XML (SQML) that represents the semantic query/request without any profile information) The request XML can contain all the state in the original request, but only, perhaps, with the request filters, excluding the KCS for the request profile. This allows other users to view the request in their own profiles, if the filters are what they find interesting.

Request Info.HTM (this file can describe the request and/or its filters)

This file can also contain the metadata for the request—e.g., the creation date/time, the last modified date/time, the request type, etc.

In one embodiment, Readme.HTM

This file can describe the contents of the folder

This file can also contain the metadata for the request—e.g., the creation date/time, the last modified date/time, the request type, etc.

NOTE: In one embodiment, the Zipped folder name can be prefixed with “Nervana.”

Example: Nervana Dossier on Cell Cycle AND Protein Folding.ZIP

In one embodiment, a similar model may be employed for serializing profiles—profiles contain folders with each request, in addition to the profile settings.

Why the ZIP Format?

1. Allows seamless pass through the majority of data systems that screen out unknown or suspicious file types (this precludes us from having a custom file type until post critical mass)

2. File makes for ease of sharing, saving, and/or management

3. Internal folder structure allows for rich metadata display with multiple views of the request state (in files and/or sub-folders)

4. ZIP is an open format with broad industry support. ZIP management may be preferably built into Windows XP allowing for easy management of the saved request and/or results. Furthermore, there may be many third-party ZIP SDKs for customers that might want to generate reports from saved Nervana requests/results. For example, a customer might want to write an application that scans through file or Web folders containing saved Nervana requests/results, extracts the contents from the ZIP folders, and/or then manipulates, analyzes, aggregates, or otherwise manages the saved RSS results within each zipped folder. So a customer (say, Zymogenetics) can have an application that monitors a shared folder, opens the zipped Nervana folders, and/or then aggregates the RSS results (from different requests) to, say, database tables or spreadsheets for analysis.

5. Compression: Because many of the elements in the saves folder is in the XML, ZIP can result in a very high (and/or significant) compression ratio (up to 10:1 from published studies/reports and also from my experience).

6. Malleability and Extensibility: ZIP can provide backward and/or forward compatibility for the “format.” Old versions of the Librarian may be able to “open” requests from previous versions and/or vice versa. ZIP would also allow us (in large measure) to add and/or remove components from the “format” without affecting the core of the “format.”

In one embodiment, Newsmakers refers to authors of inferred news (within one or more agencies or knowledge communities) in a given context. Newsmakers may be “known” (provable identities) within a user’s knowledge communities. Newsmakers may be members of agencies (knowledge communities) so a user can continue to navigate with a newsmaker as the virtual pivot object—a user can find a Newsmaker, navigate to Headlines by that Newsmaker, drag and drop one of those Headlines to find semantically relevant Best Bets, navigate to the Interest Group for one of those Best Bets, etc.

In an alternative embodiment, Newsmakers can also be people featured in the news—the system maps extracted concepts, performs entity detection to detect names, and/or attempts to authenticate those names against names in the system. The system can then assign a similar (but not identical) Newsmaker predicate that indicates that the semantic link has uncertainty (e.g., PREDICATE_TYPE(MIGHTBE-NEWMAKERON). The “Newsmaker” context template query can then include this predicate as part of the Newsmaker query—but in some cases, the predicate can also be excluded (this model preserves flexibility). In the preferred embodiment, the authors may be authenticated by their email address so this problem wouldn’t occur.
In one embodiment, Newsmakers may be authenticated authors (and/or members of the agency (knowledge community)). A separate “In the News” query can be generated for entities (including unauthenticated people) that may be featured in the news.

In one embodiment, RSS Commands/Verbs may be special signals embedded in RSS that direct the KIS to take actions on specific information items. These may be specified with namespace-qualified elements that correspond to specific verbs that the KIS invokes. Examples:

meta:insert or meta:remove (instructs the KIS to index the RSS item)
meta:delete or meta:remove (instructs the KIS to delete the RSS item)
meta:update (instructs the KIS to update the RSS item)

Let $n$ be the total number of keywords that are semantically relevant to all the filters in the query. And let $k$ be the number of semantic or keyword filters in the query.

In the general case, the order of magnitude of total number of combinations may be by which the $n$ items can be arranged in sets of $k$ may be represented by the formula:

$$C^*_n = k!,$$

where

$$P_k = \frac{n!}{(n-k)!}$$

Also, note that in this case, we use combinations and not permutations because the order of selection for semantic queries does not matter (A AND B=B AND A).

For union (OR) queries, this count may be accurate. For intersection (AND) queries, and/or if there are multiple filters, the exact count may be less than this (although of the same order of magnitude) because exclusions must be made for the keyword combinations within the same category filter.

Example: Take the semantic query: Find all chemical leads on bone diseases which are available for licensing.

This can be expressed in Nervana as: All Bets on Bone Diseases (MeSH) AND Chemical (CRISP).

In the text-box interface, this can also be expressed as a search for "Bone Diseases" AND Chemical. Alternatively, this can be expressed as a cross-ontology search for "*Bone Diseases" AND Chemical. But we can focus on the ontology-specific searches here in order to simplify the analysis.

Bone Diseases (MeSH) currently has a total of 308 keywords representing the many types of bone diseases and/or their synonyms and/or word variants. Chemical (CRISP) has a total of 5740 keywords representing the very many number of chemical compounds and/or their synonyms and/or word variants.

Adding the keyword “licensing,” this amounts to a total of $6049$ keywords.

Assuming 2 keywords per search, and/or plugging into the equation above, this result can be represented in the following:

$$P_2 = \frac{(6049-2)!}{(6049-2)!} = 36584352$$

Therefore, $C_2 = 36584352/2! = 18292176$

In other words, it can take approximately $18.3$ million 2-keyword searches to approximate the semantic query represented above (even discounting semantic ranking, filtering, and/or merging). And because these are 2-keyword queries, the quality of the search results (even in the non-semantic domain) can suffer greatly.

Assuming 3 keywords per search, and/or plugging this into the equation above, this can result in the following:

$$P_3 = (6049-3)!/6049! = 18292176$$

Therefore, $C_3 = 18292176/3! = 36870929424$

In other words, it can take approximately 36.9 billion 3-keyword searches to approximate the semantic query represented above (even discounting semantic ranking, filtering, and/or merging). Adding a third keyword would likely improve the quality of the search results (even in the non-semantic domain). But this results in an even more exponential explosion in the number of keyword searches necessary to fully exhaust all the possibilities encapsulated in the semantic query. 4-keyword searches can result in an astronomical number of searches. And so on.

And then multiply this by the different kinds of queries (like Breaking News, etc.). So if the researcher wants the results grouped in, say 6 contexts, the total may be 6 times the number of keyword queries shown above. And then multiply this by the different silos of knowledge over which the researcher must repetitively search. This represents the total astronomical number of searches required to approximate a federated Nervana Dossier.

Matters are made worse yet as the queries get more complex. For instance, if the query was: Find all chemical leads applicable to both Bone and Heart Diseases which are available for licensing, this would correspond to a Dossier on Bone Diseases (MeSH) AND Heart Diseases (MeSH) AND Chemical (CRISP) and ‘licensing’. The combinations can explode to an even more astronomical number because the value $n$ above would be much higher due to the number of keywords that represent all the types of Heart Diseases.

In one embodiment, to efficiently index real-time newsfeeds, a staging server hosts a daemon which downloads news items and/or then indexes them in an intermediate staging index. This index may be then divided up into multiple channels—allowing for indexing scale-out (with each KIS indexing one channel). More channels can then be added to provide more parallelism and/or less simultaneous read-write (while indexing)—in order to improve both query and indexing performance.

Examples of channels may be: LifeSciences, GeneralReference, and InformationTechnology.

Examples of corresponding URLs may be:


In one embodiment, the connector’s ASP.NET page takes an additional parameter Since, also case-insensitive. The format of time may be yyyy-mm-ddTHH:mm:as. For example: 2005-06-29T16:35:43. This can be easily obtained in C# by calling date.ToString("s"), where date may be an instance of System.DateTime structure. The paging parameters may be as earlier: Start and PageSize.
[2846] In one embodiment, the connector emits RSS 2.0 data which may be mapped from the staging index (with the news items). The RSS 2.0 data indicates that the data may be from a Nervana Data Connector. There may be also a paramsSupported field which indicates to the KIS which parameters the connector supports. Once the KIS downloads the RSS, it parses it. It then checks to see if the RSS is from a Nervana Data Connector. If it is, it then checks the paramsSupported field. If this is populated, it then checks if the “since” parameter is one of the comma-delimited items in the field. If the “since” parameter is found, the KIS then makes note of the current time. It continues to index the RSS and/or page through until it reaches the end of the RSS stream. At that time, and/or when the KIS starts re-indexing (the next time), it adds the since parameter to the connector URL query string with the time indicated above (the time since when the “last” indexing round began). This may be akin to the KIS asking the connector for only those data items that it (the staging index) has added “since” the last indexing round. This is a very efficient way to incrementally index news in real-time—it ensures that only new items are indexed without the I/O overhead of a full incremental index.

[2847] Here is a snippet from an RSS 2.0 item generated from a News connector:

```xml
<?xml version="1.0" encoding="utf-8"?>
<rs version="2.0" xmlns:xsi="[
[http://purl.org/dc//elements/1.1/"
] xmlns:meta="[http://schemas.nervana.com/xmlns/rs_2_0_meta.html">
  <channel>
    <title>GeneralReference2</title>
    <category>Nervana Data Connectors</category>
    <generator>News Open Database for SQL</generator>
    <meta paramsSupported="Channel,Start,PageSize,Since,Filter,NDays,
    Order"/>
    <meta paramsSupported="Channel,Start,PageSize,Since,Filter,NDays,
    Order"/>
    <meta starIndex="/home/"/>
    <meta resIndex="/home/"/>
    <language en-us"/>
    <item>
      <meta robots="nofollow"/>
      <dc:language>en-US</dc:language>
      <title>Oxford student murdered in honour killing</title>
      <pubDate>10/6/2005 11:43:00 PM</pubDate>
      <author/>
      <dc:publisher>The Tribune</dc:publisher>
      <description/>
      <guid isPermaLink="false">4024614554</guid>
    </item>
  </channel>
</rs>
```

[2848] The nofollow meta tag may be added accordingly, based on whether the link is accessible or not.

[2849] In one embodiment, the Nervana Knowledge Center may be a Federated universe of Nervana-powered content, providing the transformation of Information to Knowledge. The Knowledge Center has semantically indexed content, People (in a future version), and/or annotations (also in a future version). In various embodiments of the invention, any of the following may be included:

[2850] 1. Smart News (General News and Domain-Specific News)
[2851] 2. Smart Patents (General Patents and Domain-Specific Patents)

[2853] 4. Smart Marketplace: This may be the e-commerce scenario and/or includes sponsored listings that may be semantically indexed. The KCs therein may be first-class KCs (with people, annotations, etc.). I contend that if there is enough value in the content and/or the medium, people can independently subscribe (the one person’s ad is another person’s content scenario 1 described recently). Examples include: Products; Jobs (postings and/or resumes).

[2854] 5. Nervana-Run Research KCs (e.g., Semantic/Smart Medline).

[2855] 6. Nervana-Run Domain and Scenario-Specific KCs: Examples include Compliance, Sarbanes-Oxley, etc.


[2857] 8. Smart Libraries: This may be where we partner with content providers like Science Direct, Elsevier et al. who have been looking for premium revenue channels for many years. There may be two possible models here. In one model, they provide abstracts and/or maybe full-text to us since we drive revenue to them via smarter discovery. We can host the KCs and/or own/manage the initial consumer relationship. In another model, they can host KCs themselves and/or pay us licensing fees for our technology. NOTE: Smart Libraries preferably can have ALL the tools in the toolbox. They may be first-class Knowledge Communities, they can have people, they can have annotations, etc. See more below.

[2858] 9. Smart Groups: Smart Groups may be like a semantic (knowledge-oriented) equivalent of blogs. The scenarios here are numerous. There may be many thousands of knowledge communities around the world—on everything from gene research to fly-fishing. Users can first sign up (maybe for $5 a month) as members of the Nervana Network. As a member, you may be then able to create and/or moderate Smart Groups. Smart Groups may be different from regular groups (like Yahoo Groups) or blogs in that:

[2859] They may be semantically and/or context-aware. Knowledge types like Interest Group, Experts, Newsmakers, Conversations, Annotations, Annotated Items, provide semantic access to community publications and/or annotations.

[2860] Semantic threads à Conversations become first-class semantic objects that can be returned, ranked, and/or navigated.

[2861] The Knowledge Toolbox: All the tools in our toolbox à Breaking News, Live Mode, Deep Info, etc. can be applied to Smart Groups. These tools do not apply to regular (information) groups on the Web.

[2862] Semantic navigation (Deep Info): Emphasis is due here. Smart Groups can be semantically navigated via Deep Info. The semantic paths may be at the knowledge level.

[2863] Dynamic Linking: Users may be able to navigate from their desktop to Smart Groups, to say, Newsmakers within those Groups, to the annotations by those Newsmakers, and/or then to relevant knowledge IN DIFFERENT KNOWLEDGE COMMUNITIES—all at the speed of thought.
[2864] Awareness: Live Mode and the Watch List display Newsmakers. Newsmakers may be actionable—so a user can see Newsmakers and/or immediately start to navigate/explore.

[2865] Federation: Client and server-side

[2866] Examples of Smart Groups: Research communities, virtual communities across companies (including partners, suppliers, etc.), classes in schools (e.g., working on specific projects), informal communities of interest around specific area, etc. Imagine a group of researchers that may be able to annotate results from Nervana Semantic Medline (after a Drug and Drop) in their own Smart Groups, and/or create semantic threads based on results from Medline, and/or then annotate Smart News results around those semantic threads.

[2867] 10. Smart Books: in partnership with a large aggregator like Barnes & Noble. Subscribe to a Nervana Smart Books KC and/or semantically finds books with semantic wildcards and/or the like. Dynamically link that to Smart Groups within (Smart Books à moderated by Nervana) OR your own Smart Groups (moderated by you or a friend/colleague).

[2868] 11. Smart Images: in partnership with a large aggregator like Getty or Corbis. Semantically find professional or amateur photographs by dragging and/or dropping a picture from your desktop. And then creating semantic threads around the pictures you find—with other hobbyists that like photography as much as you do (in your Pictures-based Smart Groups). The provider may be responsible for providing rich annotations to the books.

[2869] 12. Smart Media (Music and Video): in partnership with large music and/or video (including live broadcast) aggregators. The key value proposition here may be that reviews become semantic and/or context-aware. Communities of interest may be formed around music genres, movies, etc. This needs to be more tightly moderated because it may be more consumer-oriented. Preferably ALI the tools in the toolbox can apply.

[2870] In one embodiment, live mode may be a Watch List of one and/or may be aimed at providing awareness-oriented presentation for a specific request (including special requests and/or Dossiers) or request collection. It allows users to track timely results in the context of a request or request collection.

[2871] In one embodiment, the Presenter periodically issues queries to the KISes in the contextual profile for a request in Live Mode. A request can be in normal mode or live mode. The Presenter also sorts the results based on timeliness and/or provides additional functionality for handling News Dossiers (previously described) and/or for guarding against KC starvation in the case of federated profiles.

[2872] In one embodiment, the Presenter can have a configurable refresh rate and/or other awareness parameters. On the UI side, the skin polls the Presenter for results. The Presenter polls the KISes and/or then places the results in a priority queue (as previously mentioned). The skin then picks up the results and/or shows special UI to indicate recently added results, freshness spikes, an erosion of freshness (fade), etc.

[2873] In one embodiment, the Presenter guards against KC starvation in federated profiles by making sure results from a high-traffic KC don't completely drown out results from lower-traffic KCs. The Presenter employs a round-robin algorithm to ensure this.

[2874] In one embodiment, the Live Mode skin can choose to display the metadata for the results in its own fashion. In addition, the skin can creatively display UI to indicate the relative freshness and/or “need for attention” Attributes that can be modeled in the UI may be, in accordance with various embodiments of the invention:

[2875] 1. Activity: This indicates the rate of change of results.

[2876] 2. Freshness: This indicates how old an individual result may be. The skin can show UI for new results differently from old results (e.g., in brighter colors, bigger fonts, etc.)

[2877] 3. Spike Alert: A Spike Alert may be generated/fired when a new result is the first fresh result over a given period of time. The Presenter sets a timer; if the timer expires with no results then a flag may be set. The very next “fresh” result would trigger a Spike Alert in the UI. The arrival of a new result resets the timer. The Spike Alert may be designed to draw the user’s attention to a given result. The methods of drawing attention may include a small sound, a pop up alert window, a color change, or a movement of page elements.

[2878] In one embodiment, the semantic client and/or WebUI support the saving, exporting, and/or emailing of results. All results can be saved or exported or selected results can be.

[2879] In various embodiments of the invention, some of the following features may be present.

[2880] 1. Only those results that have been cached—but NOT those on the screen. If the user clicks Next and/or then Previous, the cache expands and/or all the cached results may be selected.

[2881] 2. For the WebUI, we save from the server-side cache. For the semantic client, the client-side cache. In one embodiment, there may be no need for any communication to the server for saving at the Librarian.

[2882] 3. File formats: All Results Lists may be RSS (XML, cross-platform). Reports may be HTML (portability, cross-platform, no need for special clients, etc.). However, Dossiers may be saved in zipped folders. The folders can contain N+1 files (RSS and/or HTML, depending on the user’s selection), where N is the number of open Dossier requests (<6) and/or 1 represents the “All” list which may be a merged list of results (duplicated elided). Zipped folders provide a single thicket model (ease of sharing, ease of file management, etc.), they may be portable, cross-platform and/or pass through firewalls (most firewall extensions filters allow zips to pass through)—for email sharing. All results may be prefixed with ‘Nervana’ (e.g., Nervana Breaking News on ‘*cancer *kinases’). The user can then rename the file/folder. The HTML reports may be also branded with our logo and/or tagline and/or the logo may include a hyperlink to our web site—for viral marketing.

[2883] 4. In the preferred embodiment, we invoke a mailto: url with no recipient and/or then an auto-embedded attachment with the files/folders AND semantically relevant message title. The user is then to fill out the recipient, etc. In an alternative embodiment, there may be additional UI to provide forms—the user can do this in his/her email client. Email clients like Outlook have other features the user might want to use during the sending process (sending to an email list, validating the list, ccing to others, etc.).
In one embodiment, this infrastructure can then be used for semantic alerting—in one embodiment, the user registers his/her email address(es) and/or semantic wildcard (or other) queries. The semantic client or WebUI can then email (or via some other notification channel) periodic breaking news or headlines results to the user. These may be in HTML or RSS, as described above.

In one embodiment, the Email Companion Agent may be an agent that employs the email notification infrastructure described above and/or may be a companion to an existing distribution list. So the admin can create a distribution list to track semantic topics and/or the companion agent can email breaking news and/or headlines to the list on a periodic basis, consistent with the semantics of the distribution list.

In one exemplary embodiment, self-aware documents may be documents—using the Information Nervous System—that generate their own live, semantic references. This employs the Dynamic Linking functionality of the Information Nervous System but embeds the logic in documents themselves (the document “drags and drops itself” in real-time). A document can be configured to dynamically link to one or more knowledge communities (federated). Imagine a self-aware research paper that generates its own references. The references are as good—in the general case, with arbitrary papers—as references the author generates him or herself. This passes the Turing Test ([http://en.wikipedia.org/wiki/Turing_test](http://en.wikipedia.org/wiki/Turing_test)) and/or may be a test for whether P=NP ([http://www.claymath.org/millennium/P_vs_NP](http://www.claymath.org/millennium/P_vs_NP))

In one embodiment, self-aware documents can “call” into the semantic client runtime to invoke Dynamic Linking in real-time—as they are displayed. Imagine a research paper emailed around with live, semantic references. This is extremely powerful because the value of the paper changes over time—as the surrounding “semantic environment” changes. The documents can be configured with authentication information that may be passed into the semantic client runtime. The argument to the Dynamic Linking APIs may be the “self” URI (the document itself).

In one embodiment, semantic profiles may be wrappers around entities. For instance, a semantic profile can be built for a company (based on relevant documents, filed patents, etc.). And then semantic screening refers to tracking incoming and/or outgoing information (including documents) and/or correlating the information to one or more semantic profiles. For instance, a company might build semantic profiles for companies involved in ongoing patent litigation and/or then set up screening rules to ensure that no document leaves the company relevant to the litigation. Similar rules can be set up for incoming traffic.

Deploy Combinatorial Filters: Manage combinatorial complexity; Provide manageable, meaningful, probabilistic, ranked inputs into Disease Model; Inputs into a stochastic model; Deploy Early Warning Systems; Decision Support; Diseases to target? Projects to keep? Licensing, M&A opportunities? Safety, IP issues? Signaling systems (biomarkers, toxicogenomics, etc.); Build Drug Discovery Libraries; Research, patents, safety studies, factoids, etc.; Enable Knowledge Feedback Loop. Optimally must filter data inputs that are: Mostly unstructured text (85%); Physically fragmented; Semantically fragmented, e.g., phenotype data; Multidimensional; Full of Uncertainty, Context, and Ambiguity; Must understand and reason; Targets, phenotypes, etc. are semantic entities; NOT keywords; Provides meaning-based drug discovery and early-warning. Computers cannot reason without understanding.

Combinatorial Hypotheses: Examples include Drug Discovery: Find anticancer agents that induce apoptosis; Find small molecule drugs for spinal cord injury; Find chemicals that prevent the initial signaling and chemical reactions that turn on the immune system; Find chemicals that inhibit the migration of inflammatory cells to joint tissues; Safety: Find preclinical data for recently approved cancer drugs employing monoclonal antibodies.

Ontologies: Describe knowledge domains; Basis for semantic interpretation; Necessary but NOT sufficient; Needed: Ontologies+Combinatorial Filter; Filter: Handles combinatorial mathematics, Use ontologies as inputs; Avoid extremes of ontological simplicity & complexity; Simple enough but not too simple; “Semantic loss”; Complex enough but not too complex; “Semantic overkill”; Yet more mathematical complexity.

Why not keyword search? Does NOT address combinatorial complexity; Rather, it monetizes it (via advertising); No semantics=no discovery; Hypotheses are semantic! E.g., find chemicals that inhibit the migration of inflammatory cells to joint tissues; Keyword search results are a mirage; a very poor first-level approximation; “Lucky” results (OK for consumers, bad for research); “Objects are less relevant than they appear.”

Why not manual tagging? Scale; Humans cannot keep up with combinatorial explosion; Multi-dimensionality; Problems have multiple axes; Single-ontology tagging is insufficient; E.g., PubMed/MedSH; Context and ranking; Semantic evolution and unpredictability; Must separate content from semantic interpretation.

Why not federated keyword search? Makes a bad problem worse. Exposes MORE combinatorial complexity; Does not address semantic fragmentation; E.g., different expressions of phenotype data; Creates more problems than it solves.

The Semantic Web. W3C semantic integration effort; Good ontology standards (e.g., OWL); But . . . does not address unstructured data (85%); Ignores the hardest problems: Knowledge representation; Combinatorial ranking & filtering; and Reasoning under uncertainty & ambiguity.

Strategic Imperative: Refine your Business Processes. “Knowledge Audits”: Processes, Metrics and Accountability; Best Practices, Due Diligence: R&D: What is the history of similar efforts? What lessons have been learnt? Are we reinventing the wheel? Early Warning; Competitors, M&A, Licensing, Clinical Trials, Safety, IP, etc.; Collaboration is now mission-critical; Collective intelligence.

In one embodiment, Call to Action Phase I: Start with External Data; Deploy Combinatorial Filters; Deploy Early-Warning Systems; Use well-known ontologies; Start building Discovery Libraries; Corresponding to hypotheses; Across silos. Phase II: Refine your business processes; Processes, Metrics and Accountability; Design Knowledge Audits. Phase III: Unlock your internal data. Phase IV: Define your knowledge domains; Develop or license ontologies for your domains; Open Biological Ontologies; [http://obo.sourceforge.net/](http://obo.sourceforge.net/); National Center for Ontological Research (NCOR); [http://necor.us/](http://necor.us/); Gene ontologies, HUGO, UMLS, FMA, etc.; Phase V: Add a semantic (ontology-based) layer atop your silos. Phase VI: Complete semantic integration platform; Deploy and federate combinatorial filters; Conduct
regular knowledge audits and enable a future of amazing possibilities. Imagine “Self-Aware Information” (documents, research papers and the like).

[2898] Decompress the R&D Bottleneck: Rising costs, lower productivity, expiring patents; Dire consequences; Proposed Drug Discovery Knowledge Architecture; Combinatorial Filters; Hypothesis validation; Orders of magnitude productivity improvement; Knowledge feedback loop; Discovery Libraries; Consistent with semantic hypotheses; Early Warning Systems; Mine your existing data; Refine your business processes; Enable a future of amazing scenarios; Scientific fact, not science fiction. All approaches at the linguistic layer have generally failed for the past 50 years. Problem reformulation: Natural Language Input expressed as a Directed Acyclic Graph (DAG) — G1. Indexed corpus stored using the identical representation — G2. The goal is to find the maximum common sub-graph isomorphism between G1 and G2.

[2899] G1 and G2 are potentially infinite. Infinite number of predicates and objects. Subject, Predicate, Object (SPO) Triple Model. Linguistic layer has infinite characteristics. Maximum Common Sub-graph Isomorphism (MCS) is NP-complete. Challenge is to solve an NP-complete problem in P. Problem statement: Find an algorithm in P (polynomial time) that solves the MCS problem. Query results — G3 which is isomorphic to G1 and G2 and is the maximum common sub-graph.

[2900] Client: Document/text extraction. Text compression and optional encryption. Server: Text categorization — using one or more ontologies, Naïve Bayes, SVM, LSI. Categories become objects with URIs. Build raw graph G1 with document/text as subjects and categories (ranked by semantic density) as objects; Graph redraw: Find G2 (a reduced representation of G1) that maintains the semantics of G1; Rank ranges (patent pending) — create new context predicates to build G2. Server: Graph collapsing. Remove semantic redundancies. Cross-ontology graph consolidation. Cluster categories that share the same semantics across ontology boundaries; Graph pruning. Prune G2 graph by histogram-based analysis of semantic density distribution to yield G1; Graph caching: Cache generated G1 graph using document/text hash as key into graph hash table, this way, rerun queries run much faster.

[2901] Prune graph cache using LRU algorithm. Server: Inexact graph matching: Map G1 to G2 (corpus) using ranked sequential queries (patent pending); Start from top edge and semantic intersect lower edges; Generate structured query; Use context predicate (e.g., Best Bets) to impose maximum commonality filter for sub-graph extraction (optimized for precision); Uses rank ranges to generate context predicates from raw predicates; Category as object (post ontology processing) means match is inexact; Inference engine has added new semantic links in corpus so match is inexact (optimized for recall). Stop at curve-knee of semantic distribution, if not enough edges, prune matching steps; If still not enough, fall back to non-semantic query; Repeat and stop at next higher edge; Synthesized results from each step and elide duplicates using hash table. Multi-graph matching (multi-drag and drop).

[2902] EXCLUSION (NOT): Merely exclude edges instead of a semantic intersect; e.g., find all patents on which this document does NOT infringe; INTERSECT: N input graphs G1, G2, . . . GIN; Apply algorithm for G1 through GIN; Join edges from each graph; Ignore non-overlapping steps; e.g., find all technical reports relevant to all 3 of these classic papers; UNION: N input graphs G1, G2, . . . GIN; Receder steps for sequential queries, ranked; Round-robin; Apply algorithm for G1 through GIN; With new reordered steps; Explore sequential queries; e.g., find all technical reports relevant to any of these 3 classic papers; Optional steps: Forward chaining in order to increase recall; Use ontology hints to guarantee safe chaining; Hint-less forward chaining is dangerous and is not recommended; Graph partitioning for very long documents; Ideally, use NLP or document object model to intelligently detect partitions; Chapters, Sections, Pages, etc.; Partition G1 into Gp1 . . . Gpn; Perform inexact graph matching for each sub-graph; Synthesize the results: Practical solution for P vs. NP problem; One of 7 unsolved problems in Mathematics; Clay Mathematics Institute Millennium Problems; Should pass the Turing Test: Use Drag and Drop to generate references for a research paper. If committee of domain experts can tell if the references were human (the author) or machine generated, then Nervana has passed the Turing Test. Algorithm has numerous applications: True semantic search & discovery, Image recognition, Cartographical analysis, Fingerprint detection, Protein folding, Cheminformatics and the like.

[2903] TalentEngine™. A critical and growing need in recruiting and staffing is that of sourcing and ranking the best and most qualified candidates to ensure the highest caliber work force to any organization. Nervana’s TalentEngine™ is a powerful new software based business tool that provides HR managers the most cost effective means of managing critical staffing, Discovery, Screening, and Ranking processes while significantly reducing costs typically incurred in identifying the best possible candidates from fragmented sources, domains, and databases.

[2904] This hosted “on-demand” service employs Nervana’s award winning artificial intelligence engine to automatically source resumes and curriculum vitae from fragmented sources including the internet, job boards, social networks, proprietary databases, and any targeted domain, and to match them to relevant positions. Resulting matches are ranked using novel and proprietary algorithms with unparalleled efficiencies (employing over one hundred variables available). TalentEngine™ Services assist HR managers to increase placement quality while streamlining associated workflows.

[2905] With Nervana’s natural-language-processing technology a custom job or target profile can be submitted as query and the TalentEngine™ aggregates ideal resumes, curriculum vitae, and user profiles from multiple open and accessible domains (delivering both active and passive candidates). The system then builds an intelligent semantic index based on domain-aware ontologies and numerous other variables (standard and custom) and performs automated screening and ranking based on semantics or meaning . . . not on keywords! This helps ensure that a candidate’s skills are matched in only the most relevant context, and also helps address the now common and misleading practice of “keyword stuffing” where candidates often populate their resumes with keywords independent of their qualifications. The best matches are then periodically published, stored and made available to the user. This empowers users with a complete sole-source solution to effectively manage recruiting and staffing management of sales, administration, technologists, and engineering professionals.

[2906] TalentEngine™ provides a single platform tool that delivers its user the capability to leverage artificial intelli-
gence to match criteria similar to human thought on a super computing scale, allowing HR Managers to focus on the most critical decisions and functions of HR processes. It guarantees human capable oversight (Quality Assurance and Control) across an expansive and fully automated set of Discovery, Screening, and Ranking processes that today can over stretch the precinets of limited HR resources.

[2907] ADVANTAGES include for example: Increase your Draw; Get the most out of your advertising and posting budget; No more “blasting”; No more missed prospects; Monitor multiple fragmented sourcing channels via an integrated platform; Increase your reach to the best qualified candidates, Discover the best qualified talent across multiple fragmented touch-points, Pushing vs. pulling; Reduce your Recruiting Costs: Drastically reduce labor costs by streamlining workflows and optimizing the use of human review, Get highly targeted, qualified candidates and minimize exposure to arduous “trial and error” keyword search, and resume-keyword stuffing and other manipulation techniques, Shorten your Time-to-Hire; Substantially shorten the time to identify and recruit the best qualified candidates in an extremely competitive labor market; Use existing resumes, bios, or cover letters as natural-language queries to complement or accelerate the use of job descriptions and to bolster laser-like targeting, Automated Ranking and Bull’s-Eye Scoring Techniques, Short list qualified candidate pools via statistical ranking by determining quantifiable variable summaries, Position & Industry specific custom or standard candidate scoring.

[2908] One embodiment of TALENTENGINE™ ARTIFICIAL INTELLIGENCE COMPONENTS may include Overall Candidate Relevance, Job Industry Relevance, Job Category Relevance, Job Experience Relevance, Job Skills Relevance, General Relevance, Red Flags, Custom Relevance(s).

[2909] Pricing and Features Examples:

[2910] 1. Annual User Access License: $1000 per seat per year

[2911] 2. Standard Edition: $500 per month per query

[2912] 3. Professional Edition: $1000 per month per query


[2914] 5. One embodiment of the Custom Edition may include: Premium Edition+$100 per custom variable per month.

[2915] Standard Edition may include, but is not limited to: Screening and Ranking (customer-provided resumes, referrals, and career web sites); Emailed Reports; RSS Feeds; Secure Report-Hosting Portal; Search within Reports; Report Diaries Professional Edition: Discovery, Screening, and Ranking: Web (resumes); Free Job Boards; Subscription Job Boards; Social Networks; Career Web Site; Referrals and Custom Databases; Premium Edition: Professional Edition plus: Nervana Resume Database; Relevant Blogs; Relevant News; Relevant Inventors; Relevant Scholars. Nervana TalentEngine™ provides HR Managers a paradigm shift to staffing workflow through the power of semantics and artificial intelligence.

[2916] The present invention relates to computers and, more specifically, to information management and research systems. Specific details of certain embodiments of the invention are set forth in the following description and in FIGS. 68-103 to provide a thorough understanding of such embodiments. In one embodiment, the system incorporates not only the features and functions described in my parent application, but also at least some of the additional features, enhancements and/or properties described in this application. The present invention may have additional embodiments, or may be practiced without one or more of the details described for any particular described embodiment.

