A computer system creates a plurality of objects using facts derived from electronic documents, each object including one or more facts describing an entity associated with the object. The system generates a value for an object of an unknown entity type, of the plurality of objects, by using an entity type model for a known entity type. The entity type model is based on a set of features of a plurality of objects of the known entity type, and the value indicates whether the object of an unknown entity type is of the known entity type. The system assigns the known entity type to the object of an unknown entity type in response to a determination that the value indicates the object of an unknown entity type is of the known entity type, and stores the object with the assigned entity type.
FIG. 2(a)
Bill Clinton was the 42nd President of the United States from 1993 to 2001.
<table>
<thead>
<tr>
<th>Object ID</th>
<th>Fact ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
</tr>
</tbody>
</table>

**FIG. 2(c)**
FIG. 2(e)
Build a binary classifier using objects of known entity type

Apply binary classifiers to each object with unknown entity type to obtain probability values.

FIG. 5b
534
Build a multi-class classifier

536
Apply the classifier to the object of unknown type to obtain probability values
542
Cluster all objects with a known entity type

544
Build a cluster prototype for each cluster

545
Refine cluster prototypes using objects of unknown entity type

546
Calculate similarity values between cluster prototype and objects of unknown entity type

FIG. 5d
ENTITY TYPE ASSIGNMENT

RELATED APPLICATIONS
[0001] This application is a continuation of U.S. patent application Ser. No. 11/781,891, filed Jul. 23, 2007, entitled “Entity Type Assignment”, which is incorporated by reference herein in its entirety.

TECHNICAL FIELD
[0002] The disclosed embodiments generally relate to fact databases or repositories. Specifically, the disclosed embodiments relate to methods of learning entity type models from objects in fact repositories and using these models to assign entity types to objects of unknown entity type in the fact repository.

BACKGROUND
[0003] Computational techniques such as information extraction are used to automatically identify and extract information in the form of facts. Information extraction can be performed on a variety of sources such as web pages to extract fact data. A set of facts collected from a source describing the same entity may be organized and can be stored as an object in a repository of facts.
[0004] Facts contain an attribute that describes an entity. The entity type of the entity is the type of real world thing the object represents (e.g., Person, Dog, Book, Movie). Entity type can be defined in a fact associated with the object. This entity type information is necessary to understand relationships between sets of facts associated with different objects. For example, an object with entity name “Hillary Clinton” is associated with a fact, “spouse Bill Clinton” and it is known that the attribute “spouse” always has a value with entity type “Person”. Knowing that an entity with entity name “Bill Clinton” has an entity type of “Person” enables the identification of a relationship between the two objects.
[0005] These relationships can be used to organize the sets of facts. Similarly, in cases where it is unclear whether or not a fact is valid, entity type is used in association with the attribute defined by the fact to assign a confidence to the fact. For example, an object with entity name “Bill Clinton” has an attribute “genre” with an associated value “political”. If we know “genre” is only used to describe a closed set of entity types such as “Book” and “Movie”, knowing that “Bill Clinton” is of entity type “Book” can provide a better confidence in that fact.
[0006] Often the entity type of the entity represented by the object is unknown or confounded for a number of reasons. For instance, entity type information may not be available in the source data. Due to inherent error in information extraction, entity type information may not be extracted. The entity type of an object may also be confounded by several objects having the same name. For example, an object with entity name “Bill Clinton” could be either an object with entity type “Person” or an object with entity type “Book”.
[0007] What is needed then is a computational method of assigning entity types to objects.

SUMMARY OF THE INVENTION
[0008] The above and other needs are met by methods, systems, and computer program products that assign entity types to objects with unknown entity type. Embodiments of the method comprise identifying a plurality of objects derived from electronic documents, each object including one or more facts describing an entity with which the object is associated, wherein each fact comprises an attribute. The method generates a set of features describing the facts included with the object for each of the plurality of objects. The method further identifies a subset of the plurality of objects of a known entity type. The method further generates an entity type model based on the sets of features generated for the objects in the identified subset. The method also generates a value based on an object of unknown entity type and the entity type model, wherein the value indicates whether the object of unknown entity type is of the known entity type. The method assigns the known entity type to the object of unknown entity type responsive to a determination that the value indicates the object of unknown entity type is of the known entity type and stores the object to which the known entity type was assigned.

BRIEF DESCRIPTION OF THE DRAWINGS
[0010] FIG. 1 shows a system architecture adapted to support one embodiment.
[0011] FIGS. 2 (a)-2(d) are block diagrams illustrating a data structure for facts within a repository of FIG. 1 in accordance with some embodiments.
[0012] FIG. 2(e) is a block diagram illustrating an alternate data structure for facts and objects in accordance with some embodiments.
[0013] FIG. 3 is a block diagram illustrating an entity type assignment engine according to one embodiment.
[0014] FIG. 4 provides an illustration of feature vector creation.
[0015] FIGS. 5 (a)-5(d) are flow charts illustrating several methods of performing entity type assignment.
[0016] The figures depict various embodiments of the present invention for purposes of illustration only. One skilled in the art will readily recognize from the following discussion that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the invention described herein.