[2917] FIG. 68 is a block diagram of a method for implementing semantic advertisements in an internet browser, in accordance with an embodiment of the invention. In one embodiment, the browser 102 is in communication with an information server 104, an information server 106, and an advertisement generating service 108. The browser 102 may be in communication with additional or fewer information servers as well as additional advertisement generating services. These servers may be located on a single piece of hardware or on multiple hardware components both locally or separated by distances. In one embodiment, semantic ads in the invention are implemented by integrating a client 102 with an advertisement generating service 108. The advertisement generating service 108 may be independently operated or part of the overall invention. Furthermore, the advertisement generating service 108 may be located on the internet or located on an intranet. In another embodiment, the advertisement generating service 108 hosts advertisements. The user of browser 102 invokes a query and that is submitted to the advertisement generating service 108. In one embodiment, the query from browser 102 is also sent to information server 104 or 106 to obtain content. The advertisement generating service 108 then accepts and interprets the incoming query request and responds with advertisements that are semantically relevant to the query request. In one embodiment, the advertisement generating service 108 functions similar to the systems for returning semantically relevant content results disclosed in the parent application. In this embodiment, one difference is that the advertisement generating service 108 returns semantically relevant advertisements rather than semantically relevant content results. As an example, a query for “data mining and security” information may result in the advertisement generating service 108 returning advertisements on data mining and security. However, the advertisement generating service 108 may also return other advertisements that are semantically relevant such as advertisements on data searching and encryption, SQL and firewalls, or other similar results. In one embodiment, advertisements are delivered from the advertising generating service 108 or displayed in the browser 102 based on semantic strength or the degree of relevance to the query. However, the advertisements may be delivered from the advertisement generating service 108 or displayed in the browser 102 based in lieu of or in addition to semantic relevance, including the categories or context distinctions disclosed in the parent application. Categories may include, but are not limited to, advertisements on breaking news on the query, advertisements from experts on the query, advertisements regarding interest groups on the query, advertisements based on popularity, most recent advertisements regarding the query, recommended advertisements based on the query, advertisements in headlines based on the query, or may simply be random advertisements. In another embodiment, the advertisements are delivered or displayed based upon the price paid for the advertising service. Context distinctions may include, but are not limited to, advertisements of people, events, documents, topics, books, products, projects, texts, file-shares, distribution lists, blogs, images, local file folders, or any other context. In an alternative embodiment, the browser 102 presents the advertisements in
a side panel, on part of the browser, on the whole browser, and
the advertisements may be stationary, moving, or dy-
namically updated.

[2018] FIG. 69 is a block diagram of a method for integ-
rating HTTP metadata and RSS metadata in an informa-
tion server, in accordance with an embodiment of the invention. In
one embodiment of the invention, an information server 202
is in communication with a website 204 and an RSS feed 206
wherein the information server 202 collects metadata from
both sources and stores it in a metadata database 208. The
invention is not limited to an RSS feed, but may include any
equivalent or alternate source of metadata. Furthermore, the
invention may involve multiple websites and RSS feeds. In
another embodiment, the information server 202 solicits
metadata from the website 204. Information server 202 then
stores the resulting metadata in the metadata database 208. In
yet another embodiment, the information server 202 solicits
metadata from an RSS feed 206. Information server 202 then
stores the resulting metadata in the metadata database 208. In
one embodiment, the information server 202 detects an RSS
feed 206 while crawling websites 204. In another embodi-
ment, RSS metadata from the RSS feed 206 complements
website metadata from website 204 in the metadata database
208. Alternatively, RSS metadata from the RSS feed 206
replaces the website metadata from website 204 in the meta-
data database 208. In a further embodiment, RSS metadata
from RSS feed 206 is organized using XML. In this embodi-
ment, the information server 202 validates the RSS feed using
an XML schema. In an alternative embodiment, the metadata
in the metadata database 208 is indexed according to the URI
from which the metadata originated.

[2019] FIG. 70 is a block diagram of a method for dynami-
cally making input suggestions based upon prior user input,
in accordance with an embodiment of the invention. In one
embodiment, a browser 304 accepts input from the query
input 302 and is in communication with a server 308. The
browser 304 provides feedback in the form of suggestions for
additional queries at block 306.

[2020] In another embodiment, the query input 302 is a
request for breaking news on Y and experts on Z. However,
the query may be any query, including, without limitation,
those disclosed in the parent application. In this embodiment,
the browser 304 accepts the query input 302 and browser 304
satisifies the query request with information from the server
308. However, in one embodiment, the browser 304 also
offers query suggestions 306 based upon the query input at
302. Query suggestions 306 based upon the query input 302
of breaking news on Y and experts on Z may include, but are
not limited to experts on Y, interest groups on Y, popular sites
on Y, headlines on Y, conversations on Y, events on Y, breaking
news on Z, interest groups on Z, popular sites on Z, headlines
on Y, conversations on Y, or events on Y. In a further embodi-
ment, the query input 302 is modified and submitted to
browser 304 based upon the query suggestions 306.

[2021] FIG. 71 is a block diagram of a method for present-
ing time sensitive information to a user, in accordance with an
embodiment of the invention. In one embodiment, informa-
tion from a favorites list 406, special requests 408, or current
information 410 is obtained from profile A 404. This infor-
mation is used to present time sensitive information to the
user from news display 412, 414, or 416. In an alternative
embodiment, information from favorites list 406, special
requests 408, or current information 410 is obtained from
other profiles such as profile B 402. This information may
also be used to present time sensitive information to the user
from news display 412, 414, or 416. These and many other
profiles may be used to obtain information.

[2022] In another embodiment, the news display 412 con-
tent is inferred or deduced automatically from a favorites list
406 of a particular profile such as profile A 404. For example,
the favorites list 406 of profile A 404 may contain Experts on
X, Best Bets on X, Favorite Website on Y, or any other favorite
topic from any context. In this embodiment, news display 412
presents information on News on X or News on Y. In another,
the news display 412 removes duplicate entries. In one
embodiment, news display 412 presents similar information
based on the favorites list 406 of profile B 402. This informa-
tion may be presented in news display 412 together with or
separate from information originating from profile A 404.

[2023] In yet another embodiment, the invention accepts
custom requests for news information from a user under a
profile such as profile A 404 at block 408. The custom
requests for news information at block 408 may also be
accepted under different profiles such as profile B 402. In one
embodiment, news display 414 presents news information to
the user based on specific requests 408. News display 414 may
therefore present news information for special requests 408
for a single profile or multiple profiles. Furthermore, news
display 414 may segregate news information presented based
on the originating profile that submitted the special request
408.

[2024] In yet another embodiment, news display 416 pre-
SENTs news information based on the current information 410.
The current information 410 generally refers to the informa-
tion that a user is currently viewing. In one embodiment, the
NEWS display 416 will not present duplicative information that
is already accessible by the user or presented to the user. News
displays 412 and 414 may also be adapted to remove duplica-
tive information.

[2025] In a further embodiment, news displays 412, 414, or
416 present breaking news, headlines, and/or newsmakers
information for each topic. For example, in this embodiment,
news display 412 is based on the favorites list 406 from profile
A 404, which contains a link to experts on X, and may present
breaking news on X, headlines on X, and/or newsmakers on
X. This could be true for every topic, from every profile, and
under any news display 412, 414, and 416.

[2026] In an alternative embodiment, the news displays
412, 414, or 416 may be static, dynamic, animated, or scroll-
able. Furthermore, the news displays 412, 414, or 416 may
be presented together or separate on a portion of the display
screen, on the entire display screen, or on multiple display
screens.

[2027] FIG. 72 is a block diagram for a method of present-
ing knowledge community statistics at a client user interface,
in accordance with an embodiment of the invention. In one
embodiment, a client invokes a request for statistics on one or
more knowledge communities at block 4102. The request is
brokered by an information server at block 4104. The
information server requests statistics from one or more knowledge
communities at block 4106. The statistics are returned
directly to the client at block 4102 or through the information
server at block 4104. In another embodiment, the statistics
include results count per context-template. Alternately, any
statistics on any data from any part of the invention may be
presented.

[2028] FIG. 73 is a block diagram of a method for allowing
users to remove duplicative presented information, in accor-
dance with an embodiment of the invention. In one embodiment, duplicative information is presented to the user and noticed by the user at block 702. The user manifests an intent to delete the duplicative information at block 704 by triggering a command. The command may invoke a deletion service at block 706 thereby removing the duplicative entry at block 708.

[2929] FIGS. 74-1 and 74-2 illustrate a documents table data and index model, in accordance with an embodiment of the invention. In one embodiment, the documents table includes the fields listed under the column name 802. One or more of the fields and/or the field names under the column name 802 may be changed, added, and/or removed and still be within the teachings of this invention. Preferably, each field listed under column name 802 may have a corresponding data type listed in the data type column 804. The examples provided in the data type column 804 may be deviated from and still be within the scope of this invention. Each field listed under column name 802 may have a corresponding data type listed in the data type column 804. The examples provided in the data type column 804 may be deviated from and still be within the scope of this invention. Each field listed under column name 802 may be indexed as indicated in the indexed column 806. However, other fields listed under column name 802 may be indexed and fields shown as indexed in the indexed column 806 may be non-indexed.

[2930] In another embodiment, the SourceUri field is a unique constraint. In yet another embodiment, the Bet-Strength field indicates the aggregate semantic strength of the document. In a further embodiment, the NumConcepts field indicates the number of concepts in the document. In yet a further embodiment, the BestBetHint field indicates whether a particular object is a best bet as indicated by the semantic inference engine previously disclosed in applicant's prior applications, referenced above. In an alternative embodiment, the recommendationHint field indicates whether a particular object is a recommendation as indicated by the semantic inference engine. In one embodiment, the default for this field is two-thirds of the best bet semantic strength value. In another embodiment, the BreakingNewsHint indicates whether a particular object is breaking news as indicated by the time sensitive inference engine previously disclosed in prior applications. In a further embodiment, the HeadlinesHint field indicates whether a particular object is breaking news as indicated by the time sensitive inference engine. In yet a further embodiment, the BetRankHint field represents the score of a particular object's semantic strength. In an alternative embodiment, the RichMetadataHint field indicates whether a particular object originated from a rich metadata source. In another embodiment, the SemanticHash field represents a hash of the body of a particular document object to enable duplication detection. For example, the hash may include the key phrases of a document in alphabetical order.

[2931] FIG. 75 is an objects table data and index model, in accordance with an embodiment of the invention. In one embodiment, the objects table includes the fields listed under the column name 902 column. The field names under the column name 902 may be changed, added, or removed and still be within the teachings of this invention. Preferably, each field listed under column name 902 will have a corresponding data type listed in the data type column 904. The examples provided in the data type column 904 may be deviated from and still be within the scope of this invention. Each field listed under column name 902 may be indexed as indicated in the indexed column 906. However, other fields listed under column name 902 may be indexed and fields shown as indexed in the indexed column 906 may be non-indexed.

[2932] FIG. 76 is a semantic links table data and index model, in accordance with an embodiment of the invention. In one embodiment, the semantic links table includes the fields listed under the column name 1002. The field names under the column name 1002 may be changed, added, or removed and still be within the teachings of this invention. Each field listed under column name 1002 may have a corresponding data type listed in the data type column 1004. The examples provided in the data type column 1004 may be deviated from and still be within the scope of this invention. Each field listed under column name 1002 may be indexed as indicated in the indexed column 1006. However, other fields listed under column name 1002 may be indexed and fields shown as indexed in the indexed column 1006 may be non-indexed.

[2933] In another embodiment, the BestBetHint field represents the best bet context predicate as supplied by the semantic inference engine. In another embodiment, the RecommendationHint field represents the context predicate as supplied by the semantic inference engine. Additionally, its default value may be two-thirds (or any other fraction in alternate embodiments) of the best bet semantic strength value. In a further embodiment, the BreakingNewsHint field represents the breaking news context predicate as supplied by the time sensitive inference engine. In an alternative embodiment, the HeadlinesHint field represents the headlines context predicate as supplied by the time sensitive inference engine. In yet another embodiment, the BetRankHint field represents the score of the semantic strength of a particular object.

[2934] FIG. 77 is a composite index table model, in accordance with an embodiment of the invention. In one embodiment, the composite index table includes the fields listed under the column name 1102. The field names under the column name 1102 may be changed, added, or removed and still be within the teachings of this invention. Each field listed under column name 1102 may have a corresponding data type listed in the data type column 1104. The examples provided in the data type column 1104 may be deviated from and still be within the scope of this invention. Each field listed under column name 1102 may be indexed as indicated in the indexed column 1106. However, other fields listed under column name 1102 may be indexed and fields shown as indexed in the indexed column 1106 may be non-indexed.

[2935] FIG. 78 is a block diagram for a method of quickly indexing data contained in a metadata feed, in accordance with an embodiment of the invention. In one embodiment of the invention, a metadata processor 1204 accepts an incoming metadata feed 1202 that contains individual informational items. The metadata feed 1202 may be an RSS feed. The metadata processor 1204 then queries a database 1206 to determine whether the metadata feed 1202 had been previously processed. In one embodiment, the metadata feed 1202 is identifiable and stored in the database 1206 by its URI. However, the metadata feed could be identified and stored using a different identifier. If the query indicates that the metadata feed 1202 had been previously processed, the metadata processor 1204 skips the metadata feed 1202 in its entirety at block 1208. However, if the query indicates that the metadata feed had not been previously processed, the metadata processor 1204 then parses the individual items of the metadata feed 1202 and records the information at block 1210. The metadata processor 1204 then updates the database 1206 to indicate that the metadata feed 1202 has been processed.
FIG. 79 is a block diagram for a method of adjusting threshold values that are used to determine the most relevant objects in a given context, in accordance with an embodiment of the invention. In one embodiment, objects at block 1302 are collected (e.g., documents). Semantic strength values are assigned to each of these objects for a given context at block 1304 by the semantic inference engine discussed in prior applications. Thus, at block 1304 there is a collection of objects with associated semantic strength values. The objects with the highest semantic strength values are marked as best bets at block 1306 if their value exceeds a given threshold value. In one embodiment, the threshold value may be all documents greater than 90% of the value of the highest ranked document. Thus, in this embodiment, the threshold value is a relative value. This value could be adjusted, or it could be absolute, or relative to any other metric, or combinations of metrics as desired. As additional objects are then added and collected at block 1302, they are also assigned semantic strength values at block 1304. The added objects may render the old threshold value obsolete given that some of the newly added objects may possess a higher semantic strength value than the highest previous semantic strength value. Thus, the addition of new objects with semantic strength values trigger an adjuster at block 1308. However, the adjuster 1308 could be set to run on a periodic timer or manually triggered. The adjuster at block 1308 determines the new highest semantic value of a given set of objects and adjusts the threshold value to be used at block 1306 accordingly. Furthermore, the adjuster updates other threshold values at block 1310, including values in multiple tables or databases. In one embodiment, recommendations are objects that have a semantic strength value above another threshold value calculated from the best bets threshold value. Thus, in this embodiment, the adjuster adjusts the recommendations threshold value at block 1310 as the underlying best best threshold value changed. In a further embodiment of the invention, the adjuster at block 1308 operates when the total number of best bets exceed a given percentage of total objects. In one embodiment, this percentage is 1%.

FIG. 80 is a method for indexing and retrieving semantically relevant documents, in accordance with an embodiment of the invention. In one embodiment, a full document at block 1402 is paginated into individual page documents at block 1404. A full document may be parsed according to sections, chapters, the alphabet, or any other similar or different methodology, or any combination of methods. In another embodiment, the paginated documents at block 1404 is semantically indexed at block 1406. Accordingly, a single document may be subdivided into many subparts whereby one or more, or preferably all of each of these subparts is semantically indexed. In a further embodiment, a client semantically searches and retrieves only the paginated subparts of a document at block 1408. Alternatively, a client semantically searches and retrieves, either separately or in combination, the original full document at block 1408. In this manner, a client is presented only the semantically relevant portions of particular document at block 1408. In an additional embodiment, each paginated subpart document has a link that presents the original document from which the paginated document originated at block 1410. In one embodiment, this link is a hyperlink.

In yet a further embodiment, incoming documents or other information are also submitted for content transformation at block 1412. Examples of content transformation include converting images to text data, language translation, or content cleansing by removing advertisements or other information. In one embodiment the image to text conversion is achieved using Optical Character Recognition (OCR). Accordingly, an image may be converted to text data, an English essay may be converted to French, or advertisements may be removed from a newspaper article. In another embodiment, the content transformation may be linked together. Accordingly image data may be converted to English text data that may then be converted to French whereby advertisements may be removed. The foregoing examples of content transformation may be expanded to cover any other form of content transformation. The content transformation may occur before, after, in addition to, or in lieu of the process documents into subparts at block 1404. In one embodiment, the content transformation at block 1412 occurs prior to the parsing of the document into subparts at block 1404. Accordingly, in one embodiment a full document, subparts of a document, content transformed full documents, or content transformed subparts of a document are separately semantically indexed. Each of these materials may be searched and displayed independently or in combination on the client at block 1408. Additionally, each of these materials may include a link 1410 to any other related document, including a link to the original full document. In yet another embodiment, the transformations result in a metadata feed (e.g., an RSS feed) that is appropriately interpreted by the semantic indexing system at block 1406.

In one embodiment the invention is a method for highlighting semantically relevant keywords in displayed documents resulting from semantic searches, in accordance with an embodiment of the invention. In one embodiment, the client semantic runtime 1502 caches an ontology 1508 copy on the client computer for each knowledge community 1506 that a client user interface 1504 subscribes to. The copy may be an XML document representative of the data in an ontology that may be parsed using XPATH. The copy may also be a set of hash tables that include the terms of an ontology. Also, the copy may be stored to a client computer disk and lazily cached to memory only when necessary. In another embodiment, the server 1510 replicates the procedures of the client semantic runtime 1502 and stores a copy of each ontology 1508 for each knowledge community 1506. In yet another embodiment, the client semantic runtime 1502 downloads a copy of an ontology by requesting the locations of the ontologies 1508 directly from a knowledge community 1506. The ontology 1508 is downloaded via a communication protocol such as HTTP by invoking a dynamically constructed URI that points to the location of the ontology 1508 data. The client semantic runtime 1502 creates the URI by extracting concepts; passing the concepts to the knowledge community 1506 or server 1510; and obtaining URIs of relevant ontologies 1508. In a further embodiment where ontologies 1508 are not directly accessible from a client semantic runtime 1502 a copy of the ontology 1508 is obtained directly from a knowledge community 1506 or a server 1510. In a further embodiment, the client semantic runtime 1502 accepts input (e.g., via an exposed API) from the user interface 1504, searches the cached ontology data based on the input, and returns output containing relevant terms. The input may be comprised of the source URI of the information displayed in the user interface 1504 and the semantic search terms used to generate the displayed information. The output may consist of relevant terms that may be highlighted in the information.
displayed in the user interface 1504. These relevant terms may be those words, categories, or other items of information that are semantically relevant to the semantic search that was responsible for the information displayed in the user interface 1504. The relevant terms may be returned as part of an XML document. The XML document may also contain additional data to describe the terms or for other purposes. Additionally, the relevant terms may include the semantic search terms submitted at the client user interface 1504. In one embodiment, the search for relevant terms by the client semantic runtime 1502 is independent of the search for semantic relevant documents conducted by the server 1510. Alternatively, the search for relevant terms by the client semantic runtime 1502 may be conducted by acquiring the key concepts of the information displayed in the user interface 1504; determining the key concepts that match a user's semantic search request; and searching the cached ontology based thereon. Accordingly, in one embodiment when a semantic query or request is launched in the user interface 1504, the client semantic runtime 1502 may use this request to return relevant terms and highlight the terms in the displayed document in the user interface 1504. In yet a further embodiment, the client semantic runtime 1502 ontology cache data is updated periodically as information in the original ontology 1508 changes. This update may occur as the user subscribes to a new ontology, unsubscribes from an ontology, or when an ontology indicates that it has been updated. Other cache copies of ontologies 1508, such as those on the server 1510 or knowledge community 1506, may also be similarly updated if necessary. In additional embodiments, other updates to the client semantic runtime 1502 may include updates to the search tools (e.g., XpathDocuments, XMLTextReaders, ontology 1508 file sizes, ontology 1508 last modified file time, or file paths). In another embodiment, the client semantic runtime manages memory usage for large ontology copies by only caching copies if there is available memory. (e.g., if the copy is larger than 16 MB the available memory must be greater than 512 MB or if the copy is between 8 MB and 16 MB the available memory must be greater than 256 MB or if the copy is less than 8 MB the available memory must be greater than 128 MB).

[2940] In one embodiment, the information server used to catalog semantically marked up documents uses parallel indexing and I/O, rather than serialized indexing and I/O, so that the information server is able to index some documents while prevented from indexing other documents.

[2941] In another embodiment, the information server used to catalog semantically marked up documents removes redundant or unused indexes.

[2942] In yet another embodiment, the information server used to catalog and retrieve semantically marked up documents folds all calls to a single knowledge domain for multiple ontologies into a single call.

[2943] FIG. 82 is a block diagram showing methods for creating and managing multiple types of knowledge communities, in accordance with an embodiment of the invention. In one embodiment, client 1702 is in communication with server 1704. Server 1704 is in communication with multiple knowledge communities 1706, 1708, 1710. Standard knowledge community 1706 contains ontology data. Mirrored knowledge community 1708 also contains ontology data. However, in this embodiment, the ontology data is merely a copy of ontology data originating from the actual knowledge community 1712. In this embodiment, the updates to the copy may be periodic, automatic, or manual (e.g., every minute, hour, day, week, or never). Accordingly, a division of labor is achieved between certain knowledge communities (e.g., knowledge communities dedicated to indexing). Virtual knowledge community 1710 may not contain ontology data. Instead, virtual knowledge community 1710 redirects communications between the server 1704 and the actual knowledge community 1714. The communication brokering between the virtual knowledge community 1710 and the actual knowledge community 1714 is transparent to the client 1702. In an alternative embodiment, actual knowledge communities such as 1712 or 1714 are invisible to the client 1702.

[2944] FIG. 83 is a block diagram of a method for providing user feedback on the available knowledge communities, in accordance with an embodiment of the invention. In this embodiment, a user makes a semantic search request involving certain knowledge communities at block 1902. The search request is made via a free-text entry at block 1904 or via a menu selection at block 1906. If the user enters a text request for a knowledge community at block 1904, the system compares the input knowledge community request with the available knowledge communities at block 1908. If there is at least one matching knowledge community, the system displays the desired search results at block 1910. If there is not at least one available knowledge community at block 1908, the invention displays an error message at block 1912. Alternatively, if the user makes a selection of a knowledge community from a supplied selection (e.g., a menu), the system simply displays the results without verifying that the knowledge community is available at block 1910. However, the system could also check for the availability of the selection at block 1908.

[2945] In another embodiment, the error messages are displayed in a field. In yet another embodiment, the error messages are displayed using an icon. In a further embodiment, different messages or icons are presented depending upon whether the search request was at least partially successful. In an alternative embodiment, the error message is expanded to display details on the error.

[2946] FIG. 84 illustrates a method of using semantic sounds to notify a user regarding the arrival of news in accordance with an embodiment of the invention. In this embodiment, news content is delivered to a client computer at block 2102. The semantic sound generator analyzes this incoming news to determine the content at block 2104. The semantic sound generator then produces audible sound that is tailored to the incoming news content and is intelligently based on the semantics of the news content at block 2106.
of categories and community lists of categories. Accordingly, the personal lists may be unavailable to other users and the community lists may be available to other users. The personal lists may be further divided into a default list 2202, a favorites list 2204, a live list 2206, or a my documents list 2208. Various types of division or naming schemes may be fashioned. In this embodiment, the default list 2202 may include those categories specifically requested by a client user. The favorites list 2204 may include those categories that relate to a client-user's favorite list. The live list 2206 may include the categories embodied in other lists to which they dynamically update. The my documents list 2208 may include those categories that relate to a client user's local information (e.g., local file names, email messages, web browser favorites, or any other specified source). In one embodiment, the my documents list 2208 is established through the use of local crawling agents that periodically search local information and update the categories in the my documents list 2208 based thereon. Community lists may be provided to a client user as suggestions that the client user may choose to follow. Accordingly, an information server may mine certain categories from each knowledge community and present these categories to a client user in the context of one or more profiles. These categories may be dynamically updated. The categories in the community lists may be divided into recommended categories 2210, popular categories 2212, categories in the news 2214, or best bets categories 2216. Various types of division or naming schemes may be fashioned. Also, a client user may still search all categories. Recommended categories 2210 may include those categories that are similar to those already used by a client user. Additionally, recommended categories may include those categories that are used by other client users with similar interests. Popular categories 2212 may include those categories that are most accessed within a given knowledge community. Categories in the news 2214 may include those categories that are currently in the news. Best bet categories 2216 may include those categories that correspond to the best bets within a given knowledge community.

In another embodiment, the category lists are organized in a deep information format that include expandable and retractable nodes such as profile, category list, ontology, parent category, and category. Other forms of organization may be employed. Accordingly, a user may be able to navigate between multiple nodes. In yet another embodiment, these nodes may be dragged, dropped, copied, pasted, or used with the smart lens previously disclosed.

In a further embodiment, the deep information form is applied to the contents of an entity (e.g., a meeting entity). As an example, a meeting entity may have as its contents the participants of the meeting, the topics that were discussed during the meeting, the documents that were handed out during the meeting, or any other similar contents. Accordingly, in this embodiment a user may navigate within an entity or from an entity.

FIG. 86 is a block diagram of a method of semantically indexing and retrieving non-text data, in accordance with an embodiment of the invention. In one embodiment, non-alphabetical text data is annotated with text at block 2302. The annotations are then separated from the document and linked (e.g., via hyperlink) back to the originating document at block 2304. The annotations are then semantically indexed themselves at block 2306. A client user executes a semantic search at block 2308. The results are be interpreted by the user at the same block 2308. When a client user desires to locate the originating data from which the annotation result arose, the client user follows the link to the originating non-alphabetical text data document. In an alternative embodiment, the non-alphabetical text data is numerical, audio, video data, or any other similar data. In yet another embodiment, the non-alphabetical text data is a business report containing sales numbers, financial projections, or other similar data.

FIG. 87 is a block diagram of a method for providing ontology feedback in accordance with an embodiment of the invention. In this embodiment, a client user interacts with ontology data at block 2406. The client user then invokes a feedback request (e.g., an email form, chat room, or other communication method) to the ontology support personnel at block 2404. The ontology support personnel interprets this feedback request and makes any necessary changes to the appropriate ontology data at block 2406. In an alternative embodiment, the request information automatically populates the address, ontology name, ontology identifier, problem statement, or any other relevant field. In an alternative embodiment, a privacy statement is provided to the client user.

FIG. 88 is a block diagram of a method for advanced semantic searching in accordance with an embodiment of the invention. In this embodiment, a client user requests a topic one 2502 from a database one 2504 that is related to a topic two 2506 from database two 2508. For example, a client user may request all proteins from a protein database that are relevant to abstracts on a particular inhibitor molecule found in a medical database. Accordingly, a client user may link together two or more semantic searches. In an alternative embodiment, a client user instigates an advanced search by moving images representing a topic over another image representing an information source, database, a category, or context.

FIG. 89 is a block diagram of a method for handling floating text in an RSS feed. In an exemplary embodiment, the text information without a stored file (e.g., a document) can be gathered by the DSA or other similar service at block 2604. The text information without a stored file at block 2602 may be floating text or a result of an inability to index an associated file (e.g., a website may forbid crawlers from indexing website documents). The DSA then generates an RSS or other metadata feed at block 2606 with a namespace qualified tag that indicates the absence of a stored file. In one embodiment, the term "raw" may be used. Because of this tag, the information server and its processes may be on notice at block 2608 that the metadata does not have a stored file. Accordingly, this method may allow metadata to be indexed even if there is no associated file or the document is unable to be indexed.

FIG. 90 is a block diagram of a method for extracting a semantic query from an image, in accordance with an embodiment of the invention. In this embodiment, an image 2802 is placed on a clipboard or other similar receptacle at block 2804. The semantic query may be created based upon the concepts that are extracted from the image at block 2806. The semantic query is submitted to the information server at block 2810. In an alternative embodiment, the data in the clipboard is any data object. In yet another alternative embodiment, the image is of a chemical compound. In this embodiment, scientific researches drag an image of a chemical compound into a clipboard whereby a semantic query is created based thereon.
FIG. 91 is a block diagram for a method for improving ontology development in accordance with an embodiment of the invention. In this embodiment, a word is inputted into the system at block 2902. The word and its appropriate meaning are added to an ontology at block 2908. However, the word may also be subject to algorithms at block 2904. These algorithms reduce the word to its roots or correct misspelling errors. The results of the algorithm are then subjected to a synonym suggestion tool at block 2906. The word results of the synonym suggestion tool along with their associated meanings are added to an ontology at block 2908. This is demonstrated by reference to various alternative embodiments. In one embodiment, a public synonym suggestion API is utilized. In different embodiment, the synonym suggestion tool suggests slang words. In a different embodiment, the synonym suggestion tool suggests words that begin with the input phrase, contain the input phrase, or end with the input phrase. In a different embodiment, the suggestions are prioritized by any desired methodology. In a different embodiment, the root algorithm includes the following steps: call the synonym suggestion tool with the exact phrase, remove one letter, call the synonym suggestion tool with the truncated phrase, and repeat. In a different embodiment, the misspelling algorithm includes the following steps: submit the exact phrase to the suggestion tool, remove one vowel, submit the altered phrase to the suggestion tool, remove another vowel, and repeat. Alternatively, the misspelling algorithm may remove one of each double letter instance in the word and submit it to the suggestion tool. Alternatively, the misspelling algorithm may remove hyphens or add hyphens and submit the altered phrase to the suggestion tool. In yet a different embodiment, the algorithm corrects the word based on a pre-developed word list.

FIG. 92 is a block diagram of a method for developing and maintaining ontologies, in accordance with an embodiment of the invention. In this embodiment, a cross-ontology validation application 3008 is in communication with ontology one 3002, ontology two 3004, ontology three 3006, or ontology four 3008. The validation application 3008 is in communication with more or less than four ontologies. In an alternative embodiment, the cross-ontology validation application 3008 assists in developing and maintaining ontologies. For example, the cross-ontology validation application 3008 may determine whether there are discrepancies in naming schemes between multiple ontologies and notify an ontology administrator (e.g., artificial intelligence sub-categories may be different in the IT and Products and Services ontologies. In another example, the cross-ontology validation application 3008 suggests the hooks in one domain to be exclusions for another domain and vice versa (e.g., viruses in a health database should have exclusions that are themselves hooks for virus in an IT database). In an alternative embodiment, the cross-ontology validation application considers that multiple-word forms include the same exclusions or hooks.

In an alternative embodiment of the invention, the time-sensitive semantic interface engine (TSIE) is designed to return ranked newsworthly information from the recommendations based on context, time, and semantic strength.

In a different embodiment, the semantic interface engine (SIE) returns the semantic strength for a document or other similar container of information to a particular category, it’s parent category, or its child categories (e.g., the semantic strength of a document to encryption may also be assigned to security as a parent of encryption). In yet another embodiment, the parent-child assignments of semantic strength are attenuated as necessary.

FIG. 93 is a block diagram for a method for semantic question answering in accordance with an embodiment of the invention. In this embodiment, the client user enters a question at block 3104. The question is passed to the information server at block 3106. The information server returns a document or documents that semantically answer the question at block 3108; alternatively, the information server may return an annotation or annotations that semantically answer the question at block 3110. In a different embodiment, the annotations have links (e.g., hyperlinks) back to the originating document. Accordingly, the user uses the link when viewing the annotation to obtain the full document that the annotation was based upon. In another embodiment, the annotations are annotated at block 3112 and semantically indexed to be available for retrieval at block 3110. For example, a question of the population of Norway may result in the generation of a document that describes the population of Norway somewhere in its contents. In another example, a question of the number of people that live in the second largest Scandinavian country may result in the generation of an annotation provides the answer with a link back to the originating document.

FIG. 94 is a block diagram of a method of coupling natural language with semantic language queries in accordance with an embodiment of the invention. In one embodiment, a client user inputs a natural language query at block 3204. The natural language query is then broken down into key phrases, words, or variants at block 3206. The key phrases, words, or variants are then submitted to be compared with available ontology categories at block 3208. Based on this comparison, the system presents the user with recommended search terms at block 3210. The client user may select, remove, or add to the recommended search terms at block 3202. After review, the final semantic query is then selected at block 3212 and submitted to the information server 3214 for semantic query results. Accordingly, the client user may use natural language queries to begin the process of semantic searches. In another embodiment, a client installed plug-in maps the natural language input to semantic input before passing the query to the server for interpretation; however, this may be accomplished remotely from the client. In a further embodiment, the mapped semantic input is not reviewed by the client user before being submitted to the information server. As an example, the natural language query “develop a genetic strategy to deplete or incapacitate a disease-transmitting insect population” may result in, “diseases or disorders from a medical database and insects from a medical database and ‘transmit or transmits or transmission or transmission or transmitting.’”

Certain embodiments of Live Mode were disclosed in one or more of applicant’s prior applications listed above and are incorporated by reference herein. In one embodiment, when a Request Collection is in Live Mode some or all of its requests and entities may be presented live when the request collection is viewed. In another embodiment, the request and entities are not automatically made live themselves if they are already live. In this embodiment, only when the request collection is displayed are the requests viewed live. In yet another embodiment, a skin elects to merge the results of a Request Collection so that only one set of live results is
displayed. However, in other embodiments the skins can elect to keep the individual request collection entries viewed separately in Live Mode.

[2963] FIG. 95 is a block diagram of a method for categorizing extracted concepts from a URI, in accordance with an embodiment of the invention. In one embodiment, the ontology 3308 and the concept categorizer 3304 share the same lexicon 3306. In this regard, the information from a URI 3302 is categorized in an information server 3310 based upon lexicon 3306. Alternately, the lexicon is unique to the categorizer. In a further embodiment, when the categorizer 3304 is interpreting semantic context with non-semantic context templates (e.g., all bets, random bets) or with non-semantic ranking (e.g., bucket #0), it may map the URI information 3302 to searchable keywords. Accordingly, in this embodiment when categorization fails the URI is still retrievable via a keyword match.

[2964] FIG. 96 is a block diagram of a method for establishing context queries, in accordance with an embodiment of the invention. In one embodiment the concepts are extracted from a data source (e.g., a document) at block 3402 and submitted to a server 3404. The server then contacts multiple knowledge communities 3406, 3408, or 3410 whereby the knowledge communities categorize and return weighted values for the extracted concepts. The number of knowledge communities may be more or less. The server then maps the returned category weight values to context templates at block 3412 (e.g., best bets, recommendations, all bets, etc.). Rules are then be created to query the context templates at block 3414 and these rules are then associated with a context template at block 3416.

[2965] In another embodiment, the concepts are passed directly, rather than through the server, to the knowledge community to be categorized and weighted. In yet another embodiment, the client has a concept extraction cache to prevent multiple concept extractions of the same data source. In a further embodiment, the server has a concept-to-category cache to prevent multiple category and weight determinations of the same concept. In one embodiment these caches are purged periodically. In another embodiment, the server cache utilizes a file access lock to prevent concurrent connection errors. Examples of query rules created at block 3414 may include, but are not limited by, the following. First, for each best bet category in the source, create a query with an "and" of all the categories. Second, for each recommendation category in the source that is not a best bet, create a query with an "and" of all the categories. Third, if first query had more than one category create N queries with each category for each best bet category in the source. Fourth, if the second query had more than one category create N queries with each category for each recommendation category in the source. Fifth, for each best bet category in the source forward-chain by one up the hierarchy in the ontology corresponding to the category and create a query with an "and" of the parent categories (e.g., if there was a best bet on encryption then forward-chain to the parent Security in the same ontology and "and" that with the other best bets as well as check for and elide or eliminate duplicates as necessary when best bet categories share the same parent). In a further embodiment, forward-chaining is invoked if there are multiple unique parents. In an alternative embodiment, the threshold is increased to two for best bets. Sixth, for each recommendation category in the source that is not a best bet category apply the equivalent of query five. In one embodiment, the semantic distance threshold for forward-chaining with recommendations is 1. Seventh, for each all bets category in the source that is not a best bet or a recommendation create a query with an "and" of all the categories only if there are eventually multiple unique categories. Eight, if the source has less than a given number of keywords then add a keyword search query. In alternate embodiments, one or more of the foregoing list may be omitted, and the sequence may vary.

[2966] In one embodiment, the ontologies in the knowledge communities are also annotated with hints that indicate how the server should forward-chain to parents.

[2967] FIG. 97 is a block diagram of a method for extracting concepts from disparate sources, in accordance with an embodiment of the invention. In one embodiment, a server passes the URI of an object to a client at block 3502. The client communicates with the object located at the URI at block 3506 to obtain the metadata of the object. The concepts are extracted from the aggregate URI and object metadata at block 3508 and semantically processed at block 3510. In an alternative embodiment, the client passes the URI to an independent service that may itself gather metadata from the object located at the URI and return the object metadata to the client.

[2968] In another embodiment, the object referenced by a URI is XML. In yet another embodiment, the XML is in the SRML xml schema format. In a further embodiment of the independent service, the URI to the service is configured at the server or the client.

[2969] FIG. 98 is a block diagram of a method for reorganizing independent website data according to semantic strength, in accordance with an embodiment of the invention. In one embodiment, a user selects a profile at block 3602 and utilizes a client web browser at block 3604. The client web browser displays the content of an independent web page at block 3606. The content of the web page, including the links on the web page, are transmitted to the information server at block 3608. The information server queries at least one knowledge community at block 3610 to semantically rank the information from the independent website. The query results are returned to the client web browser at block 3604 whereby the independent webpage is reorganized, altered, or annotated with the semantic strength rankings of the knowledge community. Accordingly, in this embodiment of the invention web pages are dynamically reorganized or altered based on the semantic strength of their content to assist the user in more intelligently browsing.

[2970] In another embodiment, the knowledge community returns data in XML format that indicates whether an object is a best bet or recommendation. In another embodiment, the independent web page is annotated with the semantic ranking information (e.g., different colors, balloons, pop-ups, etc.).

[2971] FIG. 99 is a block diagram of a method for semantic analysis on the client, in accordance with an embodiment of the invention. In one embodiment, the semantic analysis 3706 of an object 3702 is performed on the server 3710. In another embodiment, the identical semantic analysis 3704 of an object 3702 is performed on the client 3708.

[2972] FIG. 100 is a block diagram for a method of generating information on experts, interest groups, or newsmakers, in accordance with an embodiment of the invention. In one embodiment, the experts 3802 are generated by selecting the best bets on people 3814. In another embodiment, interest groups 3804 are generated by selecting the recommenda-
tions 3810 on people 3814. In yet another embodiment, newsmakers 3806 are generated by selecting the headlines 3812 on people 3814.

[2973] FIG. 101 is a method for adding new ontologies to a client semantic browser, in accordance with an embodiment of the invention. In one embodiment, an add-in file 3904a is added to a client semantic browser 3906. The add-in file 3904 references a new ontology 3902 that is then cached in the client semantic browser 3906. In another embodiment, the add-in file 3904 is an XML file. The XML file may contain the following fields: DomainID, KnowledgeDomain, PublisherName, Creator, CategoryFolderDescription, AreasOfInterest, TaxonomyUri, Version, or Language. In yet another embodiment, the downloaded ontology data is registered as an available knowledge source. Accordingly, new ontologies are dynamically installed or uninstall.

[2974] In a further embodiment, the client semantic browser 3906 periodically polls a client user profile’s subscribed knowledge communities to determine whether there are subscribed ontologies that are not locally installed. In an alternative embodiment, the semantic client browser 3906 alerts the user when such ontologies exist. In one embodiment, a user selects an ontology for installation.

[2975] FIG. 102 illustrates a method for using field and category specific searches to supplement keyword searches, in accordance with an embodiment of the invention. In one embodiment, a client user enters a field specific keyword search at block 4002 (e.g., Author: “long BMI”). PubYear: 2003, PubYear: 2003-2005, etc.). This field specific keyword search is considered by the query processor at block 4006 whereby the input values are mapped to the appropriate query format and output at block 4008 (e.g., PREDICATE-TYPEID_AUTHOREDBY, PREDICATE-TYPEID_PUBLISHEDINYEAR). In another embodiment, a client user enters a category specific keyword search at block 4004 (e.g., Cancer: “Tyrosine Kinase Inhibitor”). The category specific keyword search may be considered by the query processor at block 4006 whereby the input values are mapped to the appropriate query format and output at block 4008.

[2976] In another embodiment, a client user specifies multiple fields or categories in the keyword search (e.g., *:Apoptosis may be to all categories). In yet another embodiment, the fields or category specific fields are combined using Boolean logic (e.g., PubYear: 1970-1975 OR PubYear: 1980-1985 OR Cancer: Tyrosine Kinase Inhibitor). (See, also, FIGS. 5 and 6 and corresponding description above).

[2977] FIG. 103 is a method for creating weighted indices and searching thereon, in accordance with an embodiment of the invention. In one embodiment, an object is gathered at block 4302 and submitted to the information server at block 4304 whereby the information server assigns a weighted index to the object that indicates the strength of the relationship between the object and a particular category. A client user at block 4310 selects an information type at block 4308 (e.g., best bet, recommendations, etc.). The information type is then mapped to the appropriate query at block 4306 to retrieve the desired objects from the information server.

[2978] In another embodiment, the weighted index range is between zero and nine. In yet another embodiment, the queries at block 4306 include those that retrieve objects with the following weighted indexes: 0-10, 1, 2, 3, 4, 5, 6-10, 7, 8, 9-10. In an alternative embodiment, the information types at block 4308 may be all bets, best bets, recommendations, breaking news, headlines, or random bets. In one embodiment, the information types are mapped to the queries at block 4306 according to the following rules: all bets are index weights 0-10, best bets are index weights 9-10, recommendations are index weights 6-10, breaking news are index weights 6-10, headlines are index weights 6-10, and random bets are 0-10. The information types and the associated index weights that they are mapped to retrieve may be altered or configured by an administrator. In one embodiment, the information types are segregated into ranking groups. For example, ranking group 0 may include only all bets; ranking group 1 may include all bets and recommendations; ranking group 2 may include best bets, recommendations, and all bets; and ranking group 3 may include all information types. In another embodiment, random bets are implemented within ranking groups. Also, it should be understood that additional ranking groups may be added and the example ranking groups may be removed or altered. In a further embodiment, the returned objects within an information type are further ranked according to the weighted index, time, or they may be randomly returned. In one embodiment, the returned object results are checked for duplicates. In another embodiment, the objects in the information types are updated because the weighted index assigned to objects is a relative value.

[2979] Turning now to another embodiment, a referenced end-to-end system and/or resulting knowledge medium, which may be regarded and/or referred to as an Information Nervous System, addresses the problems described herein. An embodiment of the system provides intelligent and/or dynamic semantic indexing and/or ranking of information (without requiring formal semantic markup), along with a semantic user interface that provides end-users with the flexibility of natural-language queries (without the limitations thereof), without sacrificing ease-of-use, and/or which also empowers users with dynamic knowledge retrieval, capture, sharing, federation, presentation and/or discovery—for cases where the user might not know what she doesn’t know and/or wouldn’t know to ask.

[2980] A system according to an embodiment of the invention understands what it indexes, empowers users to be able to flexibly express their intent simply yet precisely, and/or interprets that intent accurately yet quickly. A system according to an embodiment of the invention blends multiple axes for retrieval, capture, discovery, annotations, and/or presentation into a unified medium that is powerful yet easy to use.

[2981] A system according to an embodiment of the invention provides end-to-end functionality for semantic knowledge retrieval, capture, discovery, sharing, management, delivery, and/or presentation. The description herein includes the philosophical underpinnings of an embodiment of the invention, a problem formulation, a high-level end-to-end architecture, and/or a semantic indexing model. Also included, according to an embodiment of the invention, is a system’s semantic user interface, its Dynamic Linking technology, its semantic query processor, its semantic and/or context-sensitive ranking model, its support for personalized context, and/or its support for semantic knowledge sharing all of which an embodiment employs to provide a semantic user experience and/or a medium for knowledge.

[2982] Further described herein are an overview of the difference between knowledge and/or information and/or how that should apply to an intelligent information retrieval system; the problem with Search, as is currently defined and/or implemented by current search engines; context and/or semantics especially on the limitations of current search
engines and/or retrieval paradigms and/or the implications on the design of an intelligent information retrieval system; the Semantic Web and/or Metadata and/or describes how these initiatives relate to the design of an intelligent information retrieval system and/or also how they may be placed in perspective from a practical standpoint; the problems and/or limitations of current search interfaces; Semantic Indexing in general, how this relates to an intelligent information retrieval system, and/or on Dynamic Semantic Indexing as designed and/or implemented in the Information Nervous System, in accordance with at least one embodiment of the invention.

[2983] Intelligent Retrieval: Knowledge vs. Information. An intelligent information retrieval system, according to an embodiment of the invention, simulates a human reference librarian or research assistant. A reference librarian is able to understand and/or interpret user intent and/or context and/or is able to guide the user to find precisely what she wants and/or also what she might want. An intelligent assistant not only may help the user find information but also assists the user in discovering information. Furthermore, an intelligent assistant may be able to converse with the user in order to enable the user to further refine the results, explore or drill-down the results, or find more information that is semantically relevant to the results.

[2984] An intelligent information retrieval system, according to an embodiment of the invention, may allow users to find knowledge, rather than information. Knowledge may be considered information infused with semantic meaning and/or exposed in a manner that is useful to people along with the rules, purposes and/or contexts of its use. Consistent with this definition (and/or others), knowledge, unlike information or data, may be based on context, semantics, and/or purpose. Today’s search engines have none of these three elements and/or, as a consequence, are fundamentally unequipped to deal with the problem of information overload.

[2985] In an embodiment, a retrieval system blends search and/or discovery for scenarios where the user does not even know what to search for in the first place. Searching for knowledge is not the same as searching for information. An intelligent search engine according to an embodiment of the invention allows a user to search with different knowledge filters that encapsulate semantic sensitivity, time-sensitivity, context-sensitivity, people (e.g., experts), etc. These filters may employ different ranking schemes consistent with the natural equivalent of the filter (e.g., a search for Best Bets may rank results based on semantic strength, a search for Breaking News may rank results based primarily on time-sensitivity, while a search for Experts may rank results based primarily on expertise level). These form context themes or templates that can guide the user to quickly find what she wants based on the scenario at hand.

[2986] For example, a user might want only latest (but also highly semantically relevant) information on a certain topic (perhaps because she is short on time and/or is preparing for a presentation that is due shortly)—this may be the equivalent of Breaking News. Or the user might be conducting research and/or might want to go deep—she might be interested in information that is of a very high level of semantic relevance. Or the user might want to go broad because she is exploring new topics of interest and/or is open to many possibilities. Or the user might be interested in relevant people on a given topic (communities of interest, experts, etc.) rather than—or in addition to—information on that topic. These are all valid but different real-world scenarios. An embodiment of the invention supports all these semantic axes in a consistent way yet exposes them separately so the user knows in what context the results are being displayed in order to aid him or her in interpreting the results.

[2987] Expressed formulaically, today’s search engines allow users to find i, where i represents information. In contrast, an embodiment of the invention allows users to find K, where K represents knowledge.

[2988] An embodiment of the invention allows for knowledge-based retrieval (expressed above as K) via knowledge filters (which may also be referred to as special agents or knowledge requests), each corresponding to a knowledge type. FIG. 104 illustrates defined knowledge filters/types in accordance with an embodiment of the invention. As used herein, the term “debates” may be an indication of semantic emphasis due to the participation of multiple individuals with potentially diverse viewpoints. Additionally, as an illustration, an Interest Group might include those that have questions (knowledge-seekers) and/or not just those that have answers (knowledge-providers or experts). This filter may connect both constituencies.

[2989] The ranking axes can be further refined and/or configured on the fly, based on user preferences. An embodiment of the invention also defines a special knowledge filter, a Dossier, which encapsulates every individual knowledge filter. A Dossier allows the user to retrieve comprehensive knowledge from one or more sources on one or more optional contextual filters, using one or more of the individual knowledge filters. For instance, in Life Sciences, a Dossier on Cardiovascular Disorder may be semantically processed as All Bets on Cardiovascular Disorder, Best Bets on Cardiovascular Disorder, Experts on Cardiovascular Disorder, etc. A Dossier may be akin to a “super knowledge-filter” and/or may be very powerful in that it can combine search and/or discovery via the different knowledge filters and/or allows users to retrieve knowledge in different contexts.

[2990] In an embodiment of the invention, the system’s model of knowledge filters and/or Dossiers has several interesting side-effects. First, it insulates the system from having to provide perfect ranking on any given axis before it can be of value to the user. The combination of multiple ranking and/or filtering axes guides the user to find what she wants via multiple semantic paths. As such, each semantic path becomes more effective when used in concert with other semantic paths in order to reach the eventual destination. Furthermore, an embodiment of the invention introduces Dynamic Linking, which allows the user to navigate multiple semantic paths recursively. This allows the user to navigate the knowledge space from and/or across multiple angles and/or perspectives, while iterating these perspectives potentially endlessly. This further allows the user to browse a dynamic, personal web of context as opposed to a web of pages or even a pre-authored semantic web which would still be author-centric rather than user-centric.

[2991] As an illustration, an embodiment of the invention allows a user to find Breaking News on a topic, then navigate to Experts on that Breaking News, then navigate to people that share the same Interest Group as those Experts, then navigate to what those people wrote, then navigate to Best Bets relevant to what they wrote, then navigate to Headlines relevant to those Best Bets, then navigate to Newsmakers on those headlines, etc. The user is able to navigate context and/or perspectives on the fly. Just as the Web empowers users
to navigate information, an embodiment of the invention empowers users to navigate knowledge.

[2992] An embodiment of the invention also defines information types, which may be semantic versions of well-known object and/or file types. These may include Documents (General Documents, Presentations, Text Documents, Web Pages, etc.), Events (Meetings, etc.), People, Email Messages, Distribution Lists, etc.

[2993] Context and/or Semantics. As described herein, an embodiment of the invention is able to interpret the context and/or semantics of a user’s query and/or also allows the user to express his or her intent via multiple contexts.

[2994] The Problem with Keywords. To mimic the intelligent behavior exhibited by a human research assistant or reference librarian, an embodiment of the invention first is able to “understand” what it stores and/or indexes. Today’s search engines do not know the difference between keywords when those keywords are used in different contexts. For instance, the word “bank” means very different things when used in the context of a commercial bank, river bank, or “the sudden bank of an airplane.” Even within the same knowledge domain, the problem still applies: for instance in the Life Sciences domain, the word “Cancer” could refer to the disease, the genetics of the disease, the pain related to the disease, the technologies for preventing the disease, the metaphor, the epidemic, or the public policy issue. The inability of search engines to make distinctions based on semantics and/or context is one of the causes of information overload because users must then manually filter out thousands or millions of irrelevant results that have the right keywords but in the wrong context (false positives).

[2995] An embodiment of the invention also is able to retrieve information that doesn’t have the user’s expressed keywords but which is semantically relevant to those keywords. This would address the false negatives problem—wherein search engines leave out results that they deem irrelevant only because the results don’t contain the “right” keywords. For instance, the word “bank” and/or the phrase “financial institution” are semantically very similar in the domain of financial services. An embodiment of the invention is able to recognize this and/or return the right results with either set of keywords.

[2996] Today’s search engines are also unable to understand semantic queries like “Find me technical articles on Security” (in the Computer Science domain). A semantic search for “Technical Articles on Security” is not the same as a Google™ search for “technical+articles+security” or even “technical articles:security.” A semantic search for “Technical Articles on Security” also returns, for example, Bulletins on Encryption, White Papers on Cryptography, and/or Research Papers on Key Management. These queries are all semantically equivalent to “Technical Articles on Security” even though they all contain different keywords. Furthermore, a semantic search for “Technical Articles on Security” does not return results on physical or corporate security, faults or sales.

[2997] As queries get more complex, the distinction between a keyword search and/or an intelligent search grows exponentially. For example, in the Life Sciences domain, a semantic search for “Research Reports on Cardiovascular Disorder and/or Protein Engineering and/or Neoplasim and/or Cancer” is far from being the same as a keyword search for “research reports:cardiovascular disorder:protein engineering:neoplasim:cancer.” For example, from a user’s standpoint, “Research Reports on Cardiovascular Disorder and/or Protein Engineering and/or Neoplasim and/or Cancer” also returns technical articles that are relevant to Hypertension (which is semantically related to Cardiovascular Disorder but has different keywords) and/or which are also relevant to Amino Acid Substitution (which is a form of Protein Engineering), and/or which are also relevant to Minimal Residual Disease (which is a form of Neoplasim and/or Cancer). The exponential growth of information combined with an exponential divergence in semantic relevance as queries become more complex could inevitably lead to a situation where information while plentiful, loses much of its value due to the absence of semantic and/or contextual filtering and/or retrieval.

[2998] Other forms of context. As described above, today’s search engines do not semantically interpret keywords. However, even if they did, this will not be sufficient for an intelligent information retrieval system because keywords are only one of many forms of context. In the real-world, context exists in many forms such as documents, local file-folders, categories, blobs of text (e.g., sections of documents), projects, location, etc. For instance, in an embodiment, a user is able to use a local document (or a document retrieved off the Web or some other remote repository) as context for a semantic query. This greatly enhances the user’s productivity—using prior technologies, the user has to manually determine the concepts in the documents and/or then map those concepts to keywords. This is either impossible or very time-consuming. In an embodiment, users are able to choose categories from one or more taxonomies (corresponding to one or more ontologies) and/or use those categories as the basis for a semantic search. Furthermore, in an embodiment, users are able to dynamically combine categories from the same taxonomy (or from multiple taxonomies) and/or cross-reference them based on their context.

[2999] An embodiment of the invention also allows users to combine different forms of context to match the user’s intent as precisely as possible. For example, a user is able to find semantically relevant knowledge on a combination of categories, keywords, and/or documents, if such a combination (applied with a Boolean operator like OR and AND/OR) accurately captures the user’s intent. Such flexibility is possible rather than forcing the user to choose a specific form of context that might not have the correct level of richness or granularity corresponding to his or her intent.

[3000] Expressed formulaically, an embodiment of the invention combines multiple knowledge axes (as described in section 3 above) with multiple forms of context to allow the user to find K(X), where K is knowledge and/or X represents different forms of context with varying semantic types and/or levels of richness—for instance, documents, keywords, categories, or a combination thereof.

[3001] The Problem with Google™. Google™ employs a technology called PageRank to address the keywords problem. PageRank ranks web pages based on how many other pages link to each page. This is a very clever technique as it attempts to infer meaning based on human judgment as to which pages are important relative to others. Furthermore, the technique does not rely on formal semantic markup or metadata, which is optionally advantageous in making the model practical and/or scalable. However, ranking pages based on popularity also has problems. First, without semantics or context, popularity has very little value. To take the examples cited above, “Technical Articles on Security” (to a computer
scientist) is not semantically equivalent to “Popular Pages on Bank Vaults or Safes.” The popularity of the returned results is irrelevant if the context of the user’s query is not intelligently interpreted—if the results are meaningless, that they might be popular makes no difference.

Second, PageRank relies on the presence of links to infer meaning. While this works relatively well in an organic, hypertext environment such as the Web, it is ineffective in business environments where majority of the documents do not have links. These include Microsoft Office documents, PDF documents, email messages, and/or documents in content management systems and/or databases. The scarcity (or absence) of links in most of these documents implies that PageRank would have no data with which to rank. In other words, if every document in the world were a PDF with no links, all documents may have a Page Rank of 0 and/or may be ranked equally. This then degenerates to a regular keyword search.

Third, popularity is only one contextual or ranking axis. In contrast, in the real-world there are multiple axes by which users acquire knowledge. Popularity is one but there are others including time-sensitivity (e.g., Breaking News or Headlines), annotations (indicating that others have taken the time to comment on certain documents), experts (which is a semantic axis via which users can navigate to authoritative information), recommendations (based on collaborative filtering or the user’s interests), etc. An embodiment of the invention allows for the seamless integration of all these axes to provide the user a comprehensive set of perspectives relevant to his or her query.

Fourth, Google™ relies on a centralized index of the Web. The index itself is based on disparate content sources and/or is distributed across many servers but the user “sees” only one index. However, in the real-world (especially in enterprise environments), knowledge is fragmented into silos. These silos include security silos (that restrict access based on the current user) and/or semantic silos (in which different knowledge-bases employ different ontologies which could interpret the same context differently). These silos call for Dynamic Knowledge Federation and/or Semantic Interpretation, not centralization. In an embodiment, the same piece of context is able to “flow” across different semantic silos, get interpreted locally (at each silo) and/or then generate results which then get synthesized dynamically. Furthermore, a user is able to seamlessly integrate results from different silos for which he/she has access (even if that access is mediated via different security credentials). This insulates the user from having to search each silo separately thereby allowing him or her focus on the task at hand.

Expressed formulaically, applying federation to the problem formulation and/or model definition, an embodiment is the triangulation of multiple knowledge axes via multiple optional context types semantically federated from multiple knowledge sources—i.e., K(X) from S1 . . . Sn, where K is knowledge, X is optional context (of varying types), and/or Sn is a knowledge index from source n that incorporates semantics. This model is potentially orders of magnitude more powerful than today’s search model which only provides I(x) from s, where i is information (and/or on only one axis; usually relevance or time), x is context (and/or of only one type—keywords, and/or which does not incorporate semantics), and/or s represents one index that lacks semantics and/or is not semantically federated with other silos.

The Problem with Directories and/or Taxonomies. Directories and/or taxonomies can be very useful tools in helping users organize and/or find information. Users employ folders in file-systems to organize their documents and/or pictures. Similar folders exist in email clients to assist users in organizing their email. Many portal products now offer categorization tools that automatically file indexed documents into directories using predefined taxonomies. However, as the volume of information users must deal with continues to skyrocket, directories become ineffective. This happens for several reasons: First, at “publishing time,” users manually create and/or maintain folders and/or subfolders and/or manually assign documents and/or email messages to these folders. This process not only takes a lot of time and/or effort, it also assumes that there is a 1:1 correspondence of item to folder. At a semantic level, the same item could “belong” to different folders and/or categories at the same time. Tools that employ machine learning techniques to aid users in assigning categories also suffer from the same problem.

Second, there is no perfect way to organize an information hierarchy. While users have the flexibility to create their own hierarchies on their computers, problems arise when they need to merge directories from other computers or when there are shared directories (for instance, on file shares). Shared directories are particularly problematic because an administrator typically has to design the hierarchy and/or such a design might be confusing to some or all users that need to find information using that hierarchy.

Third, at “retrieval time,” users are forced to “fit” their question or intent to the predefined hierarchy. However, in the real-world, questions are typically much more fuzzy, dynamic, and/or flexible and/or they occasionally involve cross-references. As illustrated in FIG. 105, a user might create a hierarchy for digital photos on his/her computer. This hierarchy might be sufficient up to a certain volume of information. However, as more and/or more pictures accumulate on the user’s computer, the user might want to ask complex queries such as: “Find me all pictures I took with my family and/or employees while skiing in France.” Because of the static, inflexible nature of the hierarchy, such a query becomes impossible because the specific context the user wants is not distinctly represented in the directory.

This problem becomes exacerbated in the online world with millions and/or billions of documents and/or hundreds and/or thousands of taxonomy categories. As an illustration, taxonomies in the Pharmaceuticals industry typically have tens of thousands of categories and/or are slow-changing. As such, the impact of the inflexibility of taxonomies and/or directories (which in turn leads to the preclusion of flexible semantic queries and/or search permutations) becomes exponentially worse as information volumes grow and/or also as taxonomies become larger. Users need the flexibility of cross-referencing categories in a taxonomy/ontology on the fly, and/or need to be able to cross-reference topics across taxonomies/ontologies. Research is fluid. Context is dynamic. Topics come and/or go. An embodiment of the invention captures this fluidity by allowing users to flexibly “ask” very natural-like questions, possibly involving dynamic permutations of concepts and/or topics, without the limitations of full-blown natural-language processing.

Applying this to the model definition, given the formulation K(X) from S1 . . . Sn, the ideal model allows X to include dynamic permutations of context of different types.
In other words, X is not only of multiple types, it also includes flexible combinations and/or cross-references of those types.

The Semantic Web and/or Metadata

[3011] As described herein, a first step in developing an embodiment of the invention is incorporating meaning into information and/or information indexes. In its simplest form, this is akin to creating an organized, meaning-based digital library out of unorganized information. The Worldwide Web Consortium (W3C) has proposed a set of standards, under the umbrella term the “Semantic Web,” for tagging information with metadata and/or semantic markup in order to infuse meaning into information and/or in order to make information easier for machines to process. The Semantic Web effort also includes standards to creating and/or maintaining ontologies which, in the context of information retrieval, are libraries and/or tools that help users formally express what information concepts mean and/or which also help machines disambiguate keywords and/or interpret them in a given domain of knowledge.

[3012] The Semantic Web is an initiative in that it may encourage information publishers to tag their content with more metadata in order to make such content easier to search. Furthermore, standards for ontology development and/or maintenance are useful in the establishment of systems that allow publishers to assert or interpret meaning. However, metadata has many problems, especially relating to the need for discipline on the part of publishers. Generally, history has shown that most publishers (including end-users who author Web pages, blogs, documents, etc.) do not exercise such discipline on a consistent basis. Metadata creation and/or maintenance need time and/or effort. As such, it is impractical to rely on its existence at scale. This is not to minimize the importance of efforts to promote metadata adherence. However, such efforts are complemented with the development of pragmatically designed systems that exploit when available—but do not rely on the existence of—such metadata.

[3013] It is also useful to distinguish structured metadata (for instance XML fields) from semantic (meaning-oriented) metadata. The former refers to fields such as the name of the author, the date of publication, etc. while the latter refers to ontological-based markup that clearly specifies what a piece of information means. As an illustration, one can have perfectly-formed, validated, structured metadata (e.g., an XML document) that is completely meaningless. Structured metadata (such as RDF and/or RSS) is indeed beneficial especially for queries that rely on structure (e.g., a query to find a specific medical record id, author name, etc.). However, majority of the queries at the level of knowledge are semantic in nature; this is one of the reasons why Google™ has succeeded despite the fact that it does not rely on any structured metadata; to Google™, all web pages are structurally identical (a web page is a web page). Consequently, while standards such as RDF and/or RSS are useful, they still do not address a problem—that of semantic indexing, processing, interpretation, retrieval, filtering, and/or ranking.

[3014] The Semantic Web effort appears to place research emphasis on formal, publisher-driven semantic markup. In very narrow, well-controlled domains, semantic markup would have value. However, problems arise at scale. For example, in one of the W3C presentations on the Semantic Web, the following illustration was cited in advocating the benefits of uniquely identifiable semantic tags: Don’t say “color” say “http://www.pantone.com/2002/stdf0/color/”

[3015] This part of the Semantic Web vision has problems reaching critical mass. Humans don’t want to change the way they write. Language has evolved over many thousands of years and/or it is unrealistic to expect that humans may instantly change the way they express themselves (or the effort they put into doing so) for the benefit of intelligent agents. Agents (and/or computers in general) can adapt to humans, not the other way round.

[3016] Semantic metadata relies on ontologies, which generally defined, are tools and/or libraries that describe concepts, categories, objects, and/relationships in a particular domain. The W3C recently approved the Web Ontology Language (OWL) which is a standard for ontology publishers to use to create, maintain, and/or share ontologies (see http://www.w3c.org/2001/sw/WebOnt/). This is a standard which accelerates the development of ontologies and/or ontology-dependent applications.

[3017] However, the development of ontologies presents new challenges. In particular, the expression and/or interpretation of meaning has many philosophical and/or technical challenges. What an item means is usually in the eyes of the beholder. Meaning is closely tied to context and/or perspective. As such, a piece of information can mean multiple things to different people at the same time or to the same person at different times. Differences in opinion, political ideology, research philosophy, affiliation, experience, timing, or background knowledge can influence how people infer or interpret meaning. In research communities, such differences reflect valid differences in perspective and/or are particularly acute in relatively new research areas. For instance, in Theoretical Physics, an ontology on String Theory is an expression of belief by those who believe in the theory in the first place. A body of knowledge in Physics that describes the quest for the Unified Field Theory can be viewed from multiple perspectives, each of which might legitimately reflect different approaches to the problem.

[3018] Consequently, it is not completely sufficient to empower a publisher to assert what his or her publication “means.” Rather, others are also able to express their semantic interpretation of what any piece of information “means to them.” Even if humans agreed to replace keywords with URIs (as indicated in the quote above), this still leaves the URIs open to interpretation in different contexts. A URI that is bound to a given context is not completely practical because it presupposes that only the author’s perspective matters or is accurate. The basis for contextual interpretation is separated from semantic markup in order to leave open the possibility for multiple perspectives. As such, going back to the quote above, it is fine for “color” to be expressed as “color” (and/or not as a URI) if the interpretation of “color” is realized in concert with one or more semantic annotations of what “color” might mean in a given context. Users are able to dynamically “navigate” across meaning boundaries even if those boundaries are not explicitly connected via semantic markup. From a pragmatic standpoint, this makes the case for more research emphasis on semantic dynamism (code) than on semantic markup (data).

[3019] The Problem with Today’s Search User Interfaces. Most of today’s search user interfaces (such as Google™) comprise of a text box into which users type keywords and/or phrases which are then used to filter results. Other common interfaces expose a directory or taxonomy from which users can then navigate to specific categories. Google’s™ user interface is especially popular due to its minimalist design—
it has a textbox and/or little else. While simplicity is part of a search user interface, it need not be at the expense of power and/or flexibility. A well-designed intelligent search user interface addresses the following optional features, in accordance with an embodiment of the invention:

[3020] 1. User Intent: A user interface allows a user to express her or her intent in a way that is as close as possible to what the person has in mind. Search engine users currently have to manually map their intent to keywords and/or phrases, even if those keywords and/or phrases do not accurately reflect their intent. There is as little as possible “semantic mismatch” between the user's intent and/or the process and/or interface used to express that intent. Natural language queries have been touted as the ideal search user interface. Indeed, natural language querying systems have had some success in limited domains such as Help systems in PC applications. However, such systems have been unsuccessful at scale primarily due to the technical difficulty of understanding and/or intelligently processing human language. The challenge therefore is to have a search user interface which is semantic (in that it empowers the user to express intent based on context and/or meaning), yet which does not suffer from the limitations of natural language query technology and/or interfaces. Furthermore, natural language queries require the user to know beforehand what she wants to know. As described herein, this does not reflect how people acquire knowledge in the real-world. A lot of knowledge is acquired based on discovery, serendipity, and/or contextual guidance—it is very common for people not to know what they might want to know until after the fact. As such, a search user interface according to an embodiment blends semantic search and/or discovery so the user is also able to acquire relevant knowledge (based on context) even without asking.

[3021] 2. Context and/or Semantics: A user interface also allows users to use multiple forms of context to express their intent. It is easy for users to dynamically use context to create semantic queries on the fly and/or to combine different types of context to create new personalized context consistent with the user's task.

[3022] 3. Time-sensitivity: A user interface also provides time-sensitive alerts and/or notifications that are semantically relevant to the displayed results. Time-sensitivity also is seamlessly integrated with context-sensitivity.

[3023] 4. Multiple Knowledge and/or Ranking Axes: A user interface also allows the user to issue semantic queries using one or more knowledge axes with different ranking schemes. In addition search results are presented in a way that reflects the context in which the query was issued—so as to guide the user in interpreting the results correctly.

[3024] 5. Behavior and/or Understanding: A user interface is able to dynamically invoke semantic Web services (or an equivalent) in order to connect displayed items dynamically with remote ontologies for the purpose of “understanding” what it displays in a given context.

[3025] 6. Semantic Cross-Referencing: A user interface allows the user to cross-reference context across ontologies. For instance, it is possible to use one perspective to view results that were generated via another perspective. Such “cross-fertilization of perspectives” accurately reflects how knowledge is acquired and/or how research evolves in the real-world. Furthermore, a user interface allows the user to cross-reference context in order to dynamically create new semantic views.

[3026] 7. Personalization—Knowledge Profiles: A user interface allows users to create different knowledge personas based on the task the user is focused on, different work scenarios, different sources of knowledge, and/or possibly, different ontologies and/or semantic boundaries. This is consistent with the connection of knowledge to purpose, as described herein.

[3027] 8. Personalization—Flexible Presentation: A user interface allows users to be able to customize how results get presented. Users are able to customize the visual style, fonts, colors, themes, and/or other presentation elements.

[3028] 9. Personalization—Attention Profiles: A user interface allows users to configure their attention profiles. These would be employed for alerts and/or other notifications in the user interface. These are not unlike profiles in mobile phones that specify whether a user can be disturbed or not, and/or if so, how—e.g., Normal, Silent, Meeting, etc.

[3029] 10. Federation—Knowledge Source Federation: A user interface allows the user to issue semantic queries and/or retrieve relevant results from diverse knowledge indexes and/or have those results presented in a synthesized manner—as though they came from one place. This allows the user to focus on his or her task without having to perform multiple queries (to different sources) each time.

[3030] 11. Federation—Semantic Federation: A user interface allows the user to issue semantic queries to diverse knowledge indexes even if those indexes cross semantic (or ontology) boundaries. A user interface allows the user to hide semantic differences during the query process (if she so wishes for the task at hand)—the user is able to configure the knowledge indexes and/or issue queries without having to know that context-switching is dynamically occurring in the background while queries are being processed.