DETAILED DESCRIPTION OF THE DRAWINGS
[0017] Embodiments are now described with reference to the figures where like reference numbers indicate identical or functionally similar elements.
FIG. 1 shows a system architecture adapted to support one embodiment. FIG. 1 shows components used to add facts into, and retrieve facts from, a repository. The system architecture includes a network, through which any number of document hosts communicate with a data processing system, along with any number of object requesters.

Document hosts store documents and provide access to documents. A document is comprised of any machine-readable data including any combination of text, graphics, multimedia content, etc. A document may be encoded in a markup language, such as Hypertext Markup Language (HTML), i.e., a web page, in an interpreted language (e.g., JavaScript) or in any other computer readable or executable format. A document can include one or more hyperlinks to other documents. A typical document will include one or more facts within its content. A document stored in a document store may be located and/or identified by a Uniform Resource Locator (URL), or any other appropriate form of identification and/or location. A component of a document is implemented by a computer system, and typically includes a server adapted to communicate over the network via networking protocols (e.g., TCP/IP), as well as application and presentation protocols (e.g., HTTP, HTML, SOAP, D-HTML, JAVA®). The documents stored by a host are typically held in a file directory, a database, or other data repository. A host can be implemented in any computing device (e.g., from a PDA or personal computer, workstation, mini-computer, or mainframe, to a cluster or grid of computers), as well as in any processor architecture or operating system.

FIG. 1 shows components used to manage facts in a fact repository. Data processing system includes one or more importers, a build engine, a service engine, and a fact repository (also called simply a "repository"). Each of the foregoing are implemented, in one embodiment, as software modules or programs executed by processor. Importers operate to process documents received from the document hosts, read the data content of documents, and extract facts (as operationally and programmatically defined within the data processing system) from such documents. The importers also determine the subject or subjects with which the facts are associated, and extract such facts into individual items of data, for storage in the repository. In one embodiment there are different types of importers for different types of documents, for example, dependent on the format or document type.

Janitors operate to process facts extracted by the importer. This processing can include but is not limited to, data cleansing, object merging, and data induction. In one embodiment, there are a number of different janitors that perform different types of data management operations on the facts. For example, one janitor may traverse some set of facts in the repository to find duplicate facts (that is, facts that convey the same factual information) and merge them. Another janitor may also normalize facts into standard formats. Another janitor may also remove unwanted facts from the repository, such as facts related to pornographic content. Other types of janitors may be implemented, depending on the types of data management functions desired, such as translation, compression, spelling or grammar correction, and the like.

Various janitors act on facts to normalize attribute names, and values and delete duplicate and near-duplicate facts so an object does not have redundant information. For example, we might find on one page that Britney Spears's birthday is "Dec. 2, 1981" while on another page that her date of birth is "Dec. 2, 1981." Birthday and Date of Birth might both be rewritten as Birthday by one janitor and then another janitor might notice that Dec. 2, 1981 and Dec. 2, 1981 are different forms of the same date. It would choose the preferred form, remove the other fact and combine the source lists for the two facts. As a result, one source page for this fact will contain an exact match of the fact while another source page will contain text that is considered synonymous with the fact.

The build engine builds and manages the repository. The service engine is an interface for querying the repository. The service engine's main function is to process queries, score matching objects, and return them to the caller but it is also used by the janitor.

The repository stores factual information extracted from a plurality of documents that are located on document hosts. A document from which a particular fact may be extracted is a source document (or "source") of that particular fact. In other words, a source of a fact includes that fact (or a synonymous fact) within its contents.

The repository contains one or more facts. In one embodiment, each fact is associated with exactly one object. One implementation for this association includes in each fact an object ID that uniquely identifies the object of the association. In this manner, any number of facts may be associated with an individual object, by including the object ID for that object in the facts. In one embodiment, objects themselves are not physically stored in the repository, but rather are defined by the set or group of facts with the same associated object ID, as described above. Further details about facts in the repository are described below, in relation to FIGS. 2(a)-2(d).

Some embodiments operate on the facts and/or objects in different orders than described above. For example, in one embodiment the importer provides facts directly to the build engine and/or repository. The janitors, in turn, operate on the facts and/or objects in the repository. It should also be appreciated that in practice at least some of the components of the data processing system will be distributed over multiple computers, communicating over a network. For example, the repository may be deployed over multiple servers. As another example, the janitors may be located on any number of different computers. For convenience of explanation, however, the components of the data processing system are discussed as though they were implemented on a single computer.

In another embodiment, some or all of document hosts are located on the data processing system instead of being coupled to the data processing system by a network. For example, importer may import facts from a database that is a part of or associated with the data processing system.

FIG. 1 also includes components to access the repository on behalf of one or more object requesters. Object requesters are entities that request objects from the repository. Object requesters may be understood as clients of the system, and can be implemented in any computer device architecture. As shown in FIG. 1, a first object requester is located remotely from system...
while a second object requester 154 is located in the data processing system 106. For example, in a computer system hosting a blog, the blog may include a reference to an object whose facts are in the repository 115. An object requester 152, such as a browser displaying the blog, will access data processing system 106 so that the information of the facts associated with the object can be displayed as part of the blog web page. As a second example, janitor 110 or other entity considered to be part of data processing system 106 can function as an object requester 154, requesting the facts of objects from the repository 115.

**0029** FIG. 1 shows that the data processing system 106 includes a memory 107 and one or more processors 116. The memory 107 includes the importers 108, janitors 110, build engine 112, service engine 114, and requester 154, each of which is preferably implemented as instructions stored in memory 107 and executable by processor 116. Memory 107 also includes the repository 115. The repository 115 can be stored in a memory of one or more computer systems or in a type of memory such as a disk. FIG. 1 also includes a computer readable medium 118 containing, for example, at least one of importers 108, janitors 110, the build engine 112, the service engine 114, the requester 154, and at least some portions of the repository 115. FIG. 1 also includes one or more input/output devices 120 that allow data to be input and output to and from the data processing system 106. It will be understood that embodiments of the data processing system 106 also include standard software components such as operating systems and the like and further include standard hardware components not shown in the figure for clarity of example.

**0030** FIG. 2(a) shows an example format of a data structure for facts within the repository 115, according to some embodiments. As described above, the repository 115 includes facts 204. Each fact 204 includes a unique identifier for that fact, such as a fact ID 210. Each fact 204 includes at least an attribute 212 and a value 214. For example, a fact associated with an object representing George Washington may include an attribute of “date of birth” and a value of “Feb. 22, 1732.” In one embodiment, all facts are stored as alphanumeric characters since they are extracted from web pages. In another embodiment, facts also can store binary data values. Other embodiments, however, may store facts as mixed types, or in encoded formats.

**0031** As described above, each fact is associated with an object ID 209 that identifies the object that the fact describes. Thus, each fact that is associated with a same entity (such as George Washington), will have the same object ID 209. In one embodiment, objects are not stored as separate data entities in memory. In this embodiment, the facts associated with an object contain the same object ID, but no physical object exists. In another embodiment, objects are stored as data entities in memory, and include references (for example, pointers or IDs) to the facts associated with the object. The logical data structure of a fact can take various forms; in general, a fact is represented by a tuple that includes a fact ID, an attribute, a value, and an object ID. The storage implementation of a fact can be in any underlying physical data structure.

**0032** FIG. 2(b) shows an example of facts having respective fact IDs of 10, 20, and 30 in the repository 115. Facts 10 and 20 are associated with an object identified by object ID “1.” Fact 10 has an attribute of “Name” and a value of “China.” Fact 20 has an attribute of “Category” and a value of “Country.” Thus, the object identified by object ID “1” has a name fact 205 with a value of “China” and a category fact 206 with a value of “Country.” Fact 30 has an attribute of “Property” and a value of “‘Bill Clinton was the 42nd President of the United States from 1993 to 2001.’” Thus, the object identified by object ID “2” has a property fact with a fact ID of 30 and a value of “Bill Clinton was the 42nd President of the United States from 1993 to 2001.” In the illustrated embodiment, each fact has one attribute and one value. The number of facts associated with an object is not limited; thus while only two facts are shown for the “China” object, in practice there may be dozens, even hundreds of facts associated with a given object. Also, the value fields of a fact need not be limited in size or content. For example, a fact about the economy of “China” with an attribute of “Economy” would have a value including several paragraphs of text, numbers, and perhaps even tables of figures. This content can be formatted, for example, in a markup language. For example, a fact having an attribute “original html” might have a value of the original html text taken from the source web page.

**0033** Also, while the illustration of FIG. 2(b) shows the explicit coding of object ID, fact ID, attribute, and value, in practice the content of the fact can be implicitly coded as well (e.g., the first field being the object ID, the second field being the fact ID, the third field being the attribute, and the fourth field being the value). Other fields include but are not limited to: the language used to state the fact (English, etc.), how important the fact is, the source of the fact, a confidence value for the fact, and so on.