[3031] 12. Federation—Security Federation: A user interface allows the user to seamlessly issue semantic queries and/or retrieve relevant results across security silos even if she uses different security credentials to access these silos.

[3032] 13. Awareness: A user interface allows the user to keep track of context and/or time-sensitive information across multiple knowledge sources simultaneously.

[3033] 14. Attention-Management: A user interface may only be disrupted or interrupted when absolutely necessary based on the user's current task and/or the user's attention profile. This is similar to what an efficient human assistant or research librarian would do.

[3034] 15. Dynamic Follow-up and/or Drill-down: A user interface allows the user to dynamically follow-up on results that get retrieved by issuing new queries that are semantically relevant to those results or by drilling down on the results to get more insights. This is similar to what typically happens in the real-world: the retrieval of results by an efficient research librarian is not the end of the process; rather, it usually marks the beginning of a process which then involves intellectual exchange and/or follow-up so the user can dig into the results to gain additional perspective. The acquisition of knowledge is a never-ending, recursive process.

[3035] 16. Time-Management—Summaries, Previews, and/or Hints: A user interface also proactively saves the user's time to providing summaries, previews, and/or hints. For instance, a user interface allows a user to determine whether she wants to view a result or navigate a new contextual axis before the commitment to navigate actually gets made. This enhances browsing productivity.
17. Discoverability of new Knowledge Sources: A user interface allows the user to dynamically discover new knowledge sources (with semantic indexes) as they come online.

18. Seamless integration with user context and/or workflow: A user interface is seamlessly integrated with the user’s context and/or workflow. The user is able to easily “flow” between his or her context and/or the user interface.

19. Knowledge Capture and/or Sharing: A user interface enables the user to easily share knowledge with his or her communities of knowledge. This includes easy knowledge publishing that encourages users to share knowledge and/or annotations so users can provide opinions and/or commentary on results that get displayed in the user interface.

20. Context Sharing and/or Collaboration: A user interface allows users to be able to easily share dynamic context and/or queries.

21. Ease of Use and/or Feature Discoverability: A user interface is easy to use. It provides power and/or flexibility and/or should support the optional features listed above but it does so in a way that is easy to learn and/or use. Also, the features supported in a user interface are easy for users to find and/or manage, and/or are exposed in a way that is contextually relevant to the user’s task but without overwhelming the user.

22. Semantic Indexing. In order to support intelligent retrieval, an embodiment of the invention uses a model for integrating semantics into an information index. Such a semantic index meets the following optional features, in accordance with an embodiment of the invention:

1. Multiple schemas: the index allows multiple well-known object types with different schemas (e.g., documents, events, people, email messages, etc.) to co-exist in a consistent data model. However, the index does not depend on the existence of rich metadata; the index may allow for cases where the schema is sparsely populated (except for core fields such as the source of the data) due to the absence of published metadata.

2. Flexible knowledge representation: the index allows for the flexible representation of knowledge. This representation allows for a rich set of semantic links to describe how objects in the index relate to one another.

3. Seamless domain-specific and/or domain-independent knowledge representation: the semantic index also allows for semantic links that refer to category objects that are domain and/or ontology specific. However, the index has a consistent data model that also includes domain-independent semantic links. For example, the semantic link described with a predicate “is category of” is domain and/or ontology dependent whereas a semantic link described with a predicate “reports to” or “authored” is domain-independent. Such semantic links co-exist to allow for rich semantic queries that cut across both classes of predicates.

4. Multiple perspectives: seamless semantic federation and/or ontology co-existence: As described herein, a semantic system supports multiple viewpoints of the same information in order to capture the polymorphism of interpretation that exists in the real world. As such, a semantic index allows semantic links to co-exist in the same data model across diverse ontologies. Furthermore, the semantic index is able to be federated with other semantic indexes in order to create a virtual network of meaning that crosses boundaries of perspective (or semantic silos). Support for semantic federation also implies that the semantic index is complemented with an intelligent semantic query processor that can dynamically map context to the semantic index in order to retrieve results from the semantic index according to the ontologies represented in the index. These results can then be federated with results from other semantic indexes to create a consistent yet virtual query model that crosses semantic boundaries.

5. Inference: the index also supports inference engines that can “observe” the evolution of the index and/or infer new semantic links accordingly. For example, semantic links that relate to document authorship can be interpreted along with semantic links that define how documents relate to categories (of one or more ontologies) to infer topical expertise. The semantic index allows an inference engine to be able to mine and/or create semantic links.

6. Maintenance: The semantic index is maintainable. Semantic links are easily updatable and/or deleted links are removed without affecting the integrity of the entire index.

7. Performance and/or Scalability: The semantic index interprets and/or responds to real-time, dynamic semantic queries. As such, the index is carefully designed and/or tuned to be very responsive and/or to be very scaleable. Indexing speed, query response speed, and/or maximum scalability (via scale-up and/or scale-out) are on the same order of magnitude as the performance and/or scalability of today’s search engines.

7.1 Dynamic Semantic Indexing in the Information Nervous System. Semantic indexing in an embodiment of the invention is accomplished with two components: one that handles the dynamic processing of semantics (called the Knowledge Domain Service (KDS)) and/or another that integrates meaning into a semantic index (called the Knowledge Integration Service (KIS)).

7.1.1 The Knowledge Domain Service. The Knowledge Domain Service (KDS) hosts one or more ontologies belonging to one or more knowledge domains (e.g., Life Sciences, Information Technology, Aerospace, etc.). The KDS exposes its services via an XML Web Service interface. The primary methods on this interface allow clients to enumerate the ontologies installed on the KDS and/or to retrieve semantic metadata describing what a document, text blob, or list of concepts (passed in as input) “means” according to a given ontology on the KDS. The KDS Web service returns its results via XML. FIG. 106 shows an example of metadata fields that the KDS returns when “asked” to enumerate its installed ontologies, in accordance with an embodiment of the invention. The Knowledge Domain ID uniquely identifies the ontology. The Knowledge Domain Name is a friendly name that describes the knowledge domain. The Knowledge Domain Publisher Name is the name of the ontology publisher. The Knowledge Domain Publisher Domain Name identifies the publisher on the Internet, Intranet, or Extranet. The Knowledge Domain Publisher Zone indicates the scope of the domain name (Internet, Intranet, or Extranet). This model allows for both public and/or private ontologies to share the same ontology namespace.

7.1.1.1 When asked to categorize an information item according to an ontology, the KDS Web service may return XML that describes a list of mappings—nodes in the ontology and/or weights that describe the semantic density of the input item per node. For instance, in a typical scenario, a client of the KDS Web service would pass in a URL to a Web page (in the Life Sciences knowledge domain) and/or also pass in a unique identifier that refers to the ontology that the client wants the KDS to use to interpret the input (presumably
an ontology in the Life Sciences domain). FIG. 107 illustrates the schema and/or sample fields of a KDS result, in accordance with an embodiment of the invention.

[3052] This result describes the name of the node in the taxonomy/ontology (“Cardiovascular Disorder Epidemiology”), a Uniform Resource Identifier (URI) that uniquely identifies the node in the ontology, and/or a weight that captures the frequency of incidence of concepts in the input item measured against the concepts in the ontology around the returned node. The inclusion of the knowledge domain identifier (which identifies the ontology) and/or the full path of the node within that ontology ensure that the returned URI is unique from a semantic standpoint. New ontologies are assigned new unique identifiers in order to distinguish them from existing ontologies.

[3053] 7.1.2 The Knowledge Integration Service (KIS), in accordance with an embodiment of the invention, crawls and/or semantically integrates disparate sources of information (such as Web sites, file shares, Email stores, databases, etc.). The crawling functionality can be separated out into another service for scalability and/or load balancing purposes. The KIS may have an administration interface that allows the administrator to create one or more knowledge bases. The knowledge base may be called a “Knowledge Community” because it includes not only semantic information but also People. For a given knowledge community (KC), the administrator can set up information sources to be indexed for that KC. In addition, the administrator can configure the KC with one or more knowledge domains, including the URL to the KDS Web service and/or the unique identifier of the ontology to be used to create the semantic index. The KC can allow the administrator to use multiple ontologies in indexing the same set of information sources—this allows for multiple perspectives to be integrated into the semantic index.

[3054] As the KIS crawls information sources for a given KC (e.g., Web sites), it can pass the URL of the crawled information item to each of the KDS Web services it has been configured with for that KC. This is akin to the KIS “asking” each KDS what the item “means to it.” Note that there is still no universal notion of what the item means. The item could mean different things to different KDSes and/or ontologies. Because the XML returned by each KDS can uniquely identify the ontology entry, the KIS now has enough information with which to annotate the information item with meaning, while preserving the flexibility of multiple and potentially diverse semantic interpretations.

[3055] The KIS can store its data using a semantic network. The network may be represented via triples that have subject nodes, predicates, and/or object nodes and stored in a relational database. The semantic network can include objects of various semantic types (such as documents, email messages, people, email distribution lists, events, customers, products, categories, etc.). As the KIS crawls objects (e.g., documents), the objects may be added to the semantic network as subjects and/or predicates are assigned and/or linked to the network dynamically as each object gets semantically processed and/or indexed. Examples of predicates include “belongs to category” (linking a document with a category), “includes concept” (linking a document with a concept or keyword), “reports to” (linking a person with a person), etc. The subject entries in the semantic network also include rich metadata, if such metadata is available. This provides the KIS with a rich index of both structured metadata (if available) and/or semantic metadata from multiple perspectives. However, the latter does not rely on the former—the KIS is able to build a semantic network with semantic metadata even if the subjects in the network do not have structured metadata (e.g., legacy Web pages). The implication of this is that with the KIS and/or KDS, an embodiment of the invention can provide a semantic user experience even without semantic markup or a Semantic Web. FIG. 108 illustrates the representation of a semantic network in the KIS, in accordance with an embodiment of the invention. As the KIS retrieves category information back from each KDS it may be configured with, it can add new categories into the semantic network if those categories do not exist already.

[3056] FIG. 109 illustrates the schema and/or sample fields of a category that gets added to the semantic network, in accordance with an embodiment of the invention. The Name and/or URI fields are consistent with the schema of what gets returned by the KDS.

[3057] FIG. 110 illustrates the separation of the KIS and/or KDS for the purposes of supporting multiple perspectives, and/or also how they work together to build the semantic index which is managed by the KIS, in accordance with an embodiment of the invention. FIG. 110 also shows the client (the semantic browser) and/or how it interacts with the KIS to issue semantic queries and/or retrieve results. An embodiment of the invention is able to access and/or index content from diverse repositories. Many enterprises have standard and/or custom repositories that run on multiple platforms. An embodiment of the invention is able to access all these repositories. The KIS has been designed to natively support file shares, Web sites, RSS and/or OPML. Additional native connectors include email (for the System In-box, which may be used for publications and/or annotations) and/or LDAP directories (for People). Custom repositories are supported via a standard architecture involving RSS over HTTP. This keeps the KIS architecture clean and/or stable and/or abstract out schema and/or platform differences at the connector level. Each connector may be a standalone product that “speaks” RSS over HTTP. The KIS can then index the generated RSS feed similar to any “standard” RSS feed. On Windows, connectors may be implemented as ASP.NET applications. This provides HTTP accessability. Each connector can support the following: 1. Multiple Endpoints: Each connector may be configured with one or more endpoints specific to the application in question. For instance, an email connector may be able to be configured with multiple inboxes that are abstracted via RSS. Each connector can define its own endpoint and/or store configuration state as needed. Each endpoint is able to live on its own servers (endpoints can be federated). 2. RSS Feed Web Folders: Each connector can allow the administrator to configure an RSS feed web folder per endpoint or an RSS web folder for all endpoints. The administrator may want an RSS feed (and/or web folder) per endpoint or might want to have an aggregate feed that encapsulates all endpoints. Both options are allowed. 3. Automatic Updates: Each connector can automatically “crawl” its endpoints and/or generate up-to-date RSS feeds that represent these endpoints. The connector can allow the administrator to configure the crawl frequency per endpoint or for the entire application. 4. RSS Version: Each connector can generate RSS version 2.0. 5. HTTP Addressability: Each connector can generate a URL that abstracts an information item, based on the application in question. For instance, a document in a content management system has an HTTP URL that the connector ASP.NET (or equivalent) application processes to
return the contents of the document. This is a "cross-application redirect." The connector is responsible for passing HTTP GET requests across application boundaries in order to retrieve the information item(s). 6. RSS Item Caching: Optionally, each connector could cache the generated list of RSS items in a local database installed with the product (e.g., SQL Server Express). This cache would allow sophisticated filtering and/or queries in order to retrieve "sub-feeds" based on queries the administrator defines. 7. Search Queries: Optionally, each connector could accept arguments to its RSS feed HTTP URL endpoint that represents search arguments. The connector could then return a "sub-feed" that corresponds to the search. 8. Required HTTP Headers: Each connector, in an embodiment, can return the following headers in response to the HTTP "HEAD" request: CONTENT-LENGTH: This returns the size of the information item. CONTENT-TYPE: This returns the MIME type of the information item. LAST-MODIFIED: This returns the last modified date-time of the information item. CONTENT-LANGUAGE: This returns the language in which the information item is encoded. 9. Authentication Information: Each connector can allow the administrator to provide authentication information for each endpoint. The connector can perform the authentication needed to access each endpoint, using the authentication information provided by the administrator. 10. Configuration User Interface: Each connector can provide a user interface (via a Web admin or Windows forms or an equivalent) to allow the administrator to: Add/remove endpoints (including authentication information) and/or corresponding RSS feeds and/or Schedule crawls. Connector Components: The connector components include a set of base components and/or custom components that can be connector-specific. The base components are implemented so that their interfaces and/or methods can be overridden as needed by individual connectors. The Base Component set includes, in an embodiment: 1. Endpoint (IEndPoint): this component abstracts out the details of a specific endpoint. The data representation is a URI, which is a virtual identifier that represents the endpoint. Each endpoint also has optional authentication information, a username and/or password. Each connector has its own implementation of an endpoint, with code to interpret the URI. Each endpoint object is responsible for crawling itself. This is not unlike how the Directory object in .NET is responsible for enumerating its files. In this context, the component is responsible for connecting to an endpoint, retrieving data from the endpoint and/or mapping the data to EndpointItem objects. This is not unlike how the Directory object in .NET returns FileInfo objects. Objects implementing the IEndPoint interface may optionally be able to page through the data they enumerate, and/or optionally take search parameters to restrict the result set. 2. Endpoint Manager: this component manages the storing and/or retrieval of endpoint configuration settings, including the secure storage of authentication information as needed. The Endpoint Manager deals with abstract Endpoint objects. 3. EndpointItem: this component abstracts out an Endpoint item. An EndpointItem includes connector-specific endpoint information that identifies the item to be retrieved. An EndpointItem object is also responsible for fetching the data for the object it represents. Each EndpointItem is also able to convert its data representation to RSS. 4. RSS Generator: this component generates the master RSS feed for an endpoint. The component does not know how the RSS is generated—this is the responsibility of the connector. The RSS is fed into the generator via EndpointItem objects. The RSS Generator component is also able to chop this feed into multiple RSS files and/or generate a master OPML feed that refers to the RSS feeds. The generator is able to persist the RSS feed(s) to configured Web folders for remote access, via local file copy or FTP. 5. EndpointScheduler: this component stores and/or retrieves configuration settings for scheduling endpoint crawls. The component is also responsible for invoking and/or stopping crawls based on configured schedules. 6. EndpointItemCache: this component manages the storage of cached RSS items—to a local store (e.g. a SQL store). 7. EndpointConnector: this is the component that is exposed to callers, primarily the ASP.NET application. Initially, this is a managed interface (e.g., a .NET assembly). This component exposes all the methods needed for abstracting an RSS feed, and/or returning data for an RSS item, given a set of arguments. These arguments are fed to the component by the ASP.NET application in response to an HTTP request. The RSS is returned to the component either in a memory buffer or via a Web folder path, if the entire RSS feed for an endpoint is requested. 8. ASP.NET Application: this is the ASP.NET application that maps HTTP requests ("HEAD" and/or "GET") to and/or from the RSSConnector component. The following disclosure is in accordance with an embodiment of the invention. Parts of the invention may be practiced alone or in combination with one or more other parts of the invention.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Indexed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectId</td>
<td>BIGINT</td>
<td>Yes</td>
<td>(primary key; clustered)</td>
</tr>
<tr>
<td>ObjectTypeId</td>
<td>INT (4 bytes)</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td>UNICODE String</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>UNICODE String</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>SourceUri</td>
<td>UNICODE String</td>
<td>Yes (non-clustered)</td>
<td>UNIQUE constraint</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data Type</td>
<td>Indexed</td>
<td>Comments</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Language</td>
<td>UNICODE String</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OriginalCreationTime</td>
<td>DATETIME</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OriginalLastModifiedTime</td>
<td>DATETIME</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ObjectCreationTime</td>
<td>DATETIME</td>
<td>Yes (non-clustered)</td>
<td></td>
</tr>
<tr>
<td>ObjectLastModifiedTime</td>
<td>DATETIME</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>BIGINT</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>BetStrength</td>
<td>BIGINT</td>
<td>No</td>
<td>Indicates the aggregate semantic strength of the document</td>
</tr>
<tr>
<td>NumConcepts</td>
<td>BIGINT</td>
<td>No</td>
<td>Indicates the number of concepts in the document</td>
</tr>
<tr>
<td>Creators</td>
<td>UNICODE String</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Contributors</td>
<td>UNICODE String</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Publishers</td>
<td>UNICODE String</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>BestBetHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td>Indicates whether this is a Best Bet. This is updated by the Semantic Inference Engine (SIE).</td>
</tr>
<tr>
<td>RecommendationHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td>Indicates whether this is a Recommendation. This is updated by the Semantic Inference Engine (default value is 2/3 the Best Bet semantic strength).</td>
</tr>
<tr>
<td>BreakingNewsHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td>Indicates whether this is Breaking News. This is updated by the Time-Sensitivity Inference Engine. Currently, this is implemented based on the intersection of the specified Breaking News time threshold and/or the Recommendations semantic strength</td>
</tr>
<tr>
<td>HeadlinesHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td>Indicates whether this is Breaking News. This is updated by the Time-Sensitivity Inference Engine. Currently, this is implemented based on the intersection of the specified Headlines time threshold and/or the Recommendations semantic strength</td>
</tr>
<tr>
<td>BetRankHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes (non-clustered)</td>
<td>This is a representative score of the semantic strength from 0-10</td>
</tr>
<tr>
<td>RichMetadataHint</td>
<td>SMALLINT (2 bytes)</td>
<td>No</td>
<td>This indicates whether the document came from a rich metadata source (like RSS)</td>
</tr>
<tr>
<td>SemanticHash</td>
<td>UNICODE String</td>
<td>No</td>
<td>This is a hash of the body of the document; used for duplicate detection. Currently, this is implemented by appending the concepts (key phrases) of the document in alphabetical order</td>
</tr>
</tbody>
</table>
[3061] Objects Table Data and/or Index Model.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Indexed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectID</td>
<td>BIGINT (8 bytes)</td>
<td>Yes</td>
<td>(primary key; clustered)</td>
</tr>
<tr>
<td>ObjectTypeID</td>
<td>INT (4 bytes)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Uri</td>
<td>UNICODE String</td>
<td>Yes</td>
<td>(non-clustered)</td>
</tr>
</tbody>
</table>

[3062] Semantic Links Table Data and/or Index Model

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Indexed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinkID</td>
<td>BIGINT (8 bytes)</td>
<td>Yes</td>
<td>(non-clustered)</td>
</tr>
<tr>
<td>SubjectID</td>
<td>BIGINT (8 bytes)</td>
<td>Yes</td>
<td>(non-clustered)</td>
</tr>
<tr>
<td>PredicateTypeID</td>
<td>INT (4 bytes)</td>
<td>Yes</td>
<td>(non-clustered)</td>
</tr>
<tr>
<td>ObjectID</td>
<td>BIGINT (8 bytes)</td>
<td>Yes</td>
<td>(non-clustered)</td>
</tr>
<tr>
<td>LinkStrength</td>
<td>BIGINT (8 bytes)</td>
<td>Yes</td>
<td>(non-clustered)</td>
</tr>
<tr>
<td>BestBetHint</td>
<td>SMALLBINT (2 bytes)</td>
<td>Yes</td>
<td>(non-clustered)</td>
</tr>
<tr>
<td>RecommendationHit</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes</td>
<td>(non-clustered)</td>
</tr>
<tr>
<td>BreakingNewsHit</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes</td>
<td>(non-clustered)</td>
</tr>
<tr>
<td>HeadlinesHit</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes</td>
<td>(non-clustered)</td>
</tr>
<tr>
<td>BetRankHint</td>
<td>SMALLINT (2 bytes)</td>
<td>Yes</td>
<td>(non-clustered)</td>
</tr>
</tbody>
</table>

[3063] There may be a composite index which is the primary key (thereby making it clustered), thereby facilitating fast joins off the SemanticLinks table since the database query processor may be able to fetch the semantic link rows without requiring a bookmark lookup and/or which may include the following columns: SubjectID; PredicateTypeID; ObjectID; BestBetHint; RecommendationHit; BreakingNewsHit; HeadlinesHit; BetRankHint. This makes incremental indexing of meta-links (like RSS feeds) very fast because the KIS doesn’t need to check each individual item referred by the link.

[3064] Fast Incremental Meta-Indexing. Fast Incremental Meta-Indexing (FIM) refers to a feature of the Knowledge Integration Service (KIS) of an embodiment of the invention. This feature can apply to the case where the KIS indexes RSS (or other meta) feeds. On an incremental index, the KIS can check each item to see whether it has already indexed the item. In the case of a feeds like RSS feeds, the “item” (e.g., a URL to an RSS feed) contains the individual items to be indexed. In this case, the KIS keeps track of which RSS items it has indexed via a MetaLinks table in the Semantic Metadata Store (SMS). On an incremental index, the KIS checks this table to see if the meta-link (e.g., an RSS URL) has been indexed. If it has, the KIS skips the entire meta-link. This makes incremental indexing of meta-links (like RSS feeds) very fast because the KIS doesn’t need to check each individual item referred by the link.

[3065] Adaptive Ranking. The Knowledge Integration Service (KIS) in an embodiment of the invention assigns Best Bets based on the semantic strength of a semantic object (e.g., a document) in a given context (e.g., a category), based on the categorization results of the Knowledge Domain Service.
TEXT. The SIE can implement this by invoking the aforementioned rule for each category by traversing each semantic link in the semantic network.

[3067] Notes on Adaptive Ranking. In an embodiment, the implication of Adaptive Ranking is that Best Bets are now actually Best Bets and/or not Great Bets (as was the case previously); there may always be Best Bets. A document can stop being a Best Bet—if the index changes, what was previously “Best” might become “Average” or “OK.” A document can stop being a Recommendation in a manner similar to that described above. A document can suddenly stop being Breaking News, if it no longer constitutes News (if its rank is now poor, relative to the distribution). This is akin to CNN Headline News where some “Headlines” can stop being Headlines across a 30-minute boundaries (due to a new prevalence of much more important “News”). Or where “Headlines” can get “bumped” from the queue due to late-breaking news (which might be slightly older—but too longer to report—but more important). This change is not critical when all documents have a large (full-text) semantic density—with a consistent semantic distribution (Great Bets tended to be Best Bets). However, with abstracts (as is the case with Medline), this assumption doesn’t hold. This change now means that Best Bets, Recommendations, Breaking News, and/or Headlines are much more reliable and/or accurate. The Adaptive Ranking may only cause these jumps while the semantic distribution is unstable. Once the distribution stabilizes, Best Bets may remain “Best.” And/or so on . . . So these illustrations may be most apparent EARLY in the indexing cycle—before the semantic distribution matures.

[3068] Pagination and/or Content Transformation. Many documents that knowledge-workers search for are lengthy in nature and/or occasionally could cover a lot of different topics. If the complete documents are indexed by the Knowledge Integration Server (KIS), the end-user may get results at the client corresponding to the full documents. For very long documents, this could be frustrating because only specific sections of the documents could be semantically relevant in the context of the user’s request. To address this, an embodiment of the invention has a feature wherein the documents get paginated before they are semantically indexed. The pagination may be done in a staging process upstream of the indexing process. Each paginated document then may have a hyperlink to the original document. When the user views the paginated document, the user can then navigate to the original document. This model ensures that if only specific pages within a long document are semantically relevant, then those pages may get returned and/or the user may see the specific pages in the right context (e.g., Best Bets). Furthermore, with Adaptive Ranking and/or Smart Adaptive Ranking in place, there may not be any loss in relative precision or recall when indexing pages rather than full documents, due to the relativistic nature of the ranking algorithm. In another embodiment, other types of document subsets (and/or not only pages) can be indexed. For instance, chapters, sections, etc. can also be indexed using the same technique described above. See, for example, the Pagination Pipeline Architecture Diagram in FIG. 114. In one embodiment, this model is extended to cover other types of “content transformations.” Examples include optical-character-recognition (for image-to-text conversion), language translation, and/or content-cleansing (e.g., removing ads from web pages). In this model, the second stage in FIG. 114 is replaced with a generic “content transformation” stage as shown in FIG. 115. In one embodiment, this is rep-
represented by a Content Transformation Service (CTS), implemented as a Web Service. As the KIS crawls information items using the Data Source Adapters (DSAs), it can be configured to first transform the content via one or more CTSes. In this scenario, the CTS acts as a KDS except that its function is to transform content rather than categorize content. CTSes can also be chained together such that one CTS can call another CTS to perform another layer of transformation (and/or so on). In one embodiment, KIS support for the content transformation pipeline may be handled via RSS. For each RSS item, the output (transformed) RSS file may have a Nervana namespace-qualified tag (linkToBindexed). If this element has an entry, the KIS can index this link (the user may still see the original link). Else the KIS can index the original link. See, for example, FIG. 115.

[3069] Semantic Highlighting is a feature of an embodiment of the invention that allows users to view the semantically relevant terms when they get results from a semantic query using the semantic client. This is much more powerful than today’s regular keyword highlighting systems because with semantic highlighting, the user may be able to see why a result was semantically chosen by viewing the keywords, based on the context of the semantic query. The first part of the implementation has to do with the fetching of the terms to be highlighted for a given query. This can be implemented on the client or on the server. Doing it on the client has the advantage of user scalability since the local CPU power of the client can be exploited (on the other hand, the server would have to do this for each client that accesses it). However, doing this on the server has the advantage of ontology scalability because servers typically would have more CPU and/or memory resources to be able to navigate large ontology graphs in order to fetch the highlight candidate terms. The following steps describe the implementation of one embodiment (with occasional references to the alternative (server-side) embodiment): 1. The client semantic runtime may lazily cache an ontology graph for each ontology in each KC it subscribes to. In one embodiment, this graph may be handled via the XPath Navigator (e.g., the XPathNavigator object in the .NET Common Language Runtime (CLR)—the navigator object itself gets cached (for large graphs, this could take a while to load and/or caching it may make highlighting performance quick). Alternatively, this could be manually represented as a set of hash tables for quick, constant-time (O(1)) lookup. These hash tables may then point to hash tables (one set of hooks and/or another for exclusions) which would include the ontology terms. The graph may be pre-persisted to disk but may only be cached to memory lazily to minimize memory usage. In an alternative embodiment, the server may do the same. The server may cache one ontology graph across all its KCs—since there might be different KCs that have the same ontologies. 2. The client semantic runtime may download all the ontologies from the KC the user is subscribed to. It does this so as to be able to cache the graphs locally. To download the ontologies, the client asks the KC for the ontology GUIDs it is configured with as well as the KDS server names that host the ontologies. In one embodiment, the client then downloads the ontologies via HTTP by invoking a dynamically constructed URL (like http://kds.nervana.com/nervkdsnont/guid>ontology.ont.xml). “NervKDSOnt” is a virtual folder installed with the KDS and/or which points to the root of the ontology folder (containing the ontology plug-ins installed on the KDS). 3. For virtual KCs (where the KC is a redirector to standard or “real” KCs—for federation purposes), the client might not have direct access to the KDSes that the KIS that hosts the KC refers to. For instance, an Internet-facing KC might federate many local KCs within a private workgroup that isn’t accessible to clients over the Internet. In this scenario, the client first tries to download the ontologies from the KDS. If this fails, it then tries the KIS. As such, in one embodiment, the virtual KC has (locally installed) all the ontologies that the KCs it federates has. 4. The client semantic runtime may intelligently manage memory usage for large ontology graphs. It may only cache large ontology graphs if there is available memory. In this embodiment, the following rules may be employed: i. If the ontology file is larger than 16 MB, the available physical memory threshold may be set at 512 MB (the client may only cache the ontology if there is at least 512 MB of physical memory available). ii. If the ontology file is between 8 MB and/or 16 MB in size, the available physical memory threshold may be set at 256 MB. iii. If the ontology file is less than 8 MB in size, the available physical memory threshold may be set at 128 MB. 5. The client semantic runtime may expose an API to the client Presentation engine (the Presenter), which may take one argument: the SourceUri of the item being displayed. The Presenter’s semantic engine may then include the ObjectID and/or ProfileID of the containing request to the call to the client semantic runtime. 6. The API may return a list of Highlight Candidate Terms (HCTs). In the embodiment, this may be returned as an XML file. The XML can contain additional metadata for each HCT such as whether it is a keyword or category, or whether it is from an entity or document (etc.). The Presentation engine can then use this to highlight keywords and/or categories differently, and/or so on. 7. The HCT list may be generated as follows: i. In the embodiment, the HCT list XML file may be independent of any given result that is generated from the semantic query. However, in an alternative embodiment, especially if the HCT list is large (e.g., if a category in the semantic query is high up in the hierarchy of a large ontology), the client semantic runtime can retrieve the HCT list as follows: 1. It may first get the concepts (key phrases) of the result URI (for which highlighting terms are to be displayed) by calling the client-side concept extractor and/or categorizer (which is already part of the semantic client infrastructure for Dynamic Linking support—like Drag and/or Drop). This is an advantageous step as it avoids the need to return a large list of terms each time (especially for very broad categories high-up in the hierarchy). 2. For each key phrase, the runtime may check if the phrase matches ANY of the categories in the SQML representing the containing request. For each category, the runtime may walk the ontology graph and/or check if the key phrase is in the category’s hooks table is NOT in the category’s exclusions table, is any of the category’s descendant hooks tables, and/or is NOT in any of the category’s descendants exclusions tables. This algorithm may optimize for the smaller set (the key phrases in the document), rather than the [potentially] larger set (the ontologies). On average, this performs very well. This means that even for broad categories like Cancer and/or Neoplasm in the Cancer (NCI) ontology (perhaps with hundreds of thousands of hooks), the algorithm still performs O(N) where N is the number of concepts in the source document, NOT the number of terms in the broad category. ii. In one embodiment, terms for categories are obtained via the XPathNavigator. For each category in the SQML, XPath queries are used to find the hooks of the category and/or all its descendant categories. These terms are all added to the term list and/or annotated.
appropriately as having come from categories. iii. If the request involves Dynamic Linking (e.g., from Drag and/or Drop), the context may be first dynamically interpreted. The client first extracts the concepts in a domain (ontology)—independent way. In one embodiment, the client passes the extracted concepts directly to the KDSes for the KC in question (and/or does this for each KC in the profile in question—to get federated HCTs). The KDSes then return the category URLs corresponding to the concepts. In an alternative embodiment, the client passes the concepts to the KIS hosting the KC. The KIS then passes the concepts to the KDSes. Step ii above is then invoked for the categories. iv. The client may cache the categories for dynamic context so that if the user invokes the query again, a cache-hit may result in faster performance. The client holds on to the cache entry for floating text and/or flush the cache for documents or entities if the documents or entities change (before checking for a cache-hit, the client checks the last modified time-stamp of the document or entity. If there is a cache-miss, the concept extraction and/or categorization may be re-invoked and/or the cache updated. v. If there are keywords in the SQML, each keyword may be added to the term-list (the HCT list). vi. If there are exact phrases in the SQML, the exact phrases may be added to the term-list (the HCT list). vii. The client-side ontology graph may be updated periodically (for each subscribed KC). This may involve updating the ontology cache as the user subscribes to and/or unsubscribes from KCs. 9. Wire up the Ontology Graph Data Engine into the client runtime. This may involve a cache of the XPathDocument, XMLTextReader, ontology file size (to check for updates in the case of redirected or dynamically generated ontologies), ontology last modified file time (to check for updates), and/or the file path to the Ontology Cache. 10. Likewise for the server-side ontology graph (for each KDS). 11. When a semantic query request is launched in the semantic client, the Presentation engine then may call the HCT extraction API, processes the XML results, and/or then highlights the terms in the Presenter (for titles, summaries, and/or the main body, where appropriate). Once this is done, the implementation may be complete (as currently specified).

KIS Indexing Pipeline. In one embodiment, the KIS has the following optimizations. More parallel pipelines to the KIS indexing system. This change now parallelizes indexing and/or I/O so that the KIS is able to index some documents while blocked on I/O from the KDS. This also allows the KIS to scale better with the number of CPUs. In an inefficient embodiment, for one KC, these operations would be serialized. This change could result in a 2-fold to 3-fold speedup in indexing performance on one server. Streamlining the KIS data model to remove redundant (or unused indexes). This improves indexing performance. Added KDS batching to the KIS. The KIS now folds calls to the same KDS from multiple ontologies into one call and/ormarshals the inbound and/or outbound results (the marshaling cost is minimal compared to the I/O cost). This (in addition to the parallel pipeline change) resulted in a 4-fold speedup (on one server).