**0034** FIG. 2(c) shows an example object reference table 210 that is used in some embodiments. Not all embodiments include an object reference table. The object reference table 210 functions to efficiently maintain the associations between object IDs and fact IDs. In the absence of an object reference table 210, it is also possible to find all facts for a given object ID by querying the repository 115 to find all facts with a particular object ID. While FIGS. 2(b) and 2(c) illustrate the object reference table 210 with explicit coding of object and fact IDs, the table also may contain just the ID values themselves in column or pair wise arrangements.

**0035** FIG. 2(d) shows an example of a data structure for facts within the repository 115, according to some embodiments, showing an extended format of facts. In this example, the fields include an object reference link 216 to another object. The object reference link 216 can be an object ID of another object in the repository 115, or a reference to the location (e.g., table row) for the object in the object reference table 210. The object reference link 216 allows facts to have as values other objects. For example, for an object “United States,” there may be a fact with the attribute of “president” and the value of “George W. Bush,” with “George W. Bush” being an object having its own facts in the repository 115. In some embodiments, the value field 214 stores the name of the linked object and the link 216 stores the object identifier of the linked object. Thus, this “president” fact would include the value 214 of “George W. Bush,” and an object reference link 216 that contains the object ID for the “George W. Bush” object. In some other embodiments, facts 204 do not include a link field 216 because the value 214 of a fact 204 may store a link to another object.

**0036** Each fact 204 also may include one or more metrics 218. A metric provides an indication of the quality of the fact. In some embodiments, the metrics include a confidence level and an importance level. The confidence level indicates the
likelihood that the fact is correct. The importance level indicates the relevance of the fact to the object, compared to other facts for the same object. The importance level may optionally be viewed as a measure of how vital a fact is to an understanding of the entity or concept represented by the object.

[0037] Each fact 204 includes a list of one or more sources 220 that include the fact and from which the fact was extracted. Each source may be identified by a Uniform Resource Locator (URL), or Web address, or any other appropriate form of identification and/or location, such as a unique document identifier.

[0038] The facts illustrated in FIG. 2(d) include an agent field 222 that identifies the importer 108 that extracted the fact. For example, the importer 108 may be a specialized importer that extracts facts from a specific source (e.g., the pages of a particular web site, or family of web sites) or type of source (e.g., web pages that present factual information in tabular form), or an importer 108 that extracts facts from free text documents throughout the Web, and so forth.

[0039] Some embodiments include one or more specialized facts, such as a name fact 207 and a property fact 208. A name fact 207 is a fact that conveys a name for the entity or concept represented by the object ID. A name fact 207 includes an attribute 224 of “name” and a value, which is the name of the object. For example, for an object representing the country Spain, a name fact would have the value “Spain.” A name fact 207, being a special instance of a general fact 204, includes the same fields as any other fact 204; it has an attribute, a value, a fact ID, metrics, sources, etc. The attribute 224 of a name fact 207 indicates that the fact is a name fact, and the value is the actual name. The name may be a string of characters. An object ID may have one or more associated name facts, as many entities or concepts can have more than one name. For example, an object ID representing Spain may have associated name facts conveying the country’s common name “Spain” and the official name “Kingdom of Spain.”

[0040] As another example, an object ID representing the U.S. Patent and Trademark Office may have associated name facts conveying the agency’s acronyms “PTO” and “USPTO” as well as the official name “United States Patent and Trademark Office.” An object does have more than one associated name fact, one of the name facts may be designated as a primary name and other name facts may be designated as secondary names, either implicitly or explicitly.

[0041] Another property fact 208 is a fact that conveys a statement about the entity or concept represented by the object ID. Property facts are generally used for summary information about an object. A property fact 208, being a special instance of a general fact 204, also includes the same fields (such as attribute, value, fact ID, etc.) as other facts 204. The attribute field 226 of a property fact 208 indicates that the fact is a property fact (e.g., attribute is “property”) and the value is a string of text that conveys the statement of interest. For example, for the object ID representing Bill Clinton, the value of a property fact may be the text string “Bill Clinton was the 42nd President of the United States from 1993 to 2001.” Some object IDs may have one or more associated property facts while other objects may have no associated property facts. It should be appreciated that the data structures shown in FIGS. 2(a)-2(d) and described above are merely exemplary. The data structure of the repository 115 may take on other forms. Other fields may be included in facts and some of the fields described above may be omitted. Additionally, each object ID may have additional special facts aside from name facts and property facts, such as facts conveying a type or category (for example, person, place, movie, actor, organization, etc.) for categorizing the entity or concept represented by the object ID. In some embodiments, an object’s name(s) and/or properties may be represented by special records that have a different format than the general fact records 204.

[0042] FIG. 2(e) is a block diagram illustrating an alternate data structure 290 for facts and objects in accordance with embodiments of the invention. In this data structure, an object 290 contains an object ID 292 and references or pointers to facts 294. Each fact includes a fact ID 295, an attribute 297, and a value 299. In this embodiment, an object 290 actually exists in memory 107.

[0043] As described above, an object may explicitly exist in the repository 115, or it may exist merely as a collection of facts with a common object ID. Reference is made to particular objects for the purposes of illustration; one of skill in the art will recognized that the systems and methods described herein are applicable to a variety of implementations and that such references are not limiting. When reference is made to a fact being associated with an object, it should be understood that in at least one embodiment a fact is associated with an object by sharing a common object ID with other facts. For example, a fact could be associated with an object of a given type by sharing a common object ID at least with a type fact indicating the given type (or as another example, with a category fact indicating a particular category of object). Furthermore, in various embodiments, facts and objects can be stored in a variety of structures, such as fact and/or object repositories. When reference is made herein to the repository 115, it should be understood that various embodiments may store facts and/or objects in a variety of data structures.

[0044] FIG. 3 is a block diagram illustrating an entity type assignment engine 300 according to one embodiment. The entity type assignment engine 300 improves the quality of knowledge contained within the repository 115 by assigning entity types to objects with unknown entity type. The entity type of an object is the category of the “real world thing” or entity the set of facts contained within the object describe. Example entity types include “Book,” “Movie,” “Country,” and “Person.” In some embodiments, an entity type is represented as a special kind of fact associated with an object.
According to the embodiment, an entity may have a single entity type or multiple entity types. Some pairs or sets of entity types may be specified as conflicting. For instance, an object cannot be both a “Book” and a “Person”. In some embodiments, non-conflicting entity types may be associated with an object. For instance, an object can have entity types for both a “Person” and a “Politician”.

The entity type assignment engine 300 identifies objects in the repository 115 with unknown entity type and assigns entity types to these objects. Once assigned an entity type, objects are either stored in the repository 115 in place of, or in addition to the original objects with unknown entity type. Depending on the embodiment, additional information may be added to the repository 115 such as a new or updated metric 218 representing improved quality of the facts 204. New or updated metrics 218 can include the confidence level and the importance level of the facts 104.

An embodiment of the entity type assignment engine 300 is implemented as a janitor 110. However, the entity type assignment engine 300 can also be implemented by other entities shown in FIG. 1 and/or by another entity altogether. FIG. 3 illustrates that the entity type assignment engine 300 includes multiple modules for performing various functions. Some embodiments include different and/or additional modules than the ones shown in FIG. 3. Moreover, the functions attributed to the modules herein can be distributed among the modules in different ways.

An entity type module 301 identifies the objects in the repository 115 with known entity types and generates entity type models representing each entity type. According to the embodiment, the entity type model module 301 may generate entity type models once or continuously regenerate entity type models to incorporate new objects with known or assigned entity type. Entity type models can be rebuilt continuously or at set time intervals such as every hour, day or week. Additionally, entity type models may be rebuilt based on the addition of new objects with known or assigned entity type to the repository.

Depending on the particular embodiment, different kinds of entity type models may be built by the entity type module 301. In one embodiment, a binary classifier is built for each entity type (e.g. “Dog”, “Book”, “Country”) providing an entity type model of each entity type relative to all other entity types. For example, a binary classifier for entity type “Dog” would classify objects as either “Dogs” or “Non-Dogs”. Binary classifiers can be created with any type of binary classification algorithm such as support vector machines, neural networks, maximum entropy, boosting, bagging, and online gradient descent learning. Other binary classification algorithms will be apparent to those skilled in the art.

In some embodiments, a multi-class classifier is created by the entity type module 301 created in place of, or in conjunction with, binary classifiers. As opposed to a binary classifier, a multi-class classifier creates a classification model or models for all entity types. Using the above example, a multi-class classifier creates models for and classifies an object as one of “Dog”, “Book” and “Country”. Though any multi-class classification algorithm may be employed in this method, some embodiments may include multi-class classification algorithms such as multi-class support vector machines, or maximum entropy.

In some embodiments, the entity type module 301 uses generative techniques to create entity type models. Generative techniques can incorporate unsupervised or semi-supervised approaches to learn models of data. Unsupervised approaches such as clustering, expectation maximization or self-organizing maps automatically group data based on pattern recognition. Semi-supervised approaches such as n-means clustering, or transductive support vector machines contain some portion of data which is labeled according to entity type which directs pattern recognition. Other techniques for generative learning will be apparent to those skilled in the art.

The groups of data created using generative techniques are combined by the entity type module 301 to form entity type models. One method of combining data from generative models is to generate a prototype for each group or cluster which can be used as an entity type model. A prototype may be created by combining group data in any suitable way such as averaging data, weighting data, probabilistic methods or any combination thereof.