Additional KIS Features. In an exemplary embodiment, some additional admin-controllable features can be added to the KIS as additional KIS Features via KC Properties Dialog Box. The admin can select one of three types of KCs: Standard, Virtual Redirctor, and/or Gatherer. The first refers to a regular KC and/or the second refers to a virtual KC. A virtual knowledge community is a KC that federates other (real) KCs. There are two kinds of virtual KCs: Redirctors and/or Mirrors. A redirector (currently supported) isn’t real at all in that it has no data of its own. It merely reroutes queries from clients to real KCs and/or then merges the results on the fly. So it sits between—and/or “lies to”—both the client (the Librarian) and/or the real KCs. The Librarian thinks it is requesting results from a real KC and/or the real KC(s) think they are responding to the Librarian. As the name implies, a Mirror may be a synchronized copy of other (real) KCs. Mirrors would allow the admin to use some KCs mainly for indexing and/or then mirror the data on those KCs (with much less I/O overhead) to other KCs to be used primarily for query-processing. This model also allows the KIS to scale out as well as up, and/or to support large enterprise and/or online deployments. To avoid complexity and/or (potentially endless) recursion, a virtual KC cannot contain another virtual KC. Else (without very expensive and/or complicated distributed loop-detection), this could potentially result in an infinite request loop. The third option allows the admin to specify that a KC may only be used to gather links based on the specified knowledge sources. This allows the admin to use the KC to say, crawl web sites. The Gatherer KC then generates RSS based on the detecting links. The admin can then use the RSS in different ways: to transform the RSS (as described above), to index the RSS from another KC, etc. The admin can now specify the ID to be used with a newly created KC. This is a powerful feature especially for cases where the KIS database was restored or moved and/or the admin wants to restore the KC to use the same data store (the Semantic Metadata Store (SMS)). The admin can specify (and/or always change) the AliasID for the KC. This is what is used to identify the KC to clients. This is also very powerful because it means that clients don’t need to re-subscribe to the KC if the KC is renamed. Also, if the server is reinstalled (or moved) and/or the KIS is restored, the KC can be recreated and/or set to use the same AliasID as before, thereby keeping the restoration or move process transparent to client subscribers. The admin can now specify whether the KC is to be visible to “standard clients.” “Standard Clients” refers to the end-user semantic client. This feature is useful in cases where the same KIS hosts standard client-accessible KCs and/or KCs to be used solely for the purpose of federation (within a larger virtual KC). However, all KCs remain visible to all other KCs—this allows a virtual KC to be able to point to any standard KC. The admin can specify time-sensitivity settings to indicate how often, on average, the knowledge sources change. In one embodiment, the following settings are available: Everyday (good for busy file-shares and/or high-traffic web sites and/or RSS feeds); Every week (good for weekly publications or not-so-busy content sources); Every two weeks (good for seldom busy content sources); Every month (good for journal publications); Every two months (good for journal publications); Every three months good for journal publications; Never (for archival sources). The admin can specify how often the KC re-indexes the knowledge sources. By default, the KIS recommends re-index frequencies based on the type of content source (e.g., 30 minutes for web sites, and/or 5 minutes for file-shares). The frequency can also change adaptively as the KIS observes the average data change rate. However, the admin can specify a frequency. This is advantageous especially for public web sites that might have specific instructions on how often they are to be visited by crawlers.

User Model for Determining Supported Ontologies. In one embodiment, a user of the semantic client (the Nervana Librarian) has a way of knowing which ontologies a KC
“understands.” Else, it would be very easy for a user to pick categories from one of such ontologies, only to get 0 results. This could lead to user confusion because the user might think there is a problem with the system. To address this: 1. The SRML header may now include a field for “unsupported knowledge domains”—this field may have one or more knowledge domain GUIDs separated by a delimiter. 2. When the KIS receives a request, it may first check whether there are any unsupported knowledge domains in the SQL query arguments—it does this by comparing the domains against the KDS domains it is configured with. If there are unsupported domains, it may populate the field and/or return the field in the SRML response. 3. If the SQL has the AND/OR operator and/or if number of unsupported knowledge domains is equal to the number of categories in the SQL query argument, the server may return an error. If the operator is an OR and/or if the number of unsupported knowledge domains is equal to the number of arguments (categories, keywords, documents, etc.), the server may return an error. If at least one domain is supported, the server may process the request normally—as it does today; as such, the request may succeed but the unsupported field may also be populated. 4. On a per KC basis, and/or on the server, the SRML response, if there is an error (appropriately tagged), the Presenter (in the semantic client) may display the error icon to indicate this. In one embodiment, there is a different icon for this—so the user clearly knows that the error was because of a semantic mismatch. 5. On a per KC basis, and/or on the server, the Presenter may show the results but [also] displays the icon indicating that a semantic mismatch occurred. Perhaps this icon is smaller than the one displayed in #5 above (or has a different color) indicating that the error wasn’t fatal. 6. When the user clicks on the icon, the Presenter may display an error message describing the problem. The Presenter may then call SRAP (the semantic client’s semantic runtime API) with a list of the unsupported domains (retrieved from the SRML header) to get the details of the domains. SRAP may then return metadata on the domains—the Publisher and/or the category folder name—and/or this may be displayed as part of the error message. This way, the user may never see the GUID. 7. The semantic client also allows the user to browse the category folders (ontologies) a KC or profile supports.

[3073] Semantic Sounds. Semantic Sounds are a new feature in line with this model. When in Live Mode and/or when there is Breaking News, the Presenter (in the semantic client) subtly notifies the user of Breaking News by making a sound. This signal is intelligent, based on the semantics of the news request. Here are some variables that affect the kind of sound that gets played: 1. The number of breaking news results—the alert is modulated based on this value (e.g., volume/amplitude, pitch, etc.) 2. How recent the news is (e.g., volume/amplitude, pitch, etc.) 3. How long ago the bell was sounded—similar to how Microsoft Outlook (the email client) only signals new mail after a while (it doesn’t make redundant sounds as new email floods in). Also, in the future, these sound fonts can be extended to be different based on the semantics of the request. For instance, the bell for Breaking News in Aerospace might be the sound of a plane taking off or landing. The bell for Breaking News in Telecommunications might be the sound of ringing cell phones. The bell for Breaking News in Healthcare of Life Sciences might be the sound of a heartbeat. Also, in one embodiment, users would be able to customize and/or personalize Semantic Sounds.

[3074] Ontology Suggestions based on Public Search Engines (or Community Submissions) and/or Typos. An embodiment of the invention uses a synonym suggestion API (from public search engines—like Google Suggest) to suggest word and/or phrase forms for the ontology tool during the ontology development or maintenance process. This way, the system can piggyback on the collaborative filtering of public search engine users and/or their searches. This may be better than using something like Microsoft Word or WordNet which may provide the dictionary’s perspective but not an aggregation of humanity’s current perspective (which is what a good ontology represents). This, for example, may include slang words and/or the like, which we also want.

[3075] As an illustration, visit: http://www.netcaptor.net/adsense/suggest.php

[3076] Type in:

[3077] 1. Storage Area Network

[3078] 2. XML

[3079] 3. XPath

[3080] 4. Web Service

[3081] 5. Semantic Web

[3082] See the alternative forms.

[3083] For instance “Semantic Web” or “Semantic Webbing” (sounds like a slang but is actually a good hook, given current lingo). The app is good at super-phrases that are PROPER phrases AND/OR that BEGIN with the typed word/phrase but does not address super-phrases that END or CONTAIN the typed word/phrase. Note that super-phrases may generally result in less false positives because they are more context-specific. Super-phrases are good to have even when the ontology has exact phrase hooks because without them, the categorizer can get biased by stop words which might be in the super-phrase. With super-phrase hooks, the stop words may have no effect and/or the entire super-phrase may get latched. See the PHP code here for the tool: [http://www.netcaptor.net/adsense/_suggest_getter.txt]. The live Google Suggest application is here: [http://www.google.com/webhp?complete=1&hl=1=en]. Because Google gives us the approximate results count for each suggestion, this is one way to prioritize your suggestions. Also, because Google Suggest only suggests super-phrases, the following preferred embodiment is used: 1. Call the API with the exact word/phrase; 2. Take out one letter. Repeat step 1 above; 3. Take out two letters. Repeat step 1 above; 4. Continue up till 3-5 letters (rough estimate). Repeat step 1 above. For example: calling the API with just “laparoscopy” would miss “Laparoscopic.” However, typing “lapuro” yielded “laparoscopic” AND/OR many more interesting suggestions which are also likely hooks.

[3084] “Laparoscopy” also yielded results and/or is a common typo. Type this in Google, it asks whether you mean “laparoscopy.” To find reverse-recommendations from typos (likely typos, given the phrase): 1. For all vowel letters, take out one vowel at a time and/or call the API (laparoscopy: laparoscopy, laparoscopy, laparoscopy, and/or so on . . . ) 2. For double-letters (e.g., “ll”), take out one letter and/or call the API (e.g., letter>letter) 3. If there is a hyphen (for compound names), take out the hyphen and/or call the API. 4. Launch Microsoft Word 2003 and/or go to Tools>Options. See the autocorrect rule list (that way we piggyback on typo research by Microsoft). Copy the rule list into a data store (like XML) and/or apply these rules. A closely related idea is Community
Watch Lists. This is an offshoot of the Category Discovery feature wherein a Librarian user would have the option of viewing multiple watch lists:

[3085] Personal Watch Lists: My Default Watch List: this watch list may be populated with News Dossiers reflecting the default requests (with no context). My Favorites Watch List: this watch list may be populated dynamically based on the favorites list. My Live Watch List: this list may contain all requests that are currently set to Live Mode (whether or not they are favorite requests); this allows the user to dynamically watch (and/or “un-watch”) Librarian items. My Documents Watch List: this list may be dynamically built based on the categories (for all profiles) that correspond to the user’s local documents, email messages, Web browser favorites, etc. The list may be built by a local crawler and/or indexer which may periodically crawl local documents, email, Web browser favorite links, etc. and/or find the categories by using Dynamic Linking on a per item basis. These categories may then be mapped to SQML and/or used to build this watch list. Community Watch Lists: Recommended Categories Watch List: this watch list may be automatically generated based on Recommended Categories in the user’s knowledge communities (as described below). Popular Categories Watch List: this watch list may be automatically generated based on Popular Categories in the user’s knowledge communities (as described below). Categories in the News Watch List: this watch list may be automatically generated based on Categories in the News, in the user’s knowledge communities (as described below). Community Watch Lists may also be an extremely powerful feature as it would allow the user to track categories as they evolve in the knowledge space, further employing collective intelligence. You can think of this feature as facilitating Collective Awareness. In one embodiment, there may be My Favorites (favorites and/or live) and/or Community Favorites (all the Community watch lists, combined).

[3086] Category Discovery. Category discovery is a new feature of an embodiment of the invention that would allow users discover new categories of interest. Today, while browsing for categories, the user has to know what categories are interesting to him/her. In many cases, this would map to the user’s research interests, job title, etc. However, users occasionally want to find out about new areas. As such, we don’t want a situation where the user remains “stuck in the same semantic universe” without expanding his/her knowledge to additional fields over time. To address this, an embodiment of the invention can perform mining of categories at each KIS. Each KIS may mine: 1. Recommended Categories—these are categories that the system recommends based on the user’s interests and/or queries, and/or the semantic correlation between domains. This may be modeled based primarily on Categories in my Interest Group—these are categories relevant to people in the community that share the user’s interests. Extremely popular categories (even outside my interest group) would also likely qualify. 2. Categories in the News—these are categories that are currently in the news; 3. Popular Categories—these are categories that are popular within a given knowledge community; 4. Best Bet Categories—these are categories that correspond to Best Bets within a given knowledge community. You can think of these filters as forming a Categories Dossier. A special filter, My Categories, is dynamically composed by mining the user’s My Documents folder, local Web browser favorites, local email, etc. The user is able to specify local folders and/or information sources and/or Nervana profiles (all by default) to be used to determine the My Categories list. The semantic client would then periodically invoke Dynamic Linking to determine the user’s category-oriented universe. This is very powerful as it allows the user to automatically determine his/her category universe (based on his/her information history) and/or then be able to use those categories in requests, entities, etc. Other filters can also be added, not unlike a Knowledge Dossier. The Librarian may then allow the user to view the categories dossier from within the Categories Dialog (the dialog may dynamically update the categories from each KIS in the user’s profile(s)). Of course, as is the case today, the user may also be able to view “all categories.”

[3087] This feature may be very powerful. Imagine a new employee of Nervana that joins the company, subscribes to knowledge communities, and/or is eager to learn about various topics relevant to the organization (across context and/or time-sensitivity). Today, the employee would have to know which categories to browse for—likely categories relevant to his/her work. However, with Category Discovery (via a Categories Dossier), the employee may be able to discover new categories as the knowledge space evolves over time. And/or as is the case today, this discovery may be exposed in the context of one or more profiles, which could contain one or more knowledge communities—thereby resulting in Federated Category Discovery. This feature may apply collective intelligence not only to the discovery of documents and/or people but also to categories, which in turn represent an axis of discovery.

[3088] Category Discovery in Deep Info. Category Discovery also provides new “Deep Info portals or entry points.” In one embodiment, the Category Discovery filters are exposed via Deep Info. This is done on a per profile basis. An illustration is shown below:

[+] My Profile
  [+] Recommended Categories
  [+] Cancer
  [+] Amino Acids
  [+] Breaking News

[+] Headlines
  [+] Newsmakers
  [+] All Bets
  [+] Best Bets
  [+] Experts
  [+] Conversations
  [+] Mary Smith
  [+] Joe Johnson
  [+] Interest Group

[+] Breaking News

[+] Newsmakers
[+] Best Bets
[+] Conversations
  [+] Peter Marshall
  [+] Kenneth Falk

[+] Categories in the News
  [+] MeSH
  [+] Cardiovascular Diseases
  [+] Cardiac Failure

[+] Popular Categories
[+] Best Bet Categories
[+] My Categories

[3089] Notice that the user is also (in addition to the discovered category) able to navigate from parents of the dis-
covered categories (since they are also semantically relevant to the context). And/or as described in prior invention submissions, any of these "entity contents" can be dragged and/or dropped, copied and/or pasted, used with the Smart Lens. Legend:

- [3090] Blue: Ontology (Category Folder) for discovered category
- [3091] Red: Parent category for discovered category
- [3092] Green: Discovered category

[3093] Knowledge Community Watch Lists. A closely related idea to Category Discovery is Knowledge Community Watch Lists. This is an offshoot of the Category Discovery feature wherein a Librarian user would have the option of viewing multiple watch lists:

- [3094] Personal Watch Lists: My Default Watch List — this watch list may be populated with News Dossiers reflecting the default requests (with no content); My Favorites Watch List — this watch list may be populated dynamically based on the favorites list; My Live Watch List — this list may contain all requests that are currently set to Live Mode (whether or not they are favorite requests); this allows the user to dynamically watch (and/or "un-watch") Librarian items; My Documents Watch List — this list may be dynamically built based on the categories (for all profiles) that correspond to the user’s local documents, email messages, Web browser favorites, etc. The list may be viewed by a local crawler and/or indexer which may periodically crawl local documents, email, Web browser favorite links, etc. and/or find the categories by using Dynamic Linking on a per item basis. These categories may then be mapped to SQML and/or used to build this watch list. Community Watch Lists: Recommended Categories Watch List — this watch list may be automatically generated based on Recommended Categories in the user’s knowledge communities (as described below); Popular Categories Watch List — this watch list may be automatically generated based on Popular Categories in the user’s knowledge communities (as described below); Categories in the News Watch List — this watch list may be automatically generated based on Categories in the News, in the user’s knowledge communities (as described below); Best Bet Categories Watch List — this watch list may be automatically generated based on Categories that correspond to Best Bets, in the user’s knowledge communities. Knowledge Community Watch Lists may also be an extremely powerful feature as it would allow the user to track categories as they evolve in the knowledge space, further employing Collective Intelligence. You can think of this feature as facilitating Collective Awareness. In one embodiment, there may be My Favorites (favorites and/or live) and/or Community Favorites (all the Community watch lists, combined).

[3095] Part Mutual Cross-Ontology Validation and/or Ontology Development and/or Maintenance Tool Features. In one embodiment, ontologies are developed and/or maintained with the help of ontology development and/or maintenance tools that aid the ontologist by recommending semantic assertions and/or other rules. For example, in one embodiment: Some category labels occur in multiple ontologies. The ontology tool flags the user (the ontologist) when there is a discrepancy. The discrepancy might be valid but might also indicate an incomplete ontology. For instance, Artificial Intelligence occurs in both IT and/or Products & Services but the sub-categories and/or hooks are likely very different. Some of this might be legitimate but some of it might be due to oversight. Similarly, Software occurs in both Products & Services and/or General Reference (ProQuest). Furthermore, hooks that occur in one domain probably allows exclusions in another domain (for instance, hooks for "Virus" in MesH probably allows exclusions that are themselves hooks for "Virus" or "Computer Virus" in IT, and/or vice-versa. And/or so on. You can use the different ontologies to check for cross-domain mismatches of this sort. The inventor calls this Mutual Cross-Ontology Validation. It is an extremely powerful feature. This mutual cross-ontology validation approach may generate a viral network effect and/or positive feedback of ontological quality wherein as ontologies improve, others in the ontology network may also improve, which in turn may subsequently improve other ontologies. Also, hooks that have multiple word-forms probably includes exclusions and/or your tool flags this (not atypicaly, not all word forms applies in the same context). The same for hooks that occur in multiple domains—the cross-ontology validation described above, and/or the invocation of dictionaries like online search engines or tools like WordNet may help a lot here.

[3096] More on Semantic Inference Engine Types and/or Features. As may be described in the co-pending patent applications cited herein, the Semantic Inference Engine (SIE) may constantly be running, especially during the indexing process. The Time-Sensitivity Inference Engine (TSIE) may always be running as long as the service is running (because time "always runs"). The TSIE may determine what is "news-worthy" based on a triangulation of the context of the query (if any), time, and/or semantic strength. In one embodiment, only recommendations ("Good Bets" of strong, albeit not necessarily very strong, semantic density) constitutes news-worthy items (Breaking News or Headlines). However, the semantic query processor involves dynamic context-sensitive ranking such that the best headlines are returned before the next best, etc. This has been previously described but this note is aimed at proving yet another explanation. The SIE is responsible for adding semantic links for categories that are semantically related to categories that are returned during the categorization process. For instance, if the categorizer indicates that a document has the category "Encryption" with a score of 90 (out of 100), the SIE, in addition to creating a semantic link for this category, also creates a semantic link for parents of Encryption (e.g., Security). The SIE also optionally attenuates the scores as it moved up the hierarchy chain. This way, when a user semantic queries for a broad category, semantically related child categories are also found. This was described in the original invention, but this note is aimed at providing a bit more insight. The Adaptive Ranking Inference Engine (ARIE) was described above.

[3097] Semantic Business Intelligence. An embodiment of the invention can be used to provide Semantic Business Intelligence. Today, many Business Intelligence (BI) vendors provide reports on sales numbers, financial projections, etc. These reports typically are akin to Excel spreadsheets and/or usually have a lot of numerical data. One problem many BI vendors have today is that their users wish to ask semantic questions like: "What Asian market is the most promising for our localized products?"? An embodiment of the invention provides the semantic infrastructure to approximate such natural queries. In one embodiment, the System handles this via its Semantic Annotation model, already described in the original invention submission. Business Intelligence Reports would get annotated with natural text and/or the associations are maintained via hyperlinks. An embodiment of the invention then semantically indexes the natural text annotations.
Users then use the semantic client to ask natural questions. An embodiment of the invention returns the text annotations in the semantic client. The user can then interpret the context and/or also navigate to the BI reports via the hyperlinks. This model can be extended to any type of data or information, not just Business Intelligence reports. Audio, video, or any type of data or information can be annotated this way and/or semantically searched and/or discovered via an embodiment of the invention. FIG. 116 shows an illustration of the implementation of the feature, the well-known knowledge stack, and/or how this applies to this model.

Dynamic Ontology Feedback. Another feature of an embodiment of the invention is Dynamic Ontology Feedback. In one embodiment, there may be a button in the semantic client UI to allow the user to provide Nervana (or some third-party ontology intermediary) with ontology feedback via email. That way, our users can help improve the ontologies—since they, by definition, may be domain experts. The button can launch an email client (like Microsoft Outlook) preconfigured with an ontology feedback email address and/or a feedback form including the name of the ontology, the domain id, the request that triggered the response, the problem statement, etc. This can then feed to ontologies for processing and/or direct ontology improvement. In one embodiment, the semantic client may auto-fill the ontology feedback form with the details indicated above (since the semantic client may have that information on the client)—the user does not need to fill in anything. Also, ideally, there is a privacy statement for this so users can have the comfort that we are not sending any personal information back to Nervana or some third-party.

More on Dynamic Linking. One scenario that represents a common query in Life Sciences is the following: How does one find all proteins from Protein Database P relevant to abstracts on Inhibitor I found in the Medline database M? As previously described, the technology to enable this scenario, Dynamic Linking, is the essence of the invention. In Nervana, Dynamic Linking may allow the user to navigate across semantic (and/or ontological) boundaries at the speed of thought. This is what, like Knowledge itself, may make the system achieve a state of Endlessness—turning it into a true Nervous System. Drag and/or Drop, Smart Copy and/or Paste, the Smart Lens, Deep Info, etc. are some of the visual tools that may be used to invoke Dynamic Linking. In an embodiment the semantic client allows the user to drag a chemical compound image to Medline, find a semantically relevant abstract in Best Bets, copy a subscribed Protein Database KC (likely from a different profile) as a Smart Lens (via the Semantic ClipBoard), hover over the Medline abstract using the Protein Database as the Smart Lens, and/or open a Dossier on the Medline abstract from the Protein Database on the chemical compound that initiated the [Semantic] Chain Reaction. By breaking up the problem into contextual subproblems, Dynamic Linking allows the user to express semantic intent across contextual (and/or knowledge-source) boundaries ad infinitum. The system is then able to “answer” a complex question like the one above—the “question” is interpreted as a chain of smaller questions.

Handling Floating Text and/or Signaling in MS Connectors and/or Data Source Adapters. As described in the KIS Connector Specification, RSS is used to abstract out different data sources (via DSAs that return RSS). In many cases, the information items to be indexed might not have any stored documents—they might be “floating text” (e.g., from databases that contain the item’s text). In such a case, the DSA generates RSS with a Nervana-namespace qualified tag that indicates this. In one embodiment, this tag is called “nofollow.” Other uses for this are for cases where the KIS cannot index the full documents (when they do index) for administrative or business purposes. For example, the NIH web site typically forwards crawlers from indexing Medline documents. This feature would allow the metadata to be indexed even if the full documents can’t be indexed. The sample RSS from an embodiment of the Medline metadata DSA below illustrates this (the Nervana namespace is titled “meta”):
an embodiment of the invention to answer questions like: 1. What is the population of Norway? 2. Which country has the largest GDP in the European Union? A Natural-Language-Processing engine is described in at least one of the co-pending applications cited herein. In one embodiment, a Q&A layer is built on top of the Knowledge Integration Service (KIS) semantic query layer. Per the semantic query layer, for instance, a document that describes the population of Norway somewhere in its contents would get surfaced by the semantic engine in an embodiment of the invention. No additional annotations might be needed. Also, even if the factoid is written as "the number of people that live in the second largest Scandinavian country, an ontology that describes population and/or describes countries (in as many ways possible) would lead this factoid to be surfaced by an embodiment of the invention. This Q&A layer goes further and/or exposes specific answers as factoids. The Q&A layer involves annotating documents that are semantically indexed by the KIS. These annotations expose "facts" from text. These facts would then have schemas like People, Places, Things, Events, Numbers, etc. This may be an extension of the knowledge-stack model described above. The “factoids” may be akin to the Business Intelligence reports described above. Factoid reports with specific schemas may be annotated with natural text (and/or connected via hyperlinks). The semantic query layer in an embodiment of the invention would allow the user to retrieve the annotations. Once the user retrieves the annotations, the user may be able to view the factoids via hyperlinks. This model also allows multiple factoid perspectives to be exposed off the same document(s). This is extremely powerful and/or much richer than standard Q&A approaches that directly expose facts (while perhaps hiding other important viewpoints off the same document base).

[3103] Semantically Interpreting Natural Language Queries. At the beginning of at least one of the co-pending applications cited herein, I asserted that the notion of natural-language queries as the nirvana of information retrieval is wrong. I pointed out that discovery of knowledge, incorporating context-sensitivity, time-sensitivity, and/or occasional serendipity is instead possible. However, having the simplicity of natural language queries as AN OPTION (drag and/or drop and/or other semantic tools are arguably more powerful in many contexts), WITHOUT the limitations of natural-language interpretation, is also possible. In other words, natural-language queries but NOT natural-language interpretation—rather, natural-language queries coupled with semantic interpretation in an embodiment of the invention. The power of coupling these is that the user can gain the simplicity of natural expression without losing the power of semantic discovery and serendipity. In one embodiment, the natural-language-query interpretation involves mapping the query to a Nervana semantic query. An NLP plug-in is added to the semantic client to do this. This plug-in takes natural-language input on the client and/or maps these to semantic input (SQML) before passing the query to the server(s) for semantic interpretation. The NLP component parses the natural-language text input and/or looks for key phrases using a standard key phrase extractor. The key phrases are then compared against the ontologies supported by the query profile. If any categories are found using direct, stemmed, and/or fuzzy matching, these categories are added to the semantic query as candidates. Key phrases that aren’t found in the ontologies are proposed as keywords and/or stemmed variants are also proposed (and/or ORed in the SQML entry). The final candidates for semantic queries are then displayed to the user as recommended queries. The user can opt to choose one or more queries he/she finds consistent with his/her intent, or to edit the queries and/or then accept them. The accepted query (or queries) is then launched. This conversational model is very powerful because the reality is that the user might have a lot of background knowledge that would aid his/her interpretation of the natural-language query and/or which an embodiment of the invention would not have. The reasoning system may be unable to always pick the right context and/or the ontologies might not capture the background knowledge. Background, experience, and/or memory also constitute context. And/or without “knowing” this, an embodiment of the invention may not do its job properly for arbitrary natural-language queries. As such, the conversational model allows an embodiment of the invention to propose semantic queries and/or then the user can then apply his/her background knowledge, experience, and/or “outside content” to further refine the query. This is a win-win. Examples of natural-language queries with corresponding semantic queries are: 1. Develop a genetic strategy to deplete or incapacitate a disease-transmitting insect population (from the Gates Foundation Grand Challenges on Human Health), Dossier on Genet- ics (MeSH) AND/OR Diseases or Disorders (CRISP) AND/ OR Insects (MeSH) AND/OR ‘transmit or transmits or transmission or transmissions or transmitting’; 2. What is the cumulative effect of multiple pollutants on human health? (see http://www.tceq.state.tx.us/RFPS/Final_Reports/Bridge/Final%20Report.pdf); Dossier on Environmental Pollution (MeSH) AND/OR Public Health (MeSH); 3. What is the effect of pollution on learning in children? Dossier on Environmental Pollution (MeSH) AND/OR Learning Disorders (MeSH); 4. Are there cancer clusters in the Houston-Galveston area? All Bets on Neoplasm and/or Cancer (CRISP) AND/OR ‘Houston Galveston area’; 5. What are the long-term effects of fine particulate pollution on children?; Dossier on Pollutant (Cancer (NCI)) and/or Children (Cancer (NCI)); 6. How can we reduce exposure to pollution? Recommendations on Environmental Exposure (MeSH) and/or ‘reduce’; 7. What is the role of genetic susceptibility in pollution-related illnesses? Dossier on Diseases and/or Disorders (CRISP) AND/OR Environmental Pollution (MeSH) AND/OR Genetics (MeSH); 8. The full list of all Gates Foundation Grand Challenges on Human Health can be found at: http://www. grandchallenges.org/challenges.aspx?SecId=258. Here is the full list (these examples highlight the power of the Information nervous System and/or how keywords are completely ineffective): 1. Create effective single-dose vaccines that can be used soon after birth; 2. Prepare vaccines that do not require refrigeration; 3. Develop needle-free delivery systems for vaccines; 4. Devise reliable tests in model systems to evaluate live attenuated vaccines; 5. Solve how to design antigens for effective, protective immunity; 6. Learn which immunological responses provide protective immunity; 7. Develop a genetic strategy to deplete or incapacitate a disease-transmitting insect population; 8. Develop a chemical strategy to deplete or incapacitate a disease-transmitting insect population; 9. Create a full range of optimal, bioavailable nutrients in a single staple plant species; 10. Discover drugs and/or delivery systems that minimize the likelihood of drug resistant micro-organisms; 11. Create therapies that can cure latent infections; 12. Create immunological methods that can cure chronic infections; 13. Develop technologies that permit quantitative assessment of population health sta-
Develop technologies that allow assessment of individuals for multiple conditions or pathogens at point-of-care; take as an example challenge #7: Develop a genetic strategy to deplete or incapacitate a disease-transmitting insect population. With this multi-dimensional (multiple-perspectives) query, the difference in relevance between an embodiment of the invention and/or standard (non-semantic) approaches grows by orders of magnitude. Genetics is a huge field, there are many types of diseases, and/or there are many types of insects. And/or then to rank and/or group the results multidimensionally is extremely complex mathematically. An embodiment of the invention does this automatically.

Request Collections with Live Mode: Live Mode has already described in details in at least one of the co-pending applications cited herein. This is just a note to qualify how Live Mode works with Request Collections (Blenders). When a Request Collection is in Live Mode, all its requests and/or entities, are presented live when the request collection is viewed. In one embodiment, the request and/or entities are not automatically made live themselves (if they are not live already). Only when the request collection is displayed are the requests viewed live (with awareness—ticker animations, etc. showing Breaking News, Headlines, and/or Newsmakers, etc.). A skin can elect to merge the results of a Request Collection so that only one set of live results may be displayed. Other skins might elect to keep the individual request collection entries viewed separately in Live Mode.

Adapting to Weak Categorization in Non-Semantic Context Templates. In some cases, some key phrases might not get detected in the categorizer, especially if the lexicon for the categorizer has not been seeded with the terms in the ontology. Typically, with rich enough context, this is not an issue because there is a high likelihood that terms in the ontology may already lie within key phrases. However, with short documents or abstracts, this might not happen because there might not be enough context. In this case, the ontology-independent concept extraction model can lead to weak categorization. To handle this, the categorizer is seeded with a lexicon corresponding to the terms in the ontology. This ensures that the categorizer, during the concept extraction phase, "knows" to return certain concepts based on the contents of its lexicon (now domain-specific). Furthermore, the KIS when interpreting semantic context with non-semantic context templates (like All Bets and/or Random Bets) AND/OR for a non-semantic ranking bucket (bucket #0), maps the category URI in the incoming SQML to keywords and/or include the keywords in the SQML resource inner join. This is powerful as it ensures that even if the categorization failed, the keyword that corresponds to the category name may result in a hit. There is a loss of semantics in moving to keywords but because the context template is All Bets or Random Bets AND/OR because the ranking bucket is non-semantic, this doesn’t matter. This improves recall by dynamically adapting to a lack of context at the categorization layer.

Dynamic Linking Rules in the Server-Side Semantic Query Processor. The end-to-end architecture of Dynamic Linking (most typically invoked via Drag and/or Drop) has already been described in detail in at least one of the co-pending applications cited herein. This note is to clarify the supporting server-side implementation in the semantic query processor (SQP). At a high level, the philosophy of Dynamic Linking is that the system determines what the dragged is about and/or semantically retrieve items, in the context of the template of the dropped, from the source represented by the dropped. Once the semantic client retrieves the key concepts from the dropped (as has been previously described), it passes the metadata to the server(s) (possibly federated). Each server then asks the KDSes if it is configured with to categorize the context. In an alternative embodiment, the client can directly contact the KDS to categorize the context and/or then pass the categories to the servers. The client has a concept extraction cache so it doesn’t have to always extract concepts if the user repeats a query. And/or the server has a concept-to-categories cache (which it periodically purges) and/or use a Reader-Writer lock to maximize concurrency (since multiple client connections would be sharing the cache). The server then maps the weights in the categories to Best Bets, Recommendations, or All Bets, consistent with the weight ranges heuristics described in Part 6 above. The following rules are then applied in dynamically creating semantic queries in a semantic query chain (as described in at least one of the co-pending applications cited herein):

1. Query 1: For each Best Bet category in the source (if any), create a query with an AND/OR of all the categories; 2. Query 2: For each Recommendation category in the source that is NOT a Best Bet, create a query with an AND/OR of all the categories; 3. Query 3: If Query 1 had more than 1 category (i.e., if there was an AND/OR), for each Best Bet category in the source, create N queries with each category; 4. Query 4: If Query 2 had more than 1 category (i.e., if there was an AND/OR), for each Recommendation category in the source, create N queries with each category; 5. Query 5: For each Best Bet category in the source (if any), forward-chain by 1 up the hierarchy in the ontology corresponding to the category, and/or create a query with an AND/OR of the parent (forward-chained) categories. For instance, if there was a Best Bet on Encryption, forward-chain to the parent Security (in the same ontology) and/or AND/OR that with the other Best Bet parents. Check for (and/or elide as necessary) duplicates in case Best Bet categories share the same parent(s).

NOTE: This rule entry may widen the scope of the semantic mapping. This is extremely powerful as it provides discovery (subject to semantic distance) in addition to precise semantic mapping. In one embodiment, forward-chaining is only invoked if there are multiple unique parents. This is critical because ontologies are arbitrary and/or the KIS has no way of "knowing" whether even a semantic distance of 1 is "too high" for a given ontology (i.e., whether it may lead to semantic misinterpretation). In one embodiment, the threshold can be increased to 2 for Best Bets because there is a correlation between semantic strength and/or the probability of semantic distance resulting in false positives. In other words, Query 5 can then be repeated with a forward-chain length of 2 for Best Bets; 6. Query 6: For each Recommendation category in the source (if any) that is NOT a Best Bet category, apply the equivalent of Query 5. In one embodiment, the semantic distance threshold for forward-chaining with Recommendations (less semantic strength than Best Bets) is 1; 7. Query 7: For each All Bets category in the source that is NOT a Best Bet OR a Recommendation, create a query with an AND/OR of all the categories ONLY if there are eventually multiple unique categories (since All Bets also incorporates very low semantic density); 8. Query 8 (optional): If the source has less than N (configurable, 3 in one embodiment) keywords, add a keyword search query (since this would likely correspond to vacuous context that would then lead to weak mapping in Queries 1 through 7 above).
Lastly, the dynamically generated semantic queries are triangulated with the destination context template (Best Bets, Recommendations, etc.), and/or invoked using the sequential query model (previously described), with duplicate results eventually elided. The triangulation with the destination context template imposes yet another constraint to ensure that the uncertainty of the mapping rules are “contained” within the context of the destination template. So the context template eventually “boils out” the semantic and/or mathematical mapping from the “perils of uncertainty and/or complexity.” This is extremely powerful from both a mathematical and/or philosophical standpoint as it reduces an extraordinary complex mathematical space into discrete blocks and/or simultaneously honors the semantics of the query at hand. In one embodiment, the ontologies can also be annotated with hints indicating the how the Inference Engine in the KIS forward-chains to parents when performing Dynamic Linking. This may partially address the arbitrary semantic distance issue because the ontology author can indicate the level of arbitrariness for specific category nodes in the ontology. It wouldn’t fully address the issue though because the arbitrariness might depend on the context of the semantic query, and/or this may not be known at ontology-authoring time.