The entity type assignment module 302 identifies objects of unknown entity type in the repository and assigns entity types to the objects of unknown entity type. The entity type assignment module 302 uses the classifiers and generative models computed by the entity type module 301 to generate values used to assign entity type. According to the entity type model used, the values can represent a similarity to an entity type, the probability of an object having an entity type or a confidence value that the object has an entity type.

According to the embodiment, the object of unknown entity type is assigned an entity type by the entity type assignment module 302 based on a single value generated based on a single entity type model such as a binary classifier. If the generated single value is greater or less than a specified cutoff value, the object of unknown entity type is assigned the entity type of the model.

In some embodiments, the entity type assignment module 302 compares the object of unknown entity type to multiple classification models representing multiple entity types and assigns entity type based on multiple values. In embodiments where multiple values are generated, the entity type with the highest or lowest value can be selected for assigning an entity to the object of unknown entity type. Again, a threshold or cutoff value can be applied to the highest or lowest value to increase the stringency.

Using cutoff values, the entity type assignment module 302 will not assign objects of unknown entity an entity type due to weak probability or similarity values. Similarly, objects with same or close values for conflicting entity types will be identified as having conflicting values and will not be assigned an entity type.

The object update module 304 uses information from the entity type assignment module 302 to update the objects in the repository 115 with assigned entity types. In embodiments where entity type is represented by a fact, a new fact specifying entity type is added to the object. The object update module 304 may perform additional actions to clean up the repository. According to the embodiment, the object update module 304 may perform these actions as part of the entity type assignment engine 300 or in communication with another janitor 110.

In addition to updating entity types, the object update module 304 can update other object information such as the metrics 218 associated with the facts 204. The metrics 218 contain values representative of the confidence and importance of the fact. Newly assigned entity types can be
used to reweigh other facts associated with the entities based on strong correlations between entity types and attributes 212 or values 214. In embodiments where entity type is represented as a fact 204, the confidence of the entity type fact can be increased due to entity type assignment.

In some embodiments, the object update module 304 identifies relationships between objects based on assigned entity type. Relationships can be represented as facts where the value 214 is another entity represented by another object, for instance, spouse Hillary Clinton Bill Clinton. If we know that the value 214 described by the attribute 212 ‘spouse’ is supposed to be a Person, then an object for Bill Clinton with entity type Person can be associated with the value.

The feature vector module 303 processes object data into feature vectors for use by the entity type assignment 302 and entity type model 301 modules. The feature vector module 303 creates several data points or features from the fact data associated with an object in the repository 115. Features can be represented in any way, such as text strings or alphanumeric symbols.

Features can incorporate any data contained in or associated with the fact or objects. Name facts 207 and property facts 209 are special types of facts which also may be incorporated into features. In most embodiments, feature vectors will contain features derived from data contained in the attribute 212 fields of the facts. Attributes 212 are represented within a field or slot in a feature vector.

Features can also incorporate value 214 data in conjunction with attribute 212 data. Attribute 212 and value 214 data can be combined or concatenated into one representative feature. Numerical value 214 data may be binned or discretized in feature generation. Additionally, values may be incorporated which represent a closed set of categorical values such as genre or color.

Additional annotation information can be associated with the attribute 212. Attribute annotations may include various data associated with the attribute such as the entity type of the value they describe. For instance, the attribute “dateOfBirth” can have an annotation of “Date” representing the type of the value it describes. Attribute and annotation information can be combined in a single feature.

In additional to information contained within facts, features can also represent any information associated with the fact. An example of this is the source 220 of the fact such as the Uniform Resource Locator or the website from which the fact was collected or extracted.

The feature vector can also contain a weight representing the significance of a feature. Features may be weighted according to the metrics 218 associated with facts from with they were derived. Similarly, features may be weighted by the source 220 or importer 108 used to generate the fact. Features may also be weighted according to the type of fact from which they were derived. Different types of facts may be assigned different weights. For instance, special types of facts such as entity type facts or name facts 207 may be given a higher weight.

FIG. 4 is a diagram illustrating feature vector creation according to one embodiment. In the embodiment illustrated, three objects 402 are translated into feature vectors 404. In the diagram, the facts 403 are translated into features 405 based on an attribute contained within a fact or the attribute and the value of the fact. Additionally, the source of the information is translated into a feature value.

In this embodiment, features 405 contain symbols ‘a’, ‘av’, and ‘s’ respectively used to denote that the feature contains an attribute 212, an attribute value 214, and the sources 220 of the facts 403 contained in the objects. The symbols are separated from the attribute with colons. In features containing values in addition to attributes, the values are further separated from the attributes by colons.