Dynamic Client-Side Metadata Extraction for Dynamic Linking. As described in at least one of the co-pending applications cited herein, when an object (like a local or Web document or floating text) is dynamically linked on the semantic client, the conceptual (ontology-independent) metadata of the object is extracted and/or then sent to the federated KIS servers for dynamic semantic processing and/or mapping. However, in some cases, the full metadata for the “dropped or pasted object” might not be available to the semantic client at Dynamic Linking invocation time. A good (and/or common) example is a URL that is dynamically generated from metadata but which (at the presentation layer) does not contain all the metadata that might be semantically important. If the semantic client uses the presentation-layer data for Dynamic Linking, this might result in a loss of relevance because the client may not be employing all the metadata that corresponds to the object. To address this, in one embodiment, the System supports Dynamic Metadata Extraction (DME). There are two possible models:

1. Specified metadata per object: In this model, the KIS semantic index (the Semantic Metadata Store (SMS)) has a URL to an object (likely XML) that represents the metadata for each item in the index. This URL is then sent to the semantic client as part of SRML (via the SourceMetadataUri field, complementing the SourceUri field—which points to the object itself). The XML, in one embodiment, is in the SRML schema. When the object is then dragged and/or dropped (or copied and/or pasted or any other Dynamic Linking visual tool), the semantic client then extracts the aggregate metadata by accessing the object referred to via the SourceMetadataUri field. This aggregate metadata is then used for Dynamic Linking—as it represents the structured metadata for the object. In one embodiment, the aggregate metadata constitutes the coupling of the object (e.g., the contents of a document) itself and/or the metadata of the object. However, this model applies to objects that come from a KIS semantic index (i.e., objects that are SRML results).

2. Metadata Extraction Web Service (MEWS): In this model, the semantic client dynamically retrieves the metadata for an object by passing the URI (or contents, or hash, or concepts) of the object to a Metadata Extraction Web Service (MEWS). The MEWS then returns the SRML for the object from a Metadata Mapping Store (MMS). The MMS is maintained by the MEWS (and/or updated by an administrator) and/or maps an object to its metadata. The URL to the MEWS is configured at the KIS (for results that come from KISes) or at the semantic client (via Directory infrastructure—where the MEWS is a central content-management repository that is managed for a group of users).

Smart Browsing. Smart Browsing refers to a feature of an embodiment of the invention that piggybacks on the Dynamic Linking infrastructure already described in at least one of the co-pending applications cited herein. This is what I call the “Too Many Links” Problem. As I described in at least one of the co-pending applications cited herein, this arises from the lack of semantic intelligence in the World Wide Web platform. As information volumes continue to explode, there may be “too many links.” There is simply no way users may be able to navigate all the links that they would see in web sites as they browse. Smart Browsing is an application-layer feature that employs Dynamic Linking (in an embodiment of the invention) to specifically address this problem. With Smart Browsing, the semantic client would allow the user to load a Web page within the context of a System user profile. This then “places the Web page in context.” The semantic client already hosts a Web browser so loading a Web page would piggyback on this. When a Web page is loaded with Smart Browsing, the semantic client then invokes Dynamic Linking for the links on the Web page. It asks all the Knowledge Communities (KC) in the selected profile to dynamically group the links. The KCs then return XML metadata indicating whether each link is a Best Bet, Recommendation, etc., based on the ontologies configured with the KCs. Furthermore, the XML metadata includes ranking information based on the ranking information that comes from the KISes’ configured KDSes. The smart client then annotates each link (perhaps with different hyperlink colors, balloon pop-ups, etc.) with whether the link is a Best Bet in the context of the profile, a Recommendation, etc. In one embodiment, the semantic client might also rank each link based on the contextual semantic strength. This allows the user to know how to invest his/her time—by perhaps viewing the most important pages first, FOR THE SPECIFIED PROFILE. So the user can then view the same web page in different profiles and/or view the page differently with different contextual rankings per links. This is extremely powerful.

More on Client-Side Knowledge Communities. As described in at least one of the co-pending applications cited herein, I described client-side knowledge communities that would provide the user to ability to semantic search and/or discover knowledge from local information sources. This note is aimed at some added clarification: ALL the features of a server-side knowledge community would apply with a client-side knowledge community. Semantic processing of email, for instance, would employ the same model as previously described in the original invention submission. The same applies for all the context templates. For instance, the user may be able to find experts on specified context from his/her local email. The semantic processor would infer experts in the SAME WAY as with a server-side knowledge community.

Another Perspective on Experts, Newsmakers, and/or Interest Group Context Templates. An interesting way of thinking about Experts is as “Best Bets on the People Axis.”
And/or Interest Group corresponds to “Recommendations on the People Axis.” And/or Newsmakers are “Headlines on the People Axis.” In one embodiment, “People” isn’t viewed (semantically) as being radically different from “documents.” The Semantic Inference Engine (SIE) employs these philosophizations to provide a clean and/or logically coherent implementation of these context templates.

[3116] Intra-Entity Exploration in Deep Info. In at least one of the co-pending applications cited herein, I described how Deep Info would allow the user to semantically explore the knowledge space from any point of context. Entities are one such point of context. In one embodiment, Deep Info also applies to the contents of an entity (if any). For example, a “meeting entity” might have as its contents the participants of the meeting, the topics that were discussed during the meeting, the documents that were handed out during the meeting, etc. Intra-Entity Deep Info would allow the user to navigate within the entity and/or explore from there, in addition to navigating from the entity. And/or as described in at least one of the co-pending applications cited herein, any of these “entity contents” can be dragged and/or dropped, copied and/or pasted, uses with the Smart Lens, etc.

[3117] Ontology (Category Folder) Add-Ins. Ontology (Category Folder) Add-Ins is a powerful feature of an embodiment of the invention that allows the user to “plug in” a new ontology at the semantic client, even if that ontology was not installed with the client. This may be especially valuable in organizations that have their own private (or community) ontologies. In such cases, these ontologies may not come installed with the product.

[3118] The semantic client provides the infrastructure for Category Folder Add-Ins. An add-in is represented as an XML data blob as shown below:

```xml
<XmlVersion="1.0" encoding="utf-8" />
<yinflant>
  <addin>
    <domainID>36885f533-80d4-4920-8c8f-c0d6f153239</domainID>
    <knowledgeDomain>Onvia.COM/Onvia</knowledgeDomain>
    <publisherName>Onvia/publishername</publisherName>
    <creator>Onvia/creator</creator>
    <category/folderDescription>categoryfolderdescription</category/folderDescription>
    <areaOfInterest>
      <areaOfInterest>Products &amp; Services/Products/areaOfInterest
        &lt;areaOfInterest>Products &amp; Services/Services/areaOfInterest
          &lt;areaOfInterest>
            <taxonomynr>/mos1/myspace/Onvia.txt</taxonomynr>
            <version>1.0</version>
            <language>en</language>
          </areaOfInterest>
        </areaOfInterest>
      </areaOfInterest>
    </category/folderDescription>
  </addin>
</yinflant>
```

[3119] The XML file can contain multiple add-ins. An add-in has the following schema properties: DomainID: This uniquely identifies the ontology that corresponds to the add-in; KnowledgeDomain: The knowledge domain (virtual URI) for the add-in; PublisherName: The entity that published the add-in; Creator: The entity that created the add-in; CategoryFolderDescription: A description of the ontology or category folder; AreaOfInterest: The general areas of interest of the ontology or category folder; TaxonomyURI: A URL to the taxonomy file containing a list of paths to be used while displaying the taxonomy for the ontology in the Categories Dialog; Version: The version of the ontology or category folder; Language: The language of the ontology or category folder.

[3120] The semantic client exposes a user-interface to allow users to dynamically install or uninstall an add-in. The administrator (likely the publisher of the ontology) can publish the add-in XML file to a Web site or file share. Users can install the add-in from there. When an add-in is installed, the semantic client downloads and/or caches the taxonomy file (for quick lookup during category browsing), and/or also registers the metadata in a local Ontology Metadata Store (OMS). This can be implemented via the System Registry. The user can then use the ontology pass though it came with the product. The ontology can then be later uninstalled.

[3121] Boolean Keyword, Category, and/or Field-Specific Specifiers and/or Interpretation. In one embodiment, a system supports field-specific searches to supplement keyword searches. Examples are:


[3123] The KIS simply supports this with field-specific predicates (e.g., PREDICATETYPE?PREDICATE?AUTHOREDBY, PREDICATETYPE?PREDICATE?PUBLISHED?DIN, etc.). This is already in the model, as described in at least one of the co-pending applications cited herein. Additional predicate types can be added to support schema-specific field filters (as described in at least one of the co-pending applications cited herein). The KIS Semantic Query Processor (SQP) then checks keywords for each field-specific annotations. If these exist, the specific predicate corresponding to the field is chosen in the inner sub-query. Else a more generic predicate (or a union of all keyword predicates) is chosen. Furthermore, categories can also be expressed using this model. Examples are:


[3125] Cancer: “Tyrosine Kinase Inhibitor”

[3126] The KIS similarly maps these to category predicates using the appropriate category URI, based on the ontology specified in the annotated keyword. An embodiment of the invention may also allow the user to specify cross-ontology categories. For example, the specifier “Apoptosis may be mapped by the KIS to the semantically densest category (best-performing) or ALL categories with that name (highest relevance), depending on admin settings. This is very powerful as it provides better discovery and/or semantic relevance by looking at multiple ontologies simultaneously. Lastly, these specifiers can be combined using Boolean logic. One example is listed above: PubYear:1970-1975 OR PubYear: 1980-1985 OR PubYear: 2000-2005 (anything published between 1970 and/or 1975, between 1980 and/or 1985 or between 2000 and/or 2005). Any of the specifiers can be combined (keywords or categories). So a user can write PubYear:1970-1975 OR MeSH:Cardiovascular Diseases OR Cancer:Tyrosine Kinase Inhibitor OR “Apoptosis (anything published between 1970 and/or 1975, or about Cardiovascular Diseases in MeSH or about Tyrosine Kinase Inhibitors in Cancer or about Apoptosis in all supported ontologies). An
intersection (AND/OR) can also be specified as an AND/OR NOT and/or other Boolean logic specifiers. The KIS simply maps these to either sequential sub-queries for logical consistency (as previously described) or to a broader SELECT statement in the OBJECTS table before the inner join—typically using the IN keyword (multiple specifiers) instead of the operator (single specifier).

[3127] Uncertainty, Mathematical Complexity, and/or Multi-Dimensionality. In at least one of the co-pending applications cited herein, I contrasted an embodiment of the invention from the Semantic in numerous ways. One of these ways was the requirement of tagging in the Semantic Web. In my comments, I placed a lot of emphasis on the “need for discipline” on the part of the authors, arguing that this model (tagging) could not scale. I maintain this is merely writing to buttress my original argument. In addition to the “need for discipline,” the Semantic Web approach also fails to take into account the inherent uncertainty in many semantic assertions. Many assertions may be probabilistic and/or the probabilities may be conditional probabilities that are themselves dependent on context. And/or such context is typically chained to more contexts. As such, the requirement of tagging in an environment of uncertainty (dealing with human expression) is impractical at scale. Indeed, “uncertainty” is why the word “Bet” is used a lot in the Information Nervous System. The system is built to assume (rather than avoid) uncertainty. Furthermore, there is the element of mathematical complexity in the tagging process. Let us take an example research question listed above: Develop a genetic strategy to deplete or incapacitate a disease-transmitting insect population. With an embodiment of the invention, the user may be able to approximate this question with the semantic query: Dossier on Genetics (MeSH) AND/OR Diseases (CRISP) AND/OR Insects (MeSH). And/or one of the entries in the Dossier is Best Bets on Genetics (MeSH) AND/OR Diseases and/or Disorders (CRISP) AND/OR Insects (MeSH). If one is to ask humans to manually tag the most semantically relevant across all three dimensions specified in the query, and/or against millions or billions of documents (and/or incorporating uncertainty and/or multi-dimensionality), the impracticality of tagging from a mathematical complexity perspective becomes even more evident.

[3128] Viewing Knowledge Community Statistics in the Semantic Client. An embodiment of the invention now allows the user to view Knowledge Community (KC) statistics from the semantic client. The KIS exposes a Web Service API to query statistics. The semantic client calls this API in response to a UI invocation on a per-KC basis. Statistics include the results count per context-template. Additional statistics can be added. The Information Overload Crisis. More data has been generated between 1999 and/or 2002 than that generated in all of the pharmaceutical industry's history.” (Source: DrugResearcher.com); 903,652 new/modified Medline abstracts in 2005 alone (~7000/day). Information doubling yearly (Forrester, U.C. Berkeley); Increasing data fragmentation: virtual, distributed, global research and/or development; numerous data sources; Semantic complexity and/or fragmentation, increasingly complex vocabulary, new gene names, compound names; arbitrary naming schemes; fragmented vocabularies. “The problem is that data is trapped in hierarchical silos, restricted by structure, location, systems and/or semantics. The situation has become a data graveyard.” *Sheryl Torr-Brown, Head of Knowledge Management and/or Technology, Worldwide Safety Sciences at Pfizer.

Knowledge, not information, is what drives productivity. One definition of knowledge is “information infused with semantic meaning and/or exposed in a manner that is useful to people along with the rules, purposes and/or contexts of its use.” Search engines lack semantics and/or context and/or are unequipped to handle information overload. The problem with search is:

[3129] Goal should be search-discovery
[3130] “I don’t know what I don’t know”
[3131] Contextual guidance
[3132] Search along multiple contextual axes
[3133] Semantics, time, context, people
[3134] Search across semantic boundaries
[3135] Physical and/or semantic fragmentation
[3136] A lot of research is inter-disciplinary
[3137] Nervana formulation:
[3138] Search engines search for (information)
[3139] Goal should be to find K (Knowledge)
[3140] Sample Research Questions (Gates Foundation Grand Challenges in Human Health) include: Develop a genetic strategy to deplete or incapacitate a disease-transmitting insect population; Develop a chemical strategy to deplete or incapacitate a disease-transmitting insect population; Create a full range of optimal, bio-available nutrients in a single staple plant species; Discover drugs and/or delivery systems that minimize the likelihood of drug resistant micro-organisms, (Texas Council of Environmental Technology). What is the role of genetic susceptibility in pollution-related illnesses? Which clinical trials for Cancer drugs employing tyrosine kinase inhibitors just entered Phase II? What are my top competitors doing in the area of Cardiovascular Diseases? Patents, News, Press Releases, etc.? Find the top experts researching Genes relating to Mental Disorders.

[3141] An embodiment of the invention solves this problem by way of different contextual axes: Common but different scenarios, Examples: All Bets, Best Bets, Breaking News, Headlines, Recommendations, Random Bets, Conversations, Annotated Items, Popular Items, Experts, Interest Group, and/or Newsmakers. Special Knowledge Filter: Dossier. Filter of filters. E.g., Dossier on Cardiovascular Disorder: Breaking News on Cardiovascular Disorder; Experts on Cardiovascular Disorder, etc. Since filtering is on multiple axes, ranking can be “good enough.”

[3142] Mathematical complexity, uncertainty in ontological expression, imperfect ontological context, multiple semantic paths, probabilistic but sufficiently different to be valuable, navigating knowledge filters—navigating knowledge. The problem with keywords is they are a very poor approximation of semantics. Poor precision and/or recall. “Cancer”=disease, public policy issue, genetics? “Cancer”=Adenoma, carcinoma, epithelioma, mesothelioma, sarcoma?

[3143] For example, suppose you want to find all papers on Cancer written by Nobel Prize winners. Not search for “cancer” and/or nobel prize should return articles on carcinoma by Lee Hartwell (2001); articles on sarcoma by Peter Medawar (1960). Multi-dimensional precision and/or ranking. Best results in multiple dimensions. Another example would be, “Find all papers on Cardiovascular Disorder and/or Protein Engineering and/or Cancer,” not a search for “cardiovascular disorder” and/or “protein engineering” and/or “cancer” should include: technical articles on Hypervolemia and/or Amino Acid Substitution and/or Minimal Residual Disease, etc. Recall divergence increases EXPONENTIALLY with query complexity.
The problems with other forms of context are that keywords are not enough. Topics, documents, folders, text, projects, location, etc.; contextual combinations. Examples include: Find all articles on Cell Division (topic); Find Experts on this presentation (document); Find all articles on Cell Division (topic) and/or “Lee Hartwell” (keywords); Neuriva formulation: K(X), where K is knowledge and/or X is context (of varying types); Context-sensitive ranking on X by K. Google’s™ mines Hypertext links to infer relevance. “PageRank” is a very clever technique, effective enough for large-scale Hypertext Web, but no context. Articles on Cancer by Nobel Prize winners is not Popular Pages="cancer"+"Nobel prize". Popular garbage is still garbage. PageRank relies on the presence of links and/or most enterprise documents do not have links, for example: Adobe PDF, Microsoft Office documents, content management and/or popularity is only one axis of relevance. Google’s™ relies on a centralized index. The knowledge is fragmented, security silos, semantic silos.

Nervana formulation: K(X) from S1 . . . Sn, where K is Knowledge, X is polymorphic context, and/or Sn is a semantically-indexed knowledge base; Context-sensitive ranking on X, by K.

The Problem with “Natural Language” Search. Search vs. Discovery Language interpretation is NOT the same as semantic interpretation, it does not address multiple forms of context. The problem with Directories and/or Taxonomies, 1:1 vs. 1:many, documents to topics; single vs. multiple perspectives, Static vs. dynamic; Research often crosses domain boundaries; Nervana formulation: Natural-language Q&A, flexibility without natural-language queries, K(X) from S1 . . . Sn, where K is Knowledge, X is polymorphic and/or dynamically combined context, and/or Sn is a semantically-indexed knowledge base; Context-sensitive ranking on X, by K. More metadata and/or semantic markup, RDF. Ontologies: OWL. Problems include reliance on formal markup and/or metadata, impractical at scale; expressing uncertainty; conditional Probabilities? Mathematical complexity and/or multi-dimensionality: absence of context at markup time; Limitations of human expression; does not address hard problems of semantic indexing, filtering, ranking, and/or user-interface. Most knowledge-related questions are semantic not structural, witness Google’s™ success (no reliance on structure). Multiple perspectives of meaning find all articles on Cancer written by Nobel Prize Winners. Question crosses “semantic boundaries”, notion of a formal “Web”, “Web” is author-centric, not user-centric, Navigation should be dynamic (across silos); “Web” should be virtual. For example, “navigation” from local document to Experts on that document. Semantic query processing; Across ontology boundaries; Context-sensitive; Semantic dynamics; Semantic user interface; Multiple schemas; Flexible knowledge representation; Integrated data model; Domain-specific and/or domain-independent; Inference and/or reasoning. The Nervana Knowledge Domain Service (KDS). Dynamic ontology-based classification.

The Nervana Knowledge Integration Service (KIS). Semantic indexing and/or integration; does not require semantic markup; exploits structured metadata if available; multiple distributed ontologies; separates data from semantic interpretation; multiple perspectives; inference and/or Reasoning Engine; dynamic linking (semantic dynamism); semantic user experience without needing a Semantic Web. See, for example, FIGS. 108 and/or 110. The Nervana Librarian (Semantic User Interface) features User Intent, Context and/or semantics, Time-sensitivity, Discovery, Multiple knowledge axes, Semantic cross-fertilization, Personalization, Federation, Other: Awareness, Attention-management, Dynamic follow-up and/or drill-down, Seamless integration with context and/or workflow, Discoverability of knowledge, Knowledge capture and/or sharing and/or context sharing and/or collaboration. See FIG. 109. K(X) from S1 . . . Sn, where K is Knowledge, X is polymorphic and/or dynamically combined context, and/or Sn is a semantically-indexed knowledge base; Multi-dimensional, context-sensitive ranking on X, by K. Implications: Knowledge filters+semantic user interface+dynamic semantic indexing and/or query processing=approximation for natural-language queries.

Triangulation of knowledge filters+context+sources=semantic approximation. For example: Find all articles on Cancer written by Nobel Prize Winners. Dossier on Cancer (Life-Sciences ontology) AND/OR Nobel Prize Winners (General Reference ontology). Knowledge filters soften impact of imperfections in predicate interpretation, ontologies, and/or categorization; For example, “By” vs. “On”; Filters provide diverse and/or approximate semantic paths.

There is increasing pressure on the industry to improve R&D ROI, one major cause: Information Overload. Limitations of current solutions are: Knowledge vs. Information, Search vs. Discovery, Context and/or Semantics. Introduced the Nervana System (the Information Nervous System) which includes end-to-end knowledge medium; context, semantics, dynamic linking, a semantic user interface, a semantic user experience without semantic markup or a Semantic Web; approximation for natural-language queries (with Discovery and/or without its limitations).

Category result (via ontology) returned by KDS:

- Name: Cardiovascular Disorder Epidemiology
  - URI: nerc://76331eb3-e494-45b5-8939-a4dbb08808e4#type=category&path=Biology/Ecology/HumanEcology/HumanPopulationStudy/Epidemiology/CardiovascularDisorder
  - Weight: 0.431

Category object schema:

- Name: Cardiovascular Disorder Epidemiology
  - URI: nerc://76331eb3-e494-45b5-8939-a4dbb08808e4#type=category&path=Biology/Ecology/HumanEcology/HumanPopulationStudy/Epidemiology/CardiovascularDisorder
  - ObjectID: 3498

See, for example, Sample Queries—FIGS. 111 and/or 112.

Referring to FIG. 117, an embodiment of the present invention can be described in the context of an exemplary computer network system 200 as illustrated. System 200 includes an electronic client device 210, such as a personal computer or workstation, that is linked via a communication medium, such as a network 220 (e.g., the Internet), to an electronic device or system, such as a server 230. The server 230 may further be coupled, or otherwise have access, to a database 240 and/or a computer system 260. Although the embodiment illustrated in FIG. 21 includes one server 230 coupled to one client device 210 via the network 220, it should be recognized that embodiments of the invention may
be implemented using one or more such client devices coupled to one or more such servers.

[3152] In an embodiment, each of the client device 210 and/or server 230 may include all or fewer than all of the features associated with a modern computing device. Client device 210 includes or is otherwise coupled to a computer screen or display 250. As is well known in the art, client device 210 can be used for various purposes including both network- and/or local-computing processes.

[3153] The client device 210 is linked via the network 220 to server 230 so that computer programs, such as, for example, a browser, running on the client device 210 can cooperate in two-way communication with server 230. Server 230 may be coupled to database 240 to retrieve information therefrom and/or to store information thereon. Database 240 may include a plurality of different tables (not shown) that can be used by server 230 to enable performance of various aspects of embodiments of the invention. Additionally, the server 230 may be coupled to the computer system 260 in a manner allowing the server to delegate certain processing functions to the computer system.

[3154] One embodiment of the invention is a system for knowledge retrieval, management, delivery and/or presentation, including a server programmable to maintain semantic information; and/or a client providing a user interface for a user to communicate with the server, wherein the processor of the server operatively performs the steps of: securing information from information sources; semantically ascertaining one or more semantic features of the information; and/or responding to user queries based upon one or more of the semantic features.

[3155] In a more detailed embodiment of the invention, information requests that are set to a Live Mode can be automatically added to a Watch List, even if they are not favorites (since the user would have indicated a preference for viewing them live).

[3156] In another embodiment a NewsWatch provides Application-Wide Awareness. For all requests that are marked as favorites in a Librarian, the Librarian will automatically build a “Watch List”—a list of requests to be “watched.” In other exemplary embodiments, favorite entities will also be used to populate the Watch List. In yet other embodiments the Live Mode and the Watch List will also include Newsmakers (or “people with newsworthy publications or annotations IN CONTEXT”).

[3157] In another embodiment of the invention the results can be colored differently based on the requests they are based on. Characteristics such as brightness and font size can be used to indicate freshness and spike alerts.

[3158] In another exemplary embodiment, when the Watch List is built by the system, the user will be able to edit it. The user will be able to include or exclude entire profiles or specific requests within profiles. For example, a user could save a request as a favorite yet not want live results streamed for that request (especially since this might clutter up the awareness stream). The user can indicate the “priority” of profiles and requests in the context of the Watch List. There can be multiple priorities, for example:

[3159] Normal: The default priority. Results for these requests and profiles are streamed normally, with the default time-sensitivity settings.

[3160] Low: Indicates requests that are favorites yet are not that important to the user, RELATIVE to other requests.

[3161] High: Indicates requests that are favorites and that are extremely important to the user.

[3162] These priorities are important because the system is essentially managing the user’s attention, while being flooded with results (the problem we are solving). Hence it is important that the user provide some hint as to how his/her attention should be managed by the system.

[3163] These same priority classes will apply to profiles in the Watch List. So a user will be able to indicate that an entire profile is of high priority. The Librarian will then dynamically make all requests created with that profile (already or moving forward) high-priority requests EXCEPT requests that have been marked as low priority. Or if a user indicates that an entire profile is of low priority, all requests created with that profile (already or moving forward) will be marked as low priority EXCEPT those that have been marked as high priority.

[3164] Imagine marking say, Life Sciences News as high priority and some random KC indexing RSS feeds as low priority.

[3165] In another embodiment a special Librarian component called the NewsWatch Scheduler is included. This scheduler can schedule requests in the watch list by invoking periodic Breaking News or Headlines calls, for example as follows:

[3166] For each request, calls will be invoked at a period of T, proportional to the time-sensitivity settings of the KC in the profile that has the highest level of traffic (to be retrieved from the server). If there is no Breaking News, Headlines will be invoked instead and visually marked as such in the UI “skin.”

[3167] The scheduler can, for example, ensure that at anytime, the merged priority queue (with merged results) has the following distribution: 60% (High), 30% (Normal), 10% (Low). This allows each request to be invoked “normally” yet the results are inserted into the merged queue based on the priority model above (scalar factors will be applied to the values in the priority queue). Furthermore, there can be regular dumping of the queue. If, for example, there are low priority results in the queue and a high priority request gets fresh results, the low priority results will be bumped (in order of freshness) if the queue is full until the above constraint is satisfied. This ensures priority-based scheduling WITH fairness. Low priority requests will always get some time in the queue and within that time-slot, prioritization will be based on freshness.

[3168] In another embodiment a Newsstand skin that can visualize the NewsWatch, for example, other skins will include a timeline view. In cases where News Watch skins don’t involve merged results, the prioritization scheme will be applied to the allocation of real-estate (since the “layout” will be spatial as opposed to temporal).

[3169] In yet another embodiment “smart skins” can intelligently change their layout based on the nature of the priority queue. So, for example, a Smart Newsstand will dynamically fade out portions of real estate that have “old” results or lower priority results.

[3170] In another embodiment of the invention the NewsWatch can run within the context of the Librarian, but in a special viewport and special mode that will look like full-screen mode. The viewport can be dockable on the side (right or bottom) or as a strip (a filmstrip like view) on top or below—the desktop will resize. Alternatively the user can set it to full-screen mode—e.g., applicable for second monitor scenarios.
In another embodiment the NewsWatch can have several interfaces including full-screen mode, dockable sidebar mode, etc (including special UI for Vista). One of the key interfaces in a preferred embodiment is as a tab on the Home Page. TheNews tab can be positioned, for example, where people go to check the news every morning, after getting back from lunch, etc. Preferably the tab can be customizable so the user can have the news presented in flexible ways—animations, timeline views, a Virtual Inbox (with synthesized streams of Breaking News and Headlines), a newspaper view, etc.

In another embodiment Live Mode in Nova can be very strategic to give user customers the first taste of the Awareness wave.

The client also keeps a drag and drop cache. This way, text extraction only happens on demand—for new or updated dragged and drop documents. The cache also ensures that if the dragged and dropped document changes, the updates are reflected in the semantic query.

In another embodiment, the Drag N Drop essentially allows the user to alter/add to the semantic network.

In other exemplary embodiments, the semantic network can be read-only and/or read-with-publishing and annotations. Drag and Drop allows the user to express intent naturally without thinking about Booleans, qualifiers, or even semantic wildcards (despite their power).

For example, when the information is sent to the server:

1) The concept list and flat list are sent.

For each KC:

2) Categorizer

a) For each ontology

3) Generate temporary map (mini semantic network)

4) N to N map therefore return results

The entire text is now sent (compressed). The temporary map is graph G1 which has to be built (including semantic ranking), semantically mapped to graph G2 (the indexed data), and then this must yield a graph G3 that is isomorphic to G1 and G2 AND is also ranked based on the isomorphism between G1 and G2.

This process occurs across ontology boundaries.

This process also involves context-switching—the input G1 is canonicalized regardless of semantic differences from G2, in order to yield G3. This is a context-switch. Dynamic searching, open, environment-based, Live search, file-based queries, natural-language-like queries, but are not NLP.

NLP has been abused to mean the computer must be able to answer random questions out of context. It is NLP—text is natural language, so are documents. But it is NLP in context.

An exemplary embodiment of the invention includes a priority-based scheduling Model to determine appropriate visualizations that are semantically correlated with the results on a relative scale and which manage the user’s attention given the “competition” for that attention from numerous results including: Input Variables, Brand New—fresh within the last minute (Boolean), Freshness (Number), Spike Alert (Boolean), Breaking News vs. Headlines (Boolean); Buzz (sustained traffic number of notifications in which results are appearing, etc.), document Size (perhaps an indication of relevance—might indicate user spent time creating a long document), and/or document file type (e.g., PDF might indicate more publishing emphasis; more an indication of a published output than, say HTML). Each result can have this metadata in the Live Mode logic layer. Example Algorithm:

**Weighted Average:**

**Brand New:** 10% (new in the last minute)=10, old=0

**Freshness:** 40% (normalize to max window size for each KC)

**Spike Alert:** 10% (spike=10, no spike=0)

**Breaking News vs. Headlines:** 20% (BN=15%, HL=5%)

**Buzz:** 10% (normalize to 0-10)

**Document Size:** 5% (normalize to a really large number—like 32 MB)

**Document File Type:** 5%

In this example, this allocation can be assigned at the skin level. This allows skins to emphasize different things. For instance, a Buzz skin might assign a much higher weight to the Buzz variable. A document size skin can be used to emphasize large documents (e.g., from the Web), etc. The final priority should then be normalized from 0-10, using the highest priority number seen so far as the denominator. The final priority number can then be used to bias the visualization on a continuous scale (the skin makes this determination):

**Font Choice** (assign fonts to priority buckets: e.g., Times New Roman: 8-10, Arial: 6-10, etc.—these are random examples but you get the gist)

**Font Size** (e.g., normalize from 3-10)

**Font Color** (normalize on RGB scale)

**Background Color** (normalize on RGB scale)

**NOTE:** Each individual variable should be visualized in the skin (if the skin so chooses) INDEPENDENT of the final Priority score:

**Visible/Hidden** (this can be used by a skin to completely hide HL, for instance)

**Bold/Italic/Underline** (use Bold to indicate freshness/spikes)

**Fonts** (use glowing/animated fonts to indicate buzz)

**Annotating Graphics/Glyphs?**

This exemplary model provides for integration of HL and BN into one logic and visualization (via configurable skins) of the differences between them on a continuous scale and a discrete/Boolean scale without the user having to make explicit technical decisions around scheduling and assignment. The user can choose to have BN only, HL only, or BN+HL (2 consoles). This choice+the skin+the priority-based scheduling model can then generate the final visual output.

In an exemplary embodiment, in Nova, the Nervana System can use the new Advanced Encryption Standard (AES)—the Rijndael cipher—for encrypting requests over the wire. This cipher has good performance characteristics, has no known weaknesses, and will be critical in highly regulated and sensitive environments where Nervana will be deployed—including the Pharmaceutical industry and places like the CIA. Strong security guarantees while optional are for drag and drop.

**Key generation**—to generate the shared secret to be used with the Rijndael cipher—is based on the PBKDF2 standard using a pseudo-random number generator based on HMAC/SHA1 and consistent with RFC 28...
An embodiment includes a KIS Web Service API for Live Mode: GetKnowledgeCommunityNewsSettings. This returns two arguments:

NewsUpdateFrequency: this indicates the update frequency of the KC:
- Never
- Everyday
- Every Week
- Every Two Weeks
- Every Month
- Every Two Months
- Every Three Months

An embodiment includes UpdateNewsTimeSpanMinutes: this indicates the recommended polling frequency for the KC:
- Never: -1.0 (if this value is -1.0, the Presenter should never poll—it means there is NEVER Breaking News—e.g., on an Archive KC)
- Every Day: 5.0 (5 minutes)
- Every Week: 360.0 (6 hours)
- Every Two Weeks: 720.0 (12 hours)
- Every Month: 1440.0 (24 hours)
- Every Two Months: 1440.0 (24 hours)
- Every Three Months: 1440.0 (24 hours)

An embodiment includes a Presenter API of the same name: GetKnowledgeCommunityNewsSettings. This wraps the call to the Web Service via SRCClient. It takes a Web Service URL and KC Guid, consistent with the cached KCInfo structure in the Presenter code.

SRCClient caches the values (via a hash table keyed by the KC GUID) for 24 hours. This way, the Presenter can keep calling the function without making unnecessary calls over-the-wire to the Web Service—since the KC settings are unlikely to change often.

Using this API to determine the polling frequency for Live Mode.

The return flag is explicitly checked and if the flag indicates only HGLOBAL support, an IStream is created using CreateStreamOnIGlobal.

SRCClient caches the values (via a hash table keyed by the KC GUID) for 24 hours. This way, the Presenter can keep calling the function without making unnecessary calls over-the-wire to the Web Service—since the KC settings are unlikely to change often.

This API determines the polling frequency for Live Mode.

The return flag is explicitly checked and if the flag indicates only HGLOBAL support, an IStream is created using CreateStreamOnIGlobal.

In one embodiment of the invention, there can be four states for Nova—Live Mode indicator (low), HL present indicator (average), BN present indicator (high), BN+very fresh (new items) present indicator (very high).

For example, the Live Mode enabled visualization—perhaps an actively broadcasting radio antenna (which is typical for visualizing “liveness”).

Additionally a very subtle background motion (like quiet Windows Media visualizations) somewhere in the Live Mode console—to communicate activity/streaming.

For example, for a given profile, the client can intelligently computes a list of KISes from which to ask for natural-language-based highlighting tables. It can do this, for example, to avoid asking KCs that share the same ontologies. The client can attempt to generate semantic highlighting before trying non-semantic highlighting—depending on whether the server was able to perform graph matching on the natural-language input. This is done for all natural-language query components. The server attempts to mirror as much as possible the graph isomorphism algorithm, but then applies MAJOR graph reduction else the highlighting table will have billions of billions of billions of entries (read: infinity)—Drag and Drop involves infinite combinatorial complexity. The client also tracks volatile natural language (documents and links)—so that if the document(s) change(s), it will update the highlighting cache on the fly. This way, if a user drops a document, edits/updates the document, and then refreshes the query, the user sees updated highlighting reflecting the new document’s contents.

The client can also handle the case where the KIS returns category URIs it does not semantically understand due to version mismatches. In that case, it reroutes the query (locally) to semantic wildcards.

In another embodiment, a chart of time distribution (in buckets of time—e.g., every hour) can be displayed to the user. For example, if the user notices a log-normal type time distribution with a long tail (with a ton of fresh traffic that then drops off), this is a nice and very intelligent hint as to how to adjust the attention dials—to have tighter constraints on the time axis.

Live Mode Stats and Analytics

In another embodiment, as shown in FIGS. 117 and 118, the user can generate a bar chart or pie chart in Live Mode pivoted by time, publisher, author, concepts, etc. So as Live Mode is streaming by, an auto-updated chart is displayed in a slide-out pane (which can be hidden and revealed). This is a powerful feature for real-time insight AND for attention-management. If the user can quickly glance at a “report/chart” (right beside the ticker) of how the Live results are distributed, this is a powerful cue as to how much time to invest in the Live stream at that point in time.

As shown, by example, in FIG. 118, examples of the top 10 publishers in the current Live stream, ranked by frequency of occurrence: The charts themselves can be Live—to reflect the underlying results as they change in real-time. The charts can have mini-hyperlinks so the user can click on a publisher like Merck to generate a search query and quickly only see Live results published by Merck (see “Search within Live Results” below). Also, for concepts, the most mentioned concepts can be displayed—using concept extraction and stats generation of the Live stream. For example, this will allow a gene researcher to setup Live Mode on genes and then quickly see which genes are getting the most mentions in the news.

The charts can show pivot tables/charts (e.g. publisher concepts time authors, etc.) and trends over time. See FIGS. 118, 119.

In another embodiment, a feature of the invention allows the user to toggle the charts by cluster—this is a way to track the distribution of semantically unique news articles. Charting/stats model was added to Reporting.

Stats Views: A Stats View (a variant of the docked view)—where the console is minimized to only show Live stats. If the user sees interesting stats, he/she can expand the view to the standard docked view.

“Live Views” (Live Sub-Queries): In another embodiment of the invention, the charts show “default” reports/graphics that the Presenter can display. The user can setup mini queries to chart specific scenarios. If a business
an analyst can create a quick sub-query and specify their top competitors as publisher pivots. This chart can be displayed in a slide-out view alongside the default chart. Other mini-queries can also be created to have “Live Views.” This will be very powerful for the purposes of Live (or Real-Time) Analytics. The sub-queries can be saved so each time the user opens Live Mode, they are right there. These are richer queries than mere keywords as indicated in “Search within Live results” below. Here, the user will be able to specify, say, a list of publishers, authors, concepts, etc., to pivot against.

[3246] Search within Live results: In another embodiment of the invention, in addition to the Live dial to be added in the Newton timeframe (time window to restrict Live results, maximum number of display roundtrips, etc.), the user could search for Live results. This is a quick way of scanning the Live stream for keywords of interest. Additionally, then these searches could be saved so the user can quickly navigate to the searches on demand. This is important if the user wants to track broad areas but then periodically search within Live results for specific terms, publishers, authors, etc.—especially if there is too much traffic at that point in time.

[3247] “Sub-Alerts” and Custom Spike Alerts: In another embodiment of the invention sub-Alerts refer to a feature where the user can setup mini alerts in Live Mode for additional attention management. In this scenario the user can indicate a keyword or publisher and then the system can generate a Spike Alert if that shows up in a new Live result. These sub-alerts can then be saved. This allows the user to more precisely manage their attention in the context of a broader Live Mode stream. See FIG. 120.

[3248] The sub-alerts feature with Home and End Button in the Live Mode control bar to allow the user to seek to the freshest or oldest result in the ticker. This can be especially powerful in the event that the user has seen all the fresh results and then fast forward so Live Mode is quiet for a while, hits Play and then immediately wants to seek to the freshest result.

[3249] This feature can be important in cases where Live Mode is streaming but the user missed a spike alert and has no way of quickly knowing if there is new stuff downstream or upstream in the ticket. This coupled with buttons to navigate to the freshest/oldest will be very powerful and can aid usability.

[3250] A feature to determine the freshest and oldest result times (N hours/days ago)—in a Live Mode status bar. This will be important in cases where Live Mode is streaming by but the user missed a spike alert and has no way of quickly knowing if there is new stuff downstream or upstream in the ticket. This coupled with buttons to navigate to the freshest/oldest will be very powerful and would aid usability.

[3251] Average N hours ago (in addition to MIN and MAX)

[3252] Title and publisher of freshest result. This part of the bar can have an “Expand” button to show the N freshest results (where N maybe is <=5)

[3253] A feature wherein the Home and End buttons are in the Live Mode control bar. This can allow the user to seek to the freshest or oldest result in the ticker. This is especially powerful in the event that the user has seen all the fresh results and then fast forward so Live Mode is quiet for a while, hits Play and then immediately wants to seek to the freshest result.

[3254] A feature to quickly determine the freshest and oldest result times (N hours/days ago)—in a Live Mode status bar. This can be important in cases where Live Mode is streaming by but the user missed a spike alert and has no way of quickly knowing if there is new stuff downstream or upstream in the ticket. This coupled with buttons to navigate to the freshest/oldest will be very powerful and would aid usability.

[3255] Average N hours ago (in addition to MIN and MAX)

[3256] Title and publisher of freshest result. Ideally this part of the bar can have an “Expand” button to show the N freshest results (where N maybe is <=5)

[3257] Average traffic rate (new documents per hour)

[3258] This feature can also allow users to pause the ticker and watch the status bar for breaking changes. For busy people, this is a great time optimizer.

[3259] View a Federated Profile pivoted by “Knowledge community” (KC) but ALSO check and uncheck KCs that I want to view. This can be very powerful. A user can decide to view only a few KCs in the federation based on the state of the results at the time. Then as the user browses around, the user can check more KCs back in to get a more comprehensive view. Checking and un-checking KCs will automatically edit the HTML DOM in the displayed consoles—the DOM will be initially populated with all results; then parts of the DOM will be hidden or exposed based on the selected KCs—a hash table into the DOM keyed by KCID for quick lookup. For example:

---

[*] All Knowledge Communities * open by default

[U] to indicate selected KCs * all should be selected by default

[X] Medline

[ ] Life Sciences News

[ ] General News

[X] Life Sciences Web

[X] Life Sciences Patents

[ ] FDA Regulatory Information

[ ] FDA Regulatory Information Pages

[X] ProQuest Medical Library

[*] Medline * closed by default

[*] Life Sciences News * closed by default

[*] General News * closed by default

[*] Life Sciences Web * closed by default

[*] Life Sciences Patents * closed by default

[ ] FDA Regulatory Information * closed by default

[ ] FDA Regulatory Information Pages * closed by default

[ ] ProQuest Medical Library * closed by default

---

[3260] Federated results UI with a tree view so can pivot by KC. In one embodiment of the invention, a show and hide functionality so that the user can select a subset of the profile KCs (within the “All Knowledge Communities” pivot.

[3261] The Presenter can indicate WHICH KCs have results within the cached result set. “All Knowledge Communities” can be opened by default so that node does not require any hint. The rest can have hints—subtle features, such as the one describe in this embodiment actually aid discovery in powerful ways. The hints can indicate how many results each KC has in the results set AND some kind of very subtle alert if the result count is non-zero. This can allow the user to essentially “search” for results by KC—the user can continue navigating until the user sees results from KCs that the user is particularly interested in within the profile. As such:

---

[*] All Knowledge Communities (40+ results)

Show/Hide UI (popup/drop-down):

[*] Medline (28 results)

[ ] Life Sciences News (7 results)
ers can go to track issues around ongoing litigation. The semantic profile in this case could include filed documents, patents, legal briefs, and other semantic inputs that describe the lawsuit in question. This amounts to semi-automated litigation support—as opposed to hiring armies of people to find documents, track reports, etc.—Nervana Discovery API automates this process to maximize the efficiency of your litigation staff.

[3268] The Idea Exchange: In an increasingly global and competitive marketplace, the generation, capture, and sharing of ideas is critical to the survival and success of today’s businesses, especially those in IP-intensive industries. As an illustration, search engine leader, Google® requires its engineers to spend 20% of their time on new ideas and many of Google's® well-known products and services, such as Google News, were born of this concept. Competitor Products powerhouse, Procter & Gamble®, also employs innovative techniques to generate new product ideas.

[3269] This problem is exacerbated by the fragmentation of people, groups, departments, etc.—often times, ideas are not comprehensively collected and connected to other ideas and people in order to facilitate the creation of new products and services. Additionally, to make matters worse, when employees leave, their ideas usually leave with them.

[3270] In another embodiment of the invention, the Nervana Idea Exchange is a business application that facilitates the capturing, sharing, and connecting of ideas across physical and organizational boundaries. Using the Nervana Discovery API and Nervana’s proprietary Drag and Drop semantic technology, ideas can now be collected and connected to relevant people, ideas, documents, patents, and other internal or external documents. The Idea Exchange can be an enterprise portal that focuses on knowledge sharing but powered by Nervana to automate this critical business process.

[3271] In an exemplary embodiment, employees are encouraged to submit ideas to the Idea Exchange, in a standardized form—new processes can be established to add idea submission to standard employee reviews and to provide awards and other incentives for high-quality and high-quantity submissions. These ideas can be simple Word documents or email messages, and are preferably easy to capture (including with attachments)—in an unstructured form. The simplicity and speed of capture is facilitated by an unstructured text format—that is unstructured data processing of ideas can be collected in unstructured text allowing them to be intelligently processed with the Nervana platform.

[3272] In another embodiment of the invention, ideas can also be reviewed and ranked by others in the organization so the best ideas bubble up to the top. However, merely capturing ideas is not enough. For ideas to have value, they must be actionable. The Nervana Discovery API allows the intelligent semantic processing of ideas in order to facilitate a highly automated and powerful idea network. Imagine dynamic connections as shown by example below:

Idea
  * Attachments
    * Attachment 1: Nervana-Generated Links to similar ideas, people (internal and external), patents, internal documents, news, web pages, etc.
    * Attachment 2
    * Attachment N
  * Review (comments)
    * Comment 1: Nervana-Generated Links to similar ideas, people (internal and external), patents, internal documents, news, web pages, etc.
This exemplary virtual network forms a “Web of Ideas.” A user can logon to the portal and semantically search for ideas using keywords or natural language (powered by Nervana). This search can also include filters such as high-quality ideas with a high rank. The user can then select a summary view of different idea entries to preview the submissions and then also navigate to the attachments and/or comments and also view semantically similar ideas, relevant people, patents, etc. Some of these can be hyperlinked so the user can browse the virtual, dynamically generated Web of Ideas.

This is extremely powerful. Relevant documents, patents, news, etc. can be intelligently surfaced if an idea forms the context of a user’s knowledge workflow. This essentially exposes the organization’s internal IP to where it can become actionable—in the context of ideas and in the context of a knowledge-worker’s workflow. Furthermore, ideas get captured in a central repository such that they remain with the organization and remain connected to ongoing intellectual workflow, even after the employees that might have created them leave the company. This solves a critical problem in many enterprises—that of duplication of effort. With the Idea Exchange, old ideas can have new value—they can be resurrected in new contexts thereby facilitating knowledge reuse. (FIGS. 127-128).

“Semantic by Default” Mode: In another embodiment of the invention includes semantic wildcards that are “on by default.” In this mode, the UI can map queries to wildcards behind the scenes, unless the user explicitly indicates otherwise. This scenario can complement the preferred mode, where wildcards are mapped behind the scenes only if the user indicates as such.

For example this mode can employ the “~” sign as the opposite of a wildcard when in this mode. For example, “heart diseases” genes becomes “~heart diseases” ~genes (behind the scenes) but “heart diseases” =genes becomes “~heart diseases” genes (behind the scenes).

This way, the user can still indicate that they want a keyword search. This is always going to be needed. Examples: Find everything on cancer by the University of Washington.

Today: *cancer university washington
With the “Semantics by Default” mode turned on, this will become *cancer *university *washington (which is not what the user wants). The ~ sign allows: cancer =university =washington

NOTE: This compatibility mode, in some or many cases, will be the user’s intent and will eliminate any upfront training.

In an alternative mode the ~: is used an alternative to just — (to be consistent with *—)

“Group” Profiles and other Namespace Objects

The Nervana Talent Matching Agent: In one embodiment of the invention a Nervana Talent Matching Agent (TIMA) is a novel software application that helps human resource (HR) analysts scan, screen, filter, match, and rank resumes and job openings, similar to what a human domain expert would do. The software which can be used directly or integrated with existing systems, employs Nervana’s award-winning artificial intelligence engine to intelligently and automatically match resumes and job openings, with unsurpassed efficiency. This helps HR match the right candidates to the right jobs, and even proactively provide job opening recommendations based on ongoing corporate initiatives, thereby helping to better align employees with organizational goals and to increase employee retention.

In another embodiment, the Nervana Talent Matching Agent (TIMA) is a custom software application that addresses a critical and growing need in recruiting and staffing—that of most efficiently and quickly matching candidates (typically via their resumes) to the right jobs. The Nervana Talent Matching Agent, which complements and/or integrates with existing Applicant Tracking Systems, employs Nervana’s award-winning semantic matching technology (“Drag and Drop”) to intelligently scan, screen, filter, match, and rank resumes and job openings, similar to what a human domain expert would perform.

This has many business benefits. First, with an ever-increasing number of electronic job applications, HR managers need assistance in screening and matching candidates to jobs, in order to increase placement quality and save HR analysts and members of their organizations valuable time and money. Employers often complain of the low quality of matches they get from job placement sites and applications. In the process, valuable time is wasted and higher quality candidates are often missed. In today’s highly competitive job market, it is critical that the best candidates be found and matched to the right positions, and that HR managers productively spend their time sourcing candidates.

The low quality of placement matches often results from the fact that job placement sites and applications use keywords to match candidates and jobs. This has many problems. Keywords typically do not capture the essence of a candidate’s job experience or the nuances of a job opening. The requirement to use exactly the right keywords to generate an exact match places an undue burden on employers and candidates. The result is that candidates must often pick extremely broad keywords that could mean different things in different contexts or very narrow keywords that could result in false misses (“false negatives”). And oftentimes, multiple keywords are analyzed with no regard to the relative rank of those keywords to the matching process. Also, candidates and employers often want to run rich, natural searches like: “Find all job openings in the Pacific Northwest for executive-level sales candidates with experience selling to Fortune 500 companies.” Nervana is a technology that enables a rich and flexible search, akin to posing the question to a human HR consultant.

Furthermore, the same candidate or open interests or roles are often expressed differently. “Business Development” is often referred to as “biz-dev” and in some contexts is regarded as equivalent to “Corporate Development.” This problem is particularly acute with new roles in dynamic, fast-moving industries. In the 90s, some companies added a “Chief Knowledge Officer,” whereas in other companies, this responsibility was—and still is—handled by the Chief Information Officer. Some companies now have “Chief Compliance Officers,” a relatively recent role since the passage of
Sarbanes-Oxley. In the Pharmaceutical Industry, some companies have “Directors of Lead Discovery,” whereas other companies fold this position into executive-level “Informatics” positions. Indeed, in the Life Sciences industry, the “Informatics” role still means different things to different people.

[3288] This situation—fluid, changing, roles oftentimes expressed in different ways in different companies—typically leads to frustration on the part of job seekers and employers, and many times results in false misses that go completely undetected. Nervana’s software employs artificial intelligence to automatically handle these nuanced and ambiguous descriptions of resumes, openings, roles, etc., and performs intelligent matching behind the scenes, ensuring high-quality matches for both candidates and employers. As roles evolve, the software adapts accordingly; existing resumes and job openings will still be matched even if they use old terms to describe new positions that mean the same thing.

[3289] Another very severe limitation of existing job placement software systems is the fact that they typically match a candidate to a job or they don’t. In other words, the match is handled as though it is binary—a candidate is either a fit or isn’t. In reality, things are much murkier. A candidate might not be a perfect fit for a stated position, but he/she may be an acceptable (or even exceptional) fit for other reasons. And the candidate might be a better fit for another position or might become a better fit after maybe a couple of years of adding key accomplishments to his or her resume. In short, the matching process requires human-like judgment—where things are often not black or white, but rather with shades of gray. This notion of supporting and complementing human judgment—a key test of artificial intelligence—is one of the unique innovations that Nervana Talent Matching Agent provides employers and candidates. The matching process not only deals with completely unstructured text (indeed, the process is as easy as dragging and dropping an entire resume in order to find an intelligent match), and is not only tolerant of nuanced and potentially ambiguous interpretations, but also ranks the quality of matches. This is very powerful as it allows candidates and employers to find matches in the “vicinity” of what they intended; indeed, oftentimes, this could result in the discovery of candidates and openings that might even be superior to the original goal.

[3290] In one embodiment, the Nervana Talent Matching Agent allows the matching of resumes-to-resumes (to find similar resumes, even if they are described differently yet mean the same thing), and resumes-to-job-openings (and vice-versa). In addition, the software supports what Nervana calls “Proactive Recruiting.” This refers to the active integration of recruiting with other critical corporate business processes and the tracking of those business processes by HR to give proactive recruiting recommendations. This would provide a critical competitive edge—by recommending job openings and engaging with potential candidates and thought leaders before there is an explicit job opening. HR can help create job openings based on the potential strategic value to the organization. The Nervana Talent Matching Agent supports this business process by intelligently analyzing publicly available corporate email and documents, matching resumes to those emails and documents, and providing HR with intelligent recommendations of great matches between candidates and ongoing corporate projects, even if those projects do not have existing job openings. This is very powerful as it makes the business process proactive and predictive.

[3291] In yet another embodiment of the invention, the Talent Matching Agent mines and connects resumes and job descriptions to publicly available corporate emails to generate ranked interviewer candidate lists. This process uses artificial intelligence to recommend employees that share interests with the job applicant and are probably most qualified to interview the application. This is a very valuable feature as it allows HR to pick the best interviewers for each candidate (a very common and time-consuming problem), thereby further helping to ensure high-quality placement and retention.

[3292] And even if there are no current job openings, or a candidate is rejected, or there is a bad fit, Nervana’s support for “proactive recruiting” helps make future connections where none might currently exist. If there are new corporate projects in the future for which a rejected candidate might make a good fit, HR will be notified immediately, and by connecting resumes to employees, HR can proactively engage with employees that share similar interests to—or recently started working on projects relevant to—target (and likely very talented) candidates in order to periodically woo them to join the organization.

[3293] The software industry is undergoing significant change. The advent of managed code (.NET and Java) has simplified and accelerated software development and has further fueled the commoditization of software as a monetizable asset. In actuality, Nervana straddles two industries—the software industry and the information industry. And like software the information industry is also being faced with mass commoditization. As an illustration, 10 years ago, the concept of online news being free was unheard of—paid subscriptions to Reuters, Factiva, etc. were required to gain access to newsfeeds. Yet today, Google News and Yahoo News are free and aggregate valuable published content.

[3294] There is significant—and growing—tension between the software and information industries. New information-based services need both—access to the underlying content and software-based features and scenarios that make content more discoverable and valuable.

[3295] The mass commoditization of information has made it extremely difficult to anyone to monetize information directly. Google monetizes information access indirectly—they give it away and monetize the advertising it generates. The advertising business model has proven much more profitable for Google than many would have predicted even five years ago (even though many detractors still remain). And there is a growing perception—by industry watchers and customers alike—that information (and everything around it—including access) should be free.

[3296] This is an untenable situation. The laws of economics have not expired. There has always been—and will always be—a strong correlation between price and value. In large (albeit not absolute) measure, nothing of value is really free.

[3297] This “information and everything around it should be free” perception will create a significant challenge for information and content providers in the near term and huge opportunities for search engines and information provider in the long term. In the short term, information and content providers must meet these challenges by demonstrating that while information might be free, knowledge—the useful and meaningful insights gleaned from information—is not. There are two fundamental dimensions to measuring this challenge:
the quality and scope of information and the innovation we must deliver to the marketplace around that information. [3298] “Reach:” The Quality and Scope of Information [3299] The instant invention ensures that the most comprehensive and highest-quality information is flowing through a semantic medium. [3300] The Drug and Drug Wave [3301] One embodiment of the innovation is Drug and Drug feature which changes the conversation and introduces the first signs of a new navigational (as opposed to “search”) paradigm. Drag and Drop, for example allows the user to drag and drop a chemical structure image for a search. [3302] The Reporting, Analysis, and Workflow (RAW) Wave [3303] The way to think about RAW is this: it changes the medium from one that formulates and interprets queries to one that generates insights. Our semantic services and knowledge communities are Knowledge Mining platforms. As an illustration, Nervana Medline currently processes intelligent semantic queries and is able to do things PubMed and other services cannot do. This gap is significantly widened with Drag and Drop functionality. [3304] The RAW wave enables the Nervana system and the Librarian from a Question-Answering medium to a full-fledged Intelligence medium. The following are exemplary questions that are addressed with RAW technology: [3305] Print out a list of genes known to be correlated with the incidence of Alzheimer’s Disease. [3306] Which of my competitors are most actively researching chemical compounds relevant to the treatment of bone diseases? Further functionalities can include e.g., Display data as a LIVE bar graph within the Nervana Librarian showing “relative research activity” per competitor; view the results with different charts—pie charts, etc. without leaving the Librarian; email the report to the “Business Intelligence” email alias; and pipe the analysis into Microsoft Excel for further processing. [3307] Who are the most prolific authors researching the interaction of genomics and the bird flu virus? Further functionalities include e.g., hiring/recruitment of prolific authors (critical to HR) and/or setup licensing agreements or additional business development; and print out a LIVE bar graph showing the authors’ relative output. [3308] Which are the most active biotech companies researching kinase inhibitors that might show efficacy for the treatment of HIV? Further functionalities can include e.g., print out a report with the list; and email the list to the Head of Business Development for follow-up. [3309] Display a trend-line graph showing research activity over time for potential replacements for statin for the management of cholesterol but only in post-menopausal women. The graph should include trends for the Big 20 Pharma (semantically generated in real-time via an institutional ontology). And the graph should show the trend for the past 10 years. [3310] What is Pfizer working on and how has their research focus changed (if at all) in the past year and the past five years? [3311] Was Merck’s research output affected by their legal problems around Vioxx? If so, by how much? [3312] Which research institutions are our competitors partnering with most aggressively? Show me trend-lines for co-authorship amongst all our top competitors and scientific research institutions. [3313] Which of our competitors are most aggressive in research, IP, licensing and/or M&A for monoclonal antibodies? Now which of those companies have successfully submitted drug applications to the FDA? Now show me toxicology data for those drug applications. [3314] Perform a pattern-recognition analysis to show what therapeutic areas (ranked semantically) are showing the most promise for the application of interleukin-10. [3315] Which of our competitors’ drugs are generating the most buzz amongst consumer blogs? Show this to me organized by timeline—the past week, the past month, the past quarter, and the past year. [3316] In addition, the user can just drag and drop this research paper just published by Merck and using the paper as context, the exemplary embodiment can show a report of research, IP, marketing, and drug-application activity for each of the competitors. The search can be modified to show the user the same report but for internal documents—so that the user can compare how much IP the business entity of interest is generating in the area with how much work the competitors are doing. A print of the comparison report can be generated and then email it to my colleagues. [3317] If for example, Novartis just announced via a Press Release that they have received preliminary (Phase I) FDA approval for a new Matrix Metalloproteinase Inhibitor for the treatment of colorectal cancer and the Press Release includes safety information on recently concluded clinical trials conducted by the company, the user can just drag and drop the PDF of the Press Release into the search. The following questions can be posed using RAW technology: Using the Press Release as context, considering the details of the cited clinical trials, and considering the fact that this was only a limited Phase I trial, how seriously should we regard this development? Is the inhibitor family credible from a therapeutic and safety standpoint? If so, which other cancers has it shown efficacy against? If the list is substantial, which other vendors have compounds of the same family available for licensing? [3318] The ability to answer these questions constitutes the RAW wave. This invention includes a query interface at the Librarian, visualization UI in the Librarian (to visualize the results—WITHOUT requiring any third-party tools), the means to generate pivot-table like views in the Librarian, the means of generating a Report View on a per-request and per-result basis in the Librarian (the Librarian becomes an Intelligence Viewer), the means to present and transfer results into Excel using the Excel object model and using XML, etc. [3319] With the RAW wave, the Nervana Librarian is to unstructured data what Microsoft Excel is to structured data (this constitutes a massive vacuum on knowledge worker desktops). [3320] The Awareness Wave: The Awareness Wave involves the implementation of Live Mode and the News Watch. This can provide semantic and contextual awareness to users and will allow them to track their favorite semantic queries and interests in real-time. [3321] The Semantic Exploration and Visualization (SIEVe) Wave: The Semantic Exploration and Visualization Wave will involve the implementation of Category Discovery, Bookmarks, Entities, and basic forms of Deep Info.
[3322] The Communities and Collaboration (CnC) Wave: The CnC wave introduces User Publishing, Smart Groups, Annotations, and the automatic semantic inference of Experts, Interest Groups, Newsmakers, and Commentary (all of which will be added to the “Dossier” as additional semantic/discovery axes). In this wave, Newsmakers will be added to Live Mode and the News Watch, further enhancing the Awareness Wave.

[3323] The Unification Wave: In this embodiment the invention lies together fragmented concepts like local documents, online documents, people, concepts, semantics, groups, Time, etc. into a cohesive, universal canvas. Advanced Deep Info that will allow the Universal, Dynamic and Semantic navigation from any silo to any silo, enhancements to the Deep Info Mini-Bar, etc. are the product features in this wave.

[3324] Lack of researcher productivity is most readily apparent in the Life Sciences industry. It is estimated that the industry will spend nearly $60 billion dollars globally on R&D in 2007 alone. However, the number of NME’s (New Molecular Entities, an important proxy for the progress of new drug development) filed at the FDA has dropped by 58% over the last decade. Even more alarming is the comparison of dollars spent to NME output as shown in the figure to the right.

[3325] The decline in worker productivity is particularly troubling for the industry’s largest pharmaceutical and biotechnology companies. These organizations rely on an efficient drug discovery process to mitigate the threats of expiring patents, and to build a pipeline of future revenue-producing drugs. In 2005, patent expirations put $12 billion in annual pharmaceutical revenues at risk to competitive entry by generic replacements. Industry experts and managers point to inefficient knowledge management as the main driver for the precipitous decline in NMEs. Scientific complexity and information overload are predominantly driven by features such as, but not limited to, technology and large-scale institutional problems including deciphering of the Human Genome; a proliferation of new drug targets; the accumulation of massive volumes of clinical data; the digitization of health records; the increase in litigation risk and environmental factors; and a lack of collaboration among scientists.

[3326] The proliferation of data inputs and development variables has created a combinatorial complexity problem in drug discovery so great that researchers are abandoning methods of hypothesis generation and validation and reverting to simple trial and error. All these factors combined have driven the price tag to develop a blockbuster drug to nearly $1 billion. Consequently, pharmaceutical and biotechnology companies are forced to abandon drug discovery projects that do not have potential to generate at least $1 billion in future revenue. By leveraging enabling technologies and processes to drive down the cost of new drug development and more effectively collaborate on research initiatives, thereby reducing the hurdle rate for investment in new research projects, life sciences organizations can produce blockbuster drugs more quickly while profitably addressing smaller markets with niche drugs.

[3327] FIGS. 126 and 127 illustrate the combinatorial complexity inherent in today’s drug discovery process.

[3328] FIG. 127 illustrates Nervana’s Semantic infrastructure addressing the combinatorial complexity of The Nervana System™ provides sophisticated, dynamic semantic indexing and ranking of content on a wide array of data sources without the need for “manual tagging” or formal semantic markup. The solution provides workers with the ability to ask questions “naturally” within the appropriate context. These questions can cross multiple domains and information repositories. The Nervana engine correlates all the possible combinations of meaning for this request and returns the most relevant, timely results from the system. Importantly, the product offers one of a kind features for the end user including: Semantic Wildcards (the ability to ask for information across multiple areas of knowledge without having to precisely form a query), and “Drag & Drop” searching (using a document or entity to create a query, where the system analyzes the sample and then finds semantically similar materials).

[3329] Delivered as an online subscription service or an internal server installation, the Nervana System™ meets the needs of individuals, small organizations, or large enterprises. With the power of semantics and the intuitiveness of keywords, Nervana’s approach comes as close to natural language query capabilities as is currently computationally feasible. FIG. 128 illustrates Nervana’s semantic indexing and query architecture. FIG. 129 illustrates a platform and applications stack according to one exemplary embodiment of the invention.

[3330] Nervana Discovery Spaces™ is built on top of this platform. This employs Nervana’s unique and award-winning semantic matching technology to build smart connections between people, concepts and information. This application can enable knowledge workers to discover and share information in a totally natural way, and represents a new paradigm for information management for knowledge workers and enterprises. Not unlike a wiki but powered by semantics, the application employs community and intelligent connections to expose collective knowledge. This is much more powerful than mere search as it employs collective intelligence to create a collaborative knowledge discovery and sharing surface.

[3331] Nervana Discovery Spaces™ can include the following components:

[3332] The Nervana Entity Framework™

[3333] The Nervana Entity Framework™ is an application framework that can allow knowledge workers to define semantic entities of interest to them and their organizations. These semantic entities capture user and organizational intent in a way that simply is not possible today. Examples of semantic entities include: Topics, Customers, Competitors, Partners, Products & Services, Projects, Ideas, Business Plans, Meetings & Events, Annotations, Favorite Documents, Job Openings, Resumes & Bios, Patient Health Records, and Customer Support Issues.

[3334] Users can create Nervana entities corresponding to what they wish to track. These entities can be expressed with documents, keywords, and/or concepts. The Nervana System™ then automatically processes the entities based on the expressed context and the semantics of the entity type. This is an extremely powerful framework for allowing powerful research and business intelligence in a way that is natural to users. Each of the aforementioned semantic entities will empower researchers, sales staff, marketing managers, call center staff, project managers, and other knowledge workers to define and track items of interest in the way they think. This is not possible today and Nervana’s unique technology enables it. All subscribers will also be able to get Managed Query Services. With this feature, users will be able to email
Nervana natural-language descriptions of precisely what they want. Nervana support staff will then convert that description to an entity for semantic tracking. Examples of natural-language queries that Nervana can process include:

[3335] Which clinical trials for Cancer drugs employing tyrosine kinase inhibitors just entered Phase II?

[3336] What are my top competitors doing in the area of Cardiovascular Diseases?

[3337] Find recent research by Pfizer or Novartis on the impact of cell surface receptors or enzyme inhibitors on heart or kidney diseases

[3338] Find the top experts researching Genes that might cause Mental Disorders

[3339] Nervana’s technology is the only one that can intelligently and efficiently answer questions like those listed above, as has been validated by Proctor & Gamble Pharmaceuticals and others.

[3340] The Nervana Discovery Web Services™ the Nervana Discovery Web Services™ comprise of two servers, the Nervana Knowledge Integration Service™ and the Nervana Knowledge Domain Service™ that provide semantic indexing, query processing, matching, and ranking. These services are programmable via industry-standard Web service protocols and also flexibly support arbitrary content sources and ontologies.

[3341] The Nervana Information Agent™. The Nervana Information Agent™ is a middleware engine that takes user-defined semantic entities and employs the award-winning Nervana semantic engine (Nervana Discovery Web Services™) to periodically run those queries. Results are then published onto a self-authoring discovery portal (called a “Discovery Space”) that is mapped to the entities users wish to track. Results are also published as RSS, allowing users to subscribe essentially to “semantic views” of their projects and their organizations in ways they cannot currently do today. Market entry will be aided by broad industry support for RSS, including support in Microsoft Internet Explorer Version 7 (now with over 80% market share), industry-standard RSS readers like NewsGator, and deep RSS integration in Microsoft Windows Vista and Outlook 2007. Customers will be able to use the Information Agent either on-demand or on-premises, depending on their needs around enterprise context access and security.

[3342] Nervana Entity Directory™. The Nervana Entity Directory™ is a Web-based application that organizes user-defined entities into a discovery portal. Essentially, it is a semantic version of today’s corporate directories. However, it is much more powerful and intuitive in that it exposes and organizes entities at the conceptual rather than physical level. This essentially creates the equivalent of a “smart portal” that is organized and managed conceptually and naturally, based on personal and organizational entities.

[3343] The Nervana Ontology Framework™. The Nervana Ontology Framework™ refers to a group of ontologies customized for different content packages. These can be configured based on the industry vertical. The Life Sciences ontology framework consists of Cancer (NCI), the Gene Ontologies (GO), MeSH, and SNOMED. Nervana already has over 50 industry-standard ontologies across multiple vertical markets. Similar frameworks will be finalized for additional verticals as Nervana expands, based on Nervana’s ontologies and ontologies licensed from partners like Taxonomy-Warehouse and Intellisicpe. The ontology framework also includes proprietary tools for ontology automation, refinement, alignment, and certification. FIG. 130 illustrates an embodiment of an ontology framework.

[3344] Nervana Ontology Automation

[3345] Nervana’s algorithms perform optimally when there are good ontologies in the domain of interest. However, the algorithms do not require perfect ontologies as they employ sophisticated ranking and filtering heuristics to allow for imperfections at the ontology level. The Nervana Ontology Framework™ includes software for ontology automation, alignment, and certification. Nervana’s ontology automation is community-driven, thereby ensuring that the ontologies reflect the true perspectives being generated and shared by the social network. To realize this, Nervana employs a dynamic ontology feedback loop mechanism where foundational and domain-specific ontologies are “cross-fertilized” with community-based and corporate ontologies, which in turn are semi-automated—dynamically inferred and refined based on documents, ideas, projects, and annotations published to the network, and then vetted by domain experts. This model accomplished several key things:

[3346] It avoids the “cold-start” problem, common with machine learning systems. The system initially employs other ontologies in the ontology stack, selected at a broad level based on the community of interest. As users then publish and share information, higher-level ontologies are then inferred on the fly. The semantic indexes are then periodically regenerated in the background, thereby incorporating new user-driven ontologies and learning as it goes on.