The embodiment illustrated provides an example of value discretization or “binning”. The translation of the values 214 for attribute 212 ‘height’ from facts into features transforms the fact values into an order to magnitude in meters. For instance, as George Bush is over 1 meter, the feature describing his height in magnitude of meters has a value of 0. Likewise, as the book Grapes of Wrath is less than 1 meter, the feature describing the height of the book in magnitude of meters has a value of −1.

FIGS. 5(a)-5(d) are flow charts illustrating steps performed by the entity type assignment engine 300 to assign entity types to objects in the repository 115 according to various embodiments of the present invention. Other embodiments perform different and/or additional steps than the ones shown in the figures described below. In addition, other embodiments perform the steps in different orders.

FIG. 5(a) is a flow chart illustrating the high level steps of assigning entity types according to an embodiment. Initially, the entity type model module 301 and the entity type assignment module 302 respectively identify 510 the objects in the repository with known and unknown entity types.

The feature vector module 303 generates 512 feature vectors for all the objects in the repository 115. The feature vector module identifies 512 facts associated with the objects and generates 512 features based on these facts. As discussed above, the feature vector module 303 generates 512 features that contain the attribute and value of the facts. The feature vector module 303 also generates 512 features that contain information associated with the facts such as annotations or the source of the facts.

The entity type model module 301 generates 514 entity type models based on the entity type of the model. Embodiments that use binary classifiers, multi-class classifiers and generative techniques to generate 514 entity type models are illustrated in FIGS. 5(f)-5(j) and are discussed in detail below.

The entity type assignment module 302 applies 516 the set of entity type models generated by the entity type model module 301 to the objects of unknown entity type in order to generate 516 values which indicate the probability that the object of unknown entity type is of an entity type. The entity type assignment module 302 selects the highest value of the generated values for entity type assignment 516. The entity type assignment module also compares 516 the values with known conflicting entity types. If the object of unknown entity has similar values for two conflicting types, the object will not be assigned 516 a value. Otherwise, if the highest value generated is above a certain threshold, the object will be assigned 516 the entity type of the model used to generate that value.

The object update module 304 stores 518 the objects with assigned entity type in the repository 115. The object update module creates 518 a new fact specifying entity type which is associated with the objects of unknown entity to which entity type was assigned in step 512.
The process of assigning entity types may be iterated to include the objects with assigned entity type in the entity type models.

In the embodiment illustrated in FIG. 5(b), binary classification models are generated for each entity type. Using the objects of known entity type, a binary classifier is built by the entity type module 301. In a specific embodiment, the binary classifier is a stochastic gradient descent learning algorithm and a binary classifier is built for each entity type contained within the objects of known entity type. In binary classification, objects are separated into two classes, objects of a specific entity type and the set of all objects that are not of the specific entity type. In stochastic gradient descent learning, the features associated with the objects of the two classes are plotted in space. This plot is used to learn a geometric function which separates the two classes based on the features. This geometric function is used by the entity type assignment module 302 to assign entity types to objects of unknown entity type.

The entity type assignment module 302 applies the function learned by the gradient descent algorithm for each entity type to each of the objects of unknown entity type. The entity type assignment module calculates a set of probability values for each object indicative of the probability the object is of each entity type (e.g. Dog 10%, Person 50%, Book 85%, Movie 92%).

FIG. 5(c) is a flow chart illustrating an embodiment using multi-class classifiers. Using the objects of known entity type, a multi-class classifier is built by the entity type module 301. In a specific embodiment, the multi-class classifier is built using a maximum entropy algorithm. Using the set of features associated with objects of different entity type, the maximum entropy algorithm learns an entity type model represented by a set of “constraints” or expected values for the features associated with each entity type that characterize the distribution of features for the entity type.

These learned models are then applied by the entity type assignment module 302 as a multi-class classifier to estimate the conditional distribution of an object with unknown entity type. From the conditional distribution, a value indicative of the probability the object of unknown entity type has the entity type of the learned model is generated.

FIG. 5(d) is a flow chart illustrating the use of generative models such as mixture models to create entity type models according to one embodiment. Objects with known entity type are used learn groups or clusters of data representing entity types. The entity type module 301 identifies all objects with known entity type and clusters them using, for example, simple bucketing. These techniques create a fixed number of clusters representing the number of different entity types. The resultant clusters are given a semantic label representing the entity type of the cluster 542.