[3347] It incorporates community-based perspectives and vocabulary which might be difficult or impossible to acquire otherwise.

[3348] It scales much better than other artificial-intelligence systems, as the system can be deployed to new verticals and communities relatively quickly.

[3349] It strengthens and reinforces strategic lock-in because the learned ontological data becomes a key proprietary asset which is lucrative in value as time goes on and as the community builds. Links will get smarter, attracting even more users, and consequently generating a positive feedback loop. FIG. 131 illustrates an embodiment of an ontology automation feedback loop.

[3350] THE NERVANA CONTENT FRAMEWORK™

[3351] As it is an intelligence platform, the Nervana Information Agent™ needs access to content. The Nervana Content Framework™ defines the content packages that customers will be able to access naturally and semantically. An overview of the technology is presented in FIG. 145. The content framework consists of two pillars. Nervana provided Internet Content and enterprise content.

[3352] Nervana-Provided Internet-based Content: free or premium content that is generally horizontal in nature yet valuable especially to small and medium-sized businesses. This content will include, for example, but not limited to, following embodiments:

[3353] Free content (to drive usage, community, and network effect):

[3354] Industry-related News

[3355] Industry-related Web Pages (including academic & government web pages)


[3357] Scientific Literature

[3358] Blogs

[3359] Up-sell (for premium subscribers):

[3360] Scientific Lecture Videos
[3361] Podcasts
[3362] Theses & Dissertations
[3363] Industry Events
[3364] Company Profiles
[3365] Regulatory Information
[3366] Clinical Trials
[3367] Drug Applications
[3368] Drug Approvals
[3369] The strategy is to combine some valuable content with social-networking to create and take to market a revolutionary discovery and collaboration-centered application based on semantics, context, and natural-language.

[3370] Enterprise Content: Enterprise customers will also be able to semantically connect their entities with internal documents and repositories, including premium subscription content which they already pay for. Nervana’s technology integrates with the major enterprise software applications such as Lotus Notes, Outlook, Microsoft SQL Server, Oracle DBA, and Documentum. In addition to integration with the major internal data sources, Nervana can partner with various vendors to build a solution that will port any legacy data source to an XML format allowing the Nervana solution to unlock this data repository. Lastly, Nervana can provide custom ontology development and integration to enhance a company’s proprietary knowledge base. FIG. 132 illustrates an embodiment of the Nervana Content Framework.

[3371] THE NERVANA SEMANTIC SOCIAL NETWORKING FRAMEWORK™

[3372] The Nervana Semantic Social Networking Framework™ refers to an application-layer framework to host and semantically mine user profiles, projects, and other entities. The framework also connects those entities with other users in the network. As an example of the service, imagine physicians being able to semantically discover other doctors that have patients with similar health issues (for evidence-based medicine) or researchers being able to discover other people working on similar problems within their organizations and also at research partner firms and universities. Or imagine knowledge workers being able to semantically and dynamically discover each other’s favorite documents. The framework also provides for strong security, including authentication, encryption, and access control. Users can apply access control rules on their private entities in order to restrict access to people they trust while keeping public entities open, in order to allow relevant people to discover them. Customers will also be able to create group profiles, in addition to individual profiles, and configure group membership rules. When a user logs on, he/she would see his/her entities and those of all groups to which he/she belongs. This is very powerful, as it facilitates seamless knowledge sharing and allows organizations to create much more intelligent shared portals relevant to communities of interest.

[3373] THE NERVANA PRESENTATION FRAMEWORK™

[3374] The Nervana Presentation Framework™ refers to user interface components that enable searching the social network and also clustering results based on various attributes. These components enable flexible “skinning” of Discovery Spaces, enabling customers and ISVs to present parts of the social network in unique and creative ways. Components will be built on industry-standard frameworks, including AJAX and Microsoft Atlas, allowing for cross-platform skinning at the presentation layer. FIG. 134 graphically illustrates an embodiment of an end-to-end process cycle of Nervana Discovery Specs.

[3375] Product Illustration for Nervana TalentEngine™

[3376] The following are illustrations of what a typical Nervana user will see when logged in to Nervana TalentEngine™:

[3377] My Talent Space, General, My Job Queries, All, Information Technology, Program Manager in Security Business Unit (47 Days Old); Candidate Recommendations; Peter Landon @ Sun Microsystems; Open Full Candidate Profile (e.g., Search for Peter @ Sun on Google, Search for Peter on Google, Connect to Peter via LinkedIn, Search for Candidate on LinkedIn (e.g., People on the Corporate Career Website, People via Referrals, People in Nervana’s Database, People on Job Boards, People on the Web, People in Social Networks, Bloggers, People in the News, News Group Contributors, Inventors, Scholars, Institutions with Expertise, Events and Conferences); Find Talent Like This (e.g., All, General, CEO’s like Joe Smith, VPs of Marketing like Mike James); Company Projects and Initiatives (e.g., All, Information Technology, Technical Report on New Anti-Spam Techniques, Patent Application Draft on Mobile Ad Targeting, Market Projections for Worldwide Database Demand, IT Market Forecast (2008-2013); Industry Trends and Market Research (e.g., All, Information Technology, Product Launch Planning Meeting held on Feb. 22, 2007); Press Releases by Competitors (e.g., All, Information Technology, Oracle Announces Oracle 11i Beta Program); Product Illustrations for Nervana Discovery Spaces.

[3378] The following are illustrations of what a typical Nervana user will see when logged in to Nervana Discovery Spaces™:

[3379] My Discovery Space

[3380] General

[3381] My Nervana Networks

[3382] Nervana’s Global Life Sciences Network

[3383] Nervana’s Global Life Sciences Network (Pfizer Mirror)

[3384] Pfizer’s Global Innovation Network

[3385] American Cancer Institute’s Oncology Network

[3386] My People

[3387] John Smith

[3388] Philip Davies, Ph.D.

[3389] My Groups

[3390] Pfizer—Autoimmune Diseases

[3391] American Cancer Society

[3392] My Queries

[3393] Drugs used to treat infectious diseases

[3394] [+ Nervana’s Recommendations

[3395] [+ All

[3396] [+ Relevant Industry News

[3397] [+ Relevant Industry Blogs

[3398] [+ Relevant Web Pages

[3399] [+ Relevant Patents

[3400] [+ Relevant Patent Applications

[3401] [+ Relevant Scientific Literature

[3402] [+ Relevant People within the Social Network

[3403] [+ Relevant Information within the Social Network

[3404] [+ Relevant Projects

[3405] [+ Relevant Ideas

[3406] [+ Relevant Job Openings

[3407] [+ Relevant Resumes & Bios

[3408] [+ Relevant Meetings & Events

[3409] [+ Relevant Premium Scientific Content


[3410] Relevant Scientific Lecture Videos
[3411] Relevant Podcasts
[3412] Relevant Theses & Dissertations
[3413] Relevant Industry Events
[3414] Relevant Company Profiles
[3415] Relevant People Worldwide
[3416] Relevant Institutions with Expertise Worldwide
[3417] Relevant Regulatory Information
[3418] Relevant Clinical Trials
[3419] Relevant Drug Approvals
[3420] Relevant Links, Products & Services
[3421] Diagnostic techniques for cancer detection
[3422] Work by Eli Lilly on diabetes
[3424] Protein kinase inhibitors
[3425] My Projects
[3426] Technical Report on Inhibition of Cell Migration to Joint Tissues
[3427] Patent Application Draft on Chemical Compounds for Autoimmune Diseases
[3428] Meeting Report of the Annual Toxicology Conference in Chicago
[3430] My Ideas
[3431] Idea on a new Technique for Cell Signaling
[3432] Idea on Monoclonal Antibodies for Lymphoma
[3433] Idea on Toxicology Tests for COX Inhibitors
[3434] My Favorite Documents
[3435] Life Sciences Market Forecast (2008-2013)
[3436] Association of Biomarkers in Transporter Genes
[3437] My Meetings & Events
[3438] Product Launch Planning Meeting held on Feb. 22, 2007
[3439] Business
[3440] My Customers
[3441] Merck
[3442] GSK
[3443] Amgen
[3444] Genentech
[3445] My Competitors
[3446] Ipsen
[3447] Pozen
[3448] My Business & Research Partners
[3449] HR and Recruiting
[3450] My Job Openings
[3451] My Resumes & Bios
[3452] Nervana Discovery Spaces Daily Digest for Philip Rivers
[3453] Your Discovery Space has a total of 236 new items today. Here is the breakdown:
[3454] My People
[3455] John Smith: 12 new relevant items
[3456] Philip Watson, Ph.D.: 23 new relevant items
[3457] My Groups
[3458] Pfizer—Autoimmune Diseases: 211 new relevant items
[3459] American Cancer Society: 123 new relevant items
[3460] My Queries
[3461] Drugs used to treat infectious diseases: 14 new relevant items
[3462] Diagnostic techniques for cancer detection: 7 new relevant items including 2 new relevant people in the social network based on their backgrounds, and 3 relevant people in the network based on their projects, documents, and ideas.
[3463] My Projects
[3464] Technical Report on Inhibition of Cell Migration to Joint Tissues: 27 new relevant items including 5 new relevant people in the social network based on their backgrounds, and 6 relevant people in the network based on their projects, documents, and ideas.
[3465] Strategic Alliances
[3466] Nervana’s strategic alliances will balance market interests, customer interests, and revenue goals. The Company’s partner marketing approach will include strategic alliances with:
[3467] Content Providers: Nervana has formed relationships with content providers with the goal of improving access to the information researchers need. Currently, relationships are in place with the NIH (Medline), PatentCafe (Patents), Moreover (News and blogs), and Northern Light (regularly crawled Web content across all verticals). Future content providers will include premium scientific and business intelligence content aggregators.
[3468] Ontology Providers: Nervana has started to develop relationships with ontology providers and aggregators, including Taxonomy Warehouse and Intellisophic. These providers specialize in developing and maintaining industry-standard ontologies which will be then used to supplement Nervana’s internal ontologies and ontology automation tools as it expands outside Life Sciences.
[3469] Technology Licensing Partners: Nervana’s semantic platform has multiple applications in diverse vertical markets. Initially (on completion of funding), Nervana intends to aggressively pursue partnerships with vendors in the Bio-Medical space, in the following areas:
[3470] Consumer health search engines (e.g., Healthline, MedStory, RevolutionHealth, etc.)
[3471] Gene expression analysis vendors (e.g., GeneSifter)—to connect (using Nervana’s “Drag and Drop” technology) experimental genomics data to relevant information (patents, news, toxicology data, etc.) and people.
[3472] Pathway analysis vendors (e.g., Ingenuity and Teragenome)—to connect experimental pathway data to relevant information and people. See, for example, FIGS. 134 through 136
[3473] Large informatics software vendors: IBM Informatics (now investing $2 B a year in Life Sciences informatics) and Oracle Informatics for channel partnerships
[3474] Content providers and publishers focused on the Life Sciences space—including ProQuest, Thomson Pharma, Northern Light, and PatentCafe
[3475] Clinical Trials—Nervana is in talks with Clinical Trial Semantics, a service provider focusing on matching patients with ongoing clinical trials.
[3476] Electronic Health Record Aggregators—for semantic data management (search, discovery, clustering, matching, and analytics)
[3477] Evidence-Based Medicine—a new business process for matching patient health records to diagnostic information.
[3478] Products Nervana’s semantic platform, the Nervana System™, is optimized for research-driven industries including Life Sciences, Cosmetics, Food and Beverage and Specialty Chemicals—where efficient scientific discovery is vital to the success of the enterprise. Nervana is launching its latest platform and application suite, Nervana Discovery 4.0, combining the following custom-built solutions:
Nervana Semantic Search—much more intelligent search utilizing the power of semantics and ontologies

Nervana Discovery Agent—intelligent information agents for publishing and subscriptions

Nervana Social Discovery—for secure, context-aware, research-driven collaboration and social networking within and across organizational boundaries (facilitating the collaborative discovery and sharing of ideas and findings)

Nervana Discovery Integrator—for creating smart, semantic links between data embedded in scientific workbench tools (e.g., for pathways, gene expressions, etc.) and relevant internal and external data, patents, competitive intelligence, and experts

Nervana Smart Documents—for creating smart semantic links between internal and external content and the entire federation of relevant data (drug safety information, research, patents, experts, etc.) for much more powerful content management and discovery

Custom Third-Party Applications—for semantic categorization, tagging, and other ontology-based applications. See, for example, FIG. 137.

Nervana Social Discovery™ High-Level Model. See, for example, FIGS. 138-140.

Entity Framework
Object Model
Information Agent Framework
Inputs
Outputs
Object Model
Security
Message Pump
Content Framework
User-Generated
Relevant Content
Security
Presentation Framework
Ontology Framework
Performance and Scalability Initiative
Collaboration Framework
Annotations
Person-to-Person Messaging
Chat
Group Calendaring
Presence
Conferencing—voice, video, app-sharing;
A Discovery space can include the following embodiments, for example, but are not limited to:
Discovery Spaces has
DiscoveryNetwork
DiscoverySpace
DiscoveryItem
DiscoveryFolder ’contains discovery items
Person
Group
Topic
Project
General Project
Sales Campaign
Marketing Campaign
Recruiting Campaign
Litigation
Business Development Initiative
Press Releases
Corporate Documents, Brochures, and Whitepapers

Idea
Meeting
Favorite Document
Search
Question
Answer
Annotation
Text Annotation
Audio Annotation
Video Annotation
Customer
Competitor
Business Partner
Contact (“friend,” colleague, etc.)
DiscoverySource (e.g., Medline, etc.)
Automatically created Discovery Spaces ’e.g., institution entities that are collections of everyone from that institution; need not exist beforehand (patent)
Discovery Spaces
User Directory
User credentials
User profiles (person metadata, resume, bio, descriptive documents, picture, etc.)
Subscription information (mapped to discovery networks)
Discovery Networks contain discovery items
Discovery Store (Semantic Network)
Relationships—e.g., project contains documents, group contains people, etc.
Subscription information (user’s projects, topics, meetings, etc.)
Access control lists (creator, owner, can read, can edit, etc.)
Replicator ’replicates user profile state from master directory to discovery store
A Discovery network can be created independent of user directory and attaches itself to the directory, for example, Creator(s) of discovery network can be in the directory;
Discovery network can have access control rules;
Discovery network has ontology framework and content framework (points to KC’s)
XML document has list of subscribed discovery sources (servername+kc guid/name); wherein
Each discovery network has an accompanying information agent (agent crawls store for discovery item and auto-generates Discovery Spaces).
The Discovery Space is described by manifest XML—this then refers to published XML (RSS) per “semantic view”
Each discovery item can have a Discovery Space which in turn refers to relevant discovery items, for example FIGS. 142-146.
Manifest Links
All Relevant
Relevant Industry News
Relevant Industry Blogs
Relevant Industry Web Pages
Relevant Patents
Relevant Patent Applications
Relevant Scientific Literature
Relevant People within the Social Network based on their expertise
Relevant People within the Social Network based on institutions they attended
[3570] Relevant People within the Social Network based on their projects, documents, and ideas
[3571] Relevant Information within the Social Network
[3572] Relevant Searches within the Social Network
[3573] Relevant Questions within the Social Network
[3574] Relevant Answers within the Social Network
[3575] Relevant Projects within the Social Network
[3576] Relevant Ideas within the Social Network
[3577] Relevant Favorite Documents within the Social Network
[3578] Relevant Job Openings within the Social Network
[3579] Relevant Resumes & Bios within the Social Network
[3580] Relevant Meetings & Events within the Social Network
[3581] Relevant Premium Scientific Content
[3582] Relevant Videos
[3583] Relevant Podcasts
[3584] Relevant Theses & Dissertations
[3585] Relevant Industry Events
[3586] Relevant Company Profiles
[3587] Relevant People Worldwide
[3588] Relevant Institutions with Expertise Worldwide
[3589] Relevant Regulatory Information
[3590] Relevant Clinical Trials
[3591] Relevant Drug Applications
[3592] Relevant Drug Approvals
[3593] Relevant Links, Products & Services
[3594] Manifest Entry
[3595] Statistics
[3596] Predicate Guide
[3597] Link to New Results XML file (this is null if this is a people manifest)
[3598] Link to All Results XML file (this is null if this is a people manifest)
[3599] Link to All People XML file (this is null if this is an information manifest)
[3600] Manifest Builder
[3601] Builds manifest
[3602] Includes attribute annotations (ranking, etc.)
[3603] Imposes access control rules
[3604] Ranking Model
[3605] New Results
[3606] Breaking News
[3607] All News
[3608] New All Bets
[3609] All Results
[3610] Best Bets
[3611] Recommendations
[3612] All Bets
[3613] All People
[3614] Newsmakers: People linked with New Results
[3615] Experts: People linked with Best Bets
[3616] Interest Group: People linked with Recommendations
[3617] Relevant People: People linked with All Bets
[3618] The Information Agent config can include a list of KC replicas—it will then cycle through them for load-balancing, per KC set.
[3619] Each KC will have a priority queue managed by the IA
[3620] IA Message Pump Prioritization Scheme
[3621] New documents that have never had a manifest created
[3622] Newly modified documents and queries since the last manifest-generation time
[3623] User Profile info—resume, bios, profile docs
[3624] Queries
[3625] Favorite Documents (ranked by last time of manifest update)
[3626] Ideas (via contained documents, ranked by last time of manifest update)
[3627] Projects (via contained documents, ranked by last time of manifest update)
[3628] Meetings & Events (via contained documents, ranked by last time of manifest update)
[3629] Job Openings
[3630] Resumes & Bios
[3631] Collection discovery items will have Discovery Spaces showing aggregate information and links to contained items
[3632] Discovery Network Search Surface Manager
[3633] Exposes discovery items as HTML to be indexed by search engine at the application layer . . . HTML has URLs pointing to Discovery Spaces
[3634] Search engine at application layer must integrate discovery items from all subscribed networks
[3635] Ontology Automation Model
[3636] For example can be used to mine information. For example: Patents and patent applications by the relevant company or community
[3637] The company’s web site and press releases
[3638] Scientific publications by researchers in the company
[3639] User-Generated Content
[3640] TT-IDF ‘ High Frequency Terms ‘ Hook Variants based on Stemming ‘ Wikipedia Lookup to generate predicate and relationships. See, for example, FIG. 141.
[3641] User-Controlled Perspective Emphasis
[3642] User can select options to determine what information will be mined. For example,
[3643] General—everything
[3644] Domain-specific (industry-wide)
[3645] Specific to the community
[3646] Alternatively, software can generate three queries in a sequential SQML to represent different perspectives. This is then highlighted in the results.
[3647] HR TalentEngine™
[3648] A critical and growing need in recruiting and staffing is that of sourcing and ranking the best and most qualified candidates to ensure the highest caliber work force to any organization. Nervana’s TalentEngine™ is a powerful new software based business tool that provides HR managers the most cost effective means of managing critical staffing Discovery, Screening, and Ranking processes while significantly reducing costs typically incurred in identifying the best possible candidates from fragmented sources, domains, and databases.
[3649] This hosted “on-demand” service employs Nervana’s award winning artificial intelligence engine to automatically source resumes and curriculum vitae from fragmented sources including the internet, job boards, social networks, proprietary databases, and any targeted domain, and to match them to relevant positions. Resulting matches are ranked using novel and proprietary algorithms with unparalleled efficiencies (employing over one hundred variables available). TalentEngine™ Services assist HR managers to increase placement quality while streamlining associated workflows.
With Nervana's natural-language-processing technology a custom job or target profile can be submitted as query and the TalentEngine™ aggregates ideal resumes, curriculum vitae, and user profiles from multiple open and accessible domains (delivering both active and passive candidates). The system then builds an intelligent semantic index based on domain-aware ontologies and numerous other variables (standard and custom) and performs automated screening and ranking based on semantics or meaning, not on keywords! This helps ensure that a candidate's skills are matched in only the most relevant context, and also helps address the new common and misleading practice of 'keyword stuffing' where candidates often populate their resumes with keywords independent of their qualifications. The best matches are then periodically published, stored and made available to the user. This empowers users with a complete sole-source solution to effectively manage recruiting and staffing management of sales, administration, technologists, and engineering professionals.

TalentEngine™ provides a single platform tool that delivers its user the capability to leverage artificial intelligence to match criteria similar to human thought on a super computing scale, allowing HR Managers to focus on the most critical decisions and functions of HR processes. It guarantees human capable oversight (Quality Assurance and Control) across an expansive and fully automated set of Discovery, Screening, and Ranking processes that today can over stretch the precints of limited HR resources. Nervana TalentEngine™ providing HR Managers a paradigm shift to staffing workflow through the power of semantics and artificial intelligence.

There is a need for granular placement control, for example, taking into account characteristics of a job: skills relevance sub-model; skill-specific competencies; Expertise mining; Semantic relevance; Deep, ontology-based semantic analysis; Natural-language processing; Industry-specific competencies. Examples of information linked can include, in specific job: Industry rank of most recent company; Institution rank of most recent school; In classified domain; In all domains; Worked for top-ranked companies; In classified industry; In all industries; Schooled at top-ranked institutions; In classified industry; In all industries. Generic competencies can include: Generic indicators of achievement, Highest degree earned, Undergraduate GPA, graduate GPA, average GPA, etc.; Renowned, industry-agnostic awards (Rhodes Scholarships, MacArthur Fellowships, Nobel Prizes, etc.). Generic indicators of leadership can include information such as founded companies, community service, etc. Role-specific competencies can include information such as: Relevant concepts: "Sales," "Customer support," etc., Number of years of relevant experience (most recent job), Number of years of relevant experience (total), Key category-specific achievement metrics can include Goals met/exceeded, Sales quotas, Marketing metrics, etc., Number of training events/seminars, Number of awards, Role-specific awards, Publications, Patents, etc.

Experience-specific competencies can include, for example: Relevant concepts: "CEO," "Vice President," etc., Number of direct reports in last job, Average number of direct reports in last N jobs, Leadership certifications, Number of leadership awards/seminars, Red Flags Detection Submodel can include information such as: Time spent in last job, Longest time-span between jobs, Average time spent in all jobs, and gaps in work history and the like.

Advantages can include, for example:

- Increasing employees Draw: getting the most out of your advertising and posting budget. No more "blasting", No more missed prospects, Monitor multiple fragmented sourcing channels via an integrated platform; Increasing reach to the best qualified candidates; Discovering the best qualified talent across multiple fragmented touch-points; Pushing vs. pulling; and Reduce employer Recruiting Costs

- Drastically reduce labor costs by streamlining workflows and optimizing the use of human review.

- Get highly targeted, qualified candidates and minimize exposure to arduous "trial and error" keyword search, and resume-keyword-stuffing and other manipulation techniques.

- Other advantages include substantially shortening the time to identify and recruit the best qualified candidates in an extremely competitive labor market.

- The search system can use existing resumes, bios, or cover letters as natural-language queries to complement or accelerate the use of job descriptions and to bolster laser-like targeting. Automated Ranking and Bulls-Eye Scoring Techniques;

- Short list qualified candidate pools via statistical ranking by determining quantifiable variable summaries; and

- Can give Position & Industry specific custom or standard candidate scoring.

- In an exemplary embodiment, TalentEngine™Artificial Intelligence Components can include (See, e.g., FIG. 51):

- Overall Candidate Relevance: (e.g., Job Industry Relevance; Job Category Relevance; Job Experience Relevance;

- Job Skills Relevance;

- General Relevance;

- Red Flags; and

- Custom Relevance(s)).

- The software and system can be accessible to limited number of individuals and monetized as a service in the following way: Annual User Access License, Standard Edition, Professional Edition, Premium Edition, Custom Edition: Premium Edition. For example, a Standard Edition can include:

- Screening and Ranking only (customer-provided resumes, referrals, and career web sites) e.g., Email Reports, RSS Feeds, Secure Report-Hosting Portal, Search within Reports, and Report Diaries.

- The Professional Edition can include, for example:

- Discovery, Screening, and Ranking;

- Web (resumes), Free Job Boards, Subscription Job Boards, Social Networks, Career Web Site, and Referrals and Custom Databases.

- The Premium Edition can include, for example: The Professional Edition plus,Nervana Resume Database; and Relevant Blogs, News, Inventors and Scholars

- In an exemplary case study, a question is posed by pharmacists. Question facing P&G: “Find all chemical leads for bone diseases which are available for licensing.” In this context, traditional discovery methods suffer from overload.

- To search for? There are 308 bone diseases and 5740 chemical types. Additionally, data on bone diseases and chemical types are housed in different information silos. Secondly, Researchers have hit an information wall. The Combinatorial complexity of keyword searches would result in 18.3 million 2-keyword searches or 36.9 billion 3-keyword
searches. In an exemplary embodiment, the solution, the Nervana method bridges information silos and returns contextually relevant results. For example, client P&G ended up finding compounds it would have otherwise missed. In an exemplary embodiment, as a first step a profile is created (Federated sources; Federated domain-specific ontologies; Run single query; Natural language text or drag and drop documents; Results are semantically ranked; user views correlation and connections. In a preferred embodiment, there is . . . Integrated workflow, Alerts, Collaboration. Semantic search & discovery provides Semantic inference and reasoning. Including support for multiple ontologies, NLP-based semantic matching, Matching documents, user profiles, etc. Search and discovery is semantic, NOT keyword-based “more like this”. Combination of

[3676] NLP-contextual analysis-ontology-based reasoning, e.g., Matching documents to documents, people, diagnostic information, Clustering patient profiles/electronic health records, Matching patients with similar health symptoms, interests, etc. Matching users to people, physicians, experts, etc., enables entities to be smart publishers. Custom/ personalized queries (“channels”) and feeds

[3677] E.g., topics, documents, people, events, companies, clinical trials, etc.

[3678] Semantically indexed content

[3679] Life Sciences Patents (6 ontologies)

[3680] Life Sciences News (6 ontologies)

[3681] Life Sciences Web (6 ontologies)

[3682] Medline (6 ontologies)

[3683] Certified Life Sciences ontologies

[3684] Ontology Tools

[3685] Ontology Automation

[3686] Semantic Data Mining

[3687] Better ontology automation, name disambiguation

[3688] Semantically indexed annotations

[3689] Match annotations to ads, content, subscribers, experts

[3690] Analytics

[3691] Semantic integration with other Nervana properties

[3692] Dynamic linking/matching

[3693] Across ontology boundaries

[3694] Direct Technology Licensing

[3695] OEM Partnership

[3696] CTS becomes Nervana channel partner

[3697] Nervana Discovery 4.0

[3698] Information Overload Severely Impacting Productivity

[3699] The healthcare industry faces a critical challenge. Scientific complexity and the explosion of data results in INFORMATION OVERLOAD. For example, consider the following examples of combinations to overload:

[3700] Deciphering of human genome: 60x increase in the number of drug targets in the last decade; Volumes of clinical data; Digitization of health records; Litigation risk and environmental factors; Combinatorial complexity; Difficulty of identifying correlations between completely different technical disciplines and information sources. Semantic search can provide a Personalized Research Experience. See, for example, FIGS. 142 through 146. Flexible semantic filtering based on area of research gives a researcher the Ability to combine search filters to cross domain boundaries. Use Unique natural-language-processing technology; Utilize Contextual, semantic alert system. It is a Flexible Application; there is No tagging or other manual categorization.

[3701] Ontologically agnostic approach allows system to work across numerous domains and/or industries, for example: Available as hosted or enterprise application

[3702] Federation

[3703] Across physical and semantic boundaries

[3704] Internal databases, shares and other company content

[3705] External Life Sciences data sources

[3706] Subscriptions and feeds

[3707] Information overload is also a problem for advertisers. Up to 30% of new pages include search engine spam (Source: Microsoft Research). Additionally, there are Phony blogs, Phony doorway pages, Up to 64% of blog pings in English were spam (Source: Ebiquity group (February 2007)). 51% of Google blogspot blogs are spam.

[3708] There are 60 times as many blogs as 3 years ago (Source: Technorati)

[3709] 27.2 million blogs, 75K bloggers daily (Source: Technorati)

[3710] Search Engine Ad Networks are conflicted (Google AdSense™, Yahoo Publisher Network). Because Search Engines share revenues with publishers and therefore they make money either way. The end result is that Advertisers are getting fleeced. It is estimated that Up to 30% of contextual ad clicks are fraudulent (Source: Microsoft Research).

[3711] It is almost impossible for advertisers to control where their ads get placed. There are minimal tools available to advertisers to optimize this process and there is an Almost complete lack of control. Additional problems include, for example, Search engine black boxes, Ad matching quality control is a very hard problem, Natural-language processing problem (blogs, newsletters, etc.), Computationally complex, NP-hard.

[3712] No Site-specific targeting, Advertiser gets to choose which sites to place their ads.

[3713] Although Exclusion lists exist to purportedly help advertisers, The Burden is placed on advertisers to provide Google (and others) with exclusion lists, and therefore, Virtually no advertiser does this. Basically it is Outside advertisers’ core competency in Summary: Advertisers are on their own. Thirty percent of contextual-ad spend is fraudulent, there is a true impact on advertising spending. As a result, e.g., Click Fraud, Fake web pages, and Siphags. Another 20-30% of contextual-ad spending is poorly targeted. There is a complete Lack of semantics, Lack of comprehensive contextual matching tools, and Lack of policing tools.

[3714] There is an approximate $10B contextual ad market (this represents $50% of total online ad spend).

[3715] The Conclusion is that Up to $5 B in annual spend might be wasted.


[3717] Turning generally to FIGS. 148 through 151, in an embodiment of the system, the Nervana Ad Engine provides control for contextual advertising in Semantic, context-sensitive analysis. The system enables management of ad campaigns and ad pages, Semantic profile generation, and

[3718] Natural-language-processing and semantic matching.

[3719] Best Fit analyses, Web pages, Web sites, Blogs, Newsfeeds and bulletin boards, and Exclusion lists. The collaboration between publishers, Nervana, advertisers and search engines is represented in FIG. 148.

[3720] The system can significantly reduce costs due to improved targeting, fraud management, and more advertiser
control, which in turn results in higher ROI on ad spend, for example, by increased control of ad budgets, (e.g., ranked target sites and exclusion lists).

Provide "dial" for contextual targeting and optimization options based on budget constraints.

(e.g., add/remove sites to contextual network as budget changes, providing complete advertiser control and transparency); and improved brand management (e.g., control of where brand gets displayed).

An example of an ad placement provider is AdSense, with quarterly sales of $1.2 B. AdSense creates farms, e.g., Splogs, etc. It perpetuates problems, such as culprit keywords taken out of context, provides semantic mismatches, uses keyword stuffing of content pages based on AdWords bids, and used AdSense for Domains and Parked domain pages as AdSense Domains.

The problem is only going to get worse because of the growth of blogs, More web pages, RSS advertising (e.g., Feedburner), Spammers, Lower consumer spending and Tighter ad budgets.

While the preferred embodiment as well as alternative embodiments of the invention have been illustrated and described, as noted above, many changes can be made without departing from the spirit and scope of the invention. Accordingly, the scope of the invention is not limited by the disclosure of the preferred or alternative embodiments.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. A system for knowledge retrieval, management, delivery and presentation, implemented on at least one computer capable of presenting at least one semantic relationship as part of a search result that presents at least one document in response to a query, the computer system comprising a computer storage medium having a plurality of computer software components embodied thereon, the computer software components comprising:

- A knowledge indexing and classification component wherein information from both structured and unstructured information sources are semantically encoded to create a plurality of knowledge objects;
- A knowledge integration component to perform the steps of:
  - Creating a semantic network based on semantic associations between the plurality of knowledge objects having semantic encoded information;
  - Hosting domain-specific, episodic and contextual information;
  - Dynamically linking at least one knowledge object to domain-specific information creating a linkage network;
  - Maintaining the semantic attributes and dynamic linkage network of knowledge objects in a data store;
  - A semantic query processing component to perform the steps of:
    - Receiving at least one user input query for processing;
    - Extracting at least one semantic query based on user input query;
    - Inspecting the data store to determine at least one semantic relationship between the semantic query and the dynamically linked knowledge object in the linkage network based on one or more rules for determining the one semantic relationship;
    - Semantically linking the semantic query with the dynamically linked knowledge object in the linkage network to create a relational node;
    - Delivering a representation of the semantically linked relational node based on the user query to a client according to customizable user preferences.
- 4. A method for creating a semantic network of knowledge objects in a computer memory capable of storing at least one knowledge object having schema and semantic links and hosting domain-specific semantic information used to classify and categorize domain-specific information;
- Evaluating a schema of a first knowledge object;
- Obtaining domain-specific semantic information from a memory related to the first knowledge object schema if the schema of the first knowledge object lacks a domain-specific meaning; and
- Creating a semantic link between the first knowledge object and the domain-specific semantic information if the schema of the first knowledge object suggests association with domain-specific information.

2. A method for searching data products stored on a computer readable medium, implemented on at least one computer comprising:

- Building a natural language relationship of a plurality of data products forming a semantic linkage map, further comprising:
  - Analyzing the text within the plurality of data products based on a series of predefined ontologies to determine at least one semantic concept, the semantic concept built from analysis of the language, word patterns, and a context of the text within each data product, the semantic concept containing text not found within the data but inferentially related by connection in the semantic linkage map;
  - Creating semantic metadata using the determined semantic concepts;
  - Determining associations between the semantic metadata in the plurality of data products;
  - Applying a semantic ranking to the semantic metadata;
  - Indexing the ranked semantic metadata;
  - Linking the ranked semantic metadata to create a linkage map;
  - Receiving a search query of the built semantic linkage map further comprising:
    - Analyzing the search query based on the series of predefined ontologies to determine at least one semantic concept in the search query based on at least one of a meaning and a context, the semantic concept containing text not found within the data product but inferentially related by connection in the semantic linkage map;
    - Creating semantic metadata using the determined concepts;
    - Comparing the semantic metadata to the indexed and ranked semantic metadata; and
    - Displaying a list of data products in a ranked order based on the compared semantic metadata.

* * * * *