Clusters with semantic labels are used to generate entity type cluster prototypes. The entity type cluster prototypes are represented as vectors of probabilities, each probability associated with a feature. Each probability is determined by looking up the number of objects that have the feature and determining the fraction of occurrences of the feature over the different entity types associated with the objects. For example, the attribute ‘mother’ is identified as occurring 30 times in objects with entity type “Person”, 14 times in objects with entity type “Dog” and 1 time in an object with entity type “Country”. These numbers are then normalized to sum up to one. Using the above example would provide the following values: 0.66 for Person, 0.31 for Dog and 0.03 for Country. These normalized numbers are then associated with the features in the entity type cluster prototypes to represent the probability of the feature given the entity type of the entity type cluster prototype.

The entity type cluster prototypes are used by the entity type assignment module 302 to calculate similarity values for objects of unknown entity type. The feature vectors created for the objects of unknown entity type are compared to the entity type cluster prototypes to calculate a similarity metric for each cluster prototype. The value of each feature associated with an object of unknown entity is identified as a probability in the cluster prototype. The identified probability values are then summed or combined to create a similarity value or “distance” between the object of unknown entity type and the entity type cluster prototypes. The similarity values are used to assign entity type to objects of unknown entity type based on the nearest cluster prototype as defined by the similarity values or “distances”.

Semi-supervised learning approaches use both labeled and unlabeled data to learn statistical models for classification. According to the embodiment, the objects of unknown entity type may be used to refine the cluster prototypes using semi-supervised learning such as expectation maximization. Expectation maximization is a two step learning algorithm, which is repeated until convergence.

In the expectation step, the feature vectors for each object are compared to the cluster prototypes using a set distance metric. The distances between the object of unknown entity type (e.g. object X) and the cluster prototypes are then normalized to 1 representing the posterior probabilities that object X has each entity type. Object X is now fractionally assigned to several different clusters. Consequently, the features contained within object X now fractionally occur in the different entity type cluster prototypes.

In the maximization step, the entity type cluster prototypes are re-generated as in step 544 incorporating the posterior probabilities. The fractional assignment of objects to entity type clusters is used to weight the features in the entity type cluster prototype generation.

The expectation and maximization steps are iterated until the cluster prototypes are stable, that is the probabilities associated with the features do not change over iterations between the two steps and the posterior probabilities associated with the objects of unknown entity type do not change. The refined entity type cluster prototypes are used by the entity type assignment module 302 to calculate similarity values for objects of unknown entity type.

Through the construction of entity type models based on generated feature vectors, entity types are assigned to objects with unknown entity type. Various embodiments of generating entity type models, generating feature vectors and assigning entity types are disclosed herein. The process of assigning entity type to objects with unknown entity type may be performed iteratively.

Reference in the specification to “one embodiment” or to “an embodiment” means that a particular feature, structure, or characteristic described in connection with the embodiments are included in at least one embodiment of the invention. The appearances of the phrase “in one embodiment” in various places in the specification are not necessarily all referring to the same embodiment.
Some portions of the above are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art.

An algorithm is here, and generally, conceived to be a self-consistent sequence of steps (instructions) leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic or optical signals capable of being stored, transferred, combined, compared and otherwise manipulated. It is convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. Furthermore, it is also convenient at times, to refer to certain arrangements of steps requiring physical manipulations of physical quantities as modules or code devices, without loss of generality.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to those quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or "determining" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system memories or registers or other such information storage, transmission or display devices.

Certain aspects of the present invention include process steps and instructions described herein in the form of an algorithm. It should be noted that the process steps and instructions of the present invention can be embodied in software, firmware or hardware, and when embodied in software, can be downloaded to reside on and be operated from different platforms used by a variety of operating systems.

The present invention also relates to an apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, application specific integrated circuits (ASICs), or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus. Furthermore, the computers referred to in the specification may include a single processor or may be architectures employing multiple processor designs for increased computing capability.

The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose inventors may be utilized with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the present invention as described herein, and any references below to specific languages are provided for disclosure of enablement and best mode of the present invention.

While the invention has been particularly shown and described with reference to a preferred embodiment and several alternate embodiments, it will be understood by persons skilled in the relevant art that various changes in form and details can be made therein without departing from the spirit and scope of the invention.

Finally, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.

What is claimed is:

1. A computer-implemented method of assigning an entity type to an object of an unknown entity type at a computer system, the computer system comprising one or more processors and memory storing one or more programs, the object stored in the memory, the method comprising:
   creating a plurality of objects using facts derived from electronic documents, each object including one or more facts describing an entity with which the object is associated, wherein each fact comprises an attribute;
   generating a value for an object of an unknown entity type, of the plurality of objects, by using an entity type model for a known entity type, wherein the entity type model is based on a set of features of a plurality of objects of the known entity type, and the value indicates whether the object of unknown entity type is of the known entity type;
   assigning the known entity type to the object of unknown entity type in response to a determination that the value indicates the object of unknown entity type is of the known entity type; and
   storing the object to which the known entity type was assigned with the assigned entity type.

* * * * *