(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 May 2011 (19.05.2011)

PCT

(10) International Publication Number
WO 2011/060231 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/US2010/056469

(22) International Filing Date:
12 November 2010 (12.11.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/261,277 13 November 2009 (13.11.2009) US
12/944,034 11 November 2010 (11.11.2010) US

(71) Applicant *(for all designated States except US)*: **TIGER-LOGIC CORPORATION** [US/US]; 25a Technology Drive, Irvine, CA 92618 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **DEXTER, Jeffrey, M.** [CA/US]; 176 Morning Glory, Rancho Santa Margarita, CA 92688 (US). **JOSHI, Aparna** [IN/US]; 3665 Benton Street #96, Santa Clara, CA 95051 (US). **GAMBHIR, Manish** [IN/US]; 2634 Britt Way, San Jose, CA 95148 (US). **GARISH, Ilesh** [IN/US]; 10669 Maplewood Road, #a, Cupertino, CA 95014 (US). **SMIK, Robert** [US/US]; 10831 North Oak Square, Cupertino, CA 95014 (US).

(74) Agents: **SUN, Yalei** et al.; Morgan Lewis & Bockius LLP, 2 Palo Alto Square, 3000 El Camino Real, Suite 700, Palo Alto, CA 94306 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK,

*[Continued on next page]*

(54) Title: METHOD AND SYSTEM FOR GROUPING CHUNKS EXTRACTED FROM A DOCUMENT, HIGHLIGHTING THE LOCATION OF A DOCUMENT CHUNK WITHIN A DOCUMENT, AND RANKING HYPERLINKS WITHIN A DOCUMENT

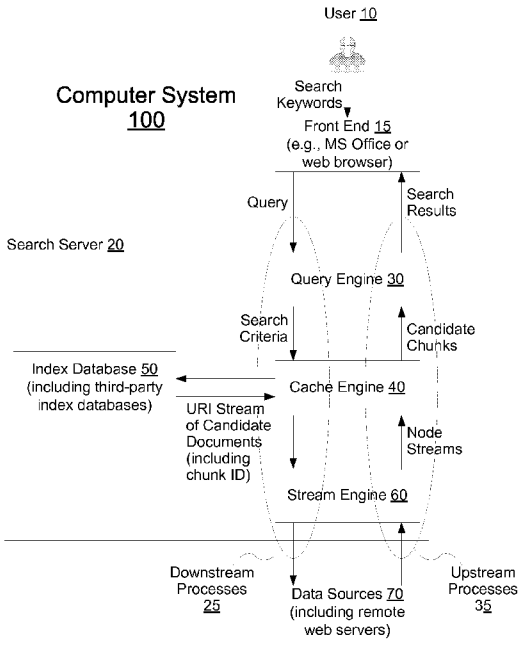



Figure 1

(57) Abstract: A system and method for grouping chunks, highlighting a chunk location within a document, and ranking hyperlinks of a document. A portion of a document including one or more hyperlinks to linked documents at respective data sources is displayed in a first window. In response to a search request including one or more search terms, one or more of the linked documents are requested from the respective data sources. When a respective linked document is received from a respective data source, it is determined whether the respective linked document includes chunks that match at least one of the search terms. If true, at least a subset of the chunks are displayed as a respective group in a second window only if a number of groups displayed in the second window is less than a predefined number of groups.

WO 2011/060231 A2

SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Published**:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

# METHOD AND SYSTEM FOR GROUPING CHUNKS EXTRACTED FROM A DOCUMENT, HIGHLIGHTING THE LOCATION OF A DOCUMENT CHUNK WITHIN A DOCUMENT, AND RANKING HYPERLINKS WITHIN A DOCUMENT

## FIELD OF THE INVENTION

**[0001]** The present invention relates generally to the field of information retrieval in a computer system, and in particular to systems and methods for grouping and displaying chunks, highlighting a location of a document chunk within a document, and ranking hyperlinks of a document.

## BACKGROUND OF THE INVENTION

**[0002]** The growth of information technology enables a user of a desktop or laptop computer to easily access information stored within a large number of documents at different locations such as the computer's local hard drive or a remote web server on the Internet. But quickly locating the information sought by the user within one or more documents remains a challenging task with today's information retrieval technologies.

**[0003]** In response to search keywords provided by a user, conventional web and desktop search engines typically return a list of document names with one or two sentences from each document that match the search keywords as search results. From the one or two matching sentences, the user often has trouble understanding the meaning of the search keywords in the context of the document. To determine whether the document has the user sought-after information, the user has no choice but to open the document using its native application (e.g., the Microsoft Office application if the document is a Word document) and repeat the process if the document does not have the information sought by the user.

**[0004]** There are multiple issues with this approach. First, opening a document using its native application is a time-consuming operation. Second, and more importantly, the native application does not highlight any particular portion of the document that may contain the user-provided search keywords. To locate any search keywords within the document, the

user has to do a new search of the document using a search tool of the native application. If the search tool can only look for multiple search keywords in exactly the same order (which is often the case), the user may end up finding nothing interesting in the document even if the document has a paragraph that contains the multiple search keywords but in a slightly different order. Alternatively, if the user limits the search to a subset of the multiple search keywords, many instances of the subset of search keywords may be in the document and the user could spend a significant effort before finding the document content of interest.

## SUMMARY

**[0005]** The above deficiencies and other problems associated with conventional search tools are reduced or eliminated by the invention disclosed below. In some embodiments, the invention is implemented in a computer system that has a graphical user interface (GUI), one or more processors, memory and one or more modules, programs or sets of instructions stored in the memory for performing multiple functions. Instructions for performing these functions may be included in a computer program product configured for execution by one or more processors.

**[0006]** Some embodiments provide a system, a computer readable storage medium including instructions, and a method for grouping and displaying chunks. A portion of a document is displayed in a first window, the document including one or more hyperlinks to linked documents at respective data sources. In response to a search request including one or more search terms, one or more of the linked documents are requested from the respective data sources in parallel. When a respective linked document is received from a respective data source, it is determined whether the respective linked document includes chunks that match at least one of the one or more search terms. In response to determining that the respective linked document includes chunks that match at least one of the one or more search terms, at least a subset of the chunks are displayed as a respective group in a second window only if a number of groups displayed in the second window is less than a predefined number of groups.

**[0007]** In some embodiments, in response to determining that the respective linked document does not include chunks that match at least one of the one or more search terms, linked documents that have not been previously requested are requested until the number of groups displayed in the second window is the predefined number of groups or until there are no more linked documents that have not been previously requested.

**[0008]** In some embodiments, after the predefined number of groups has been displayed, a respective page link corresponding to the groups displayed in the second window is displayed in the second window.

**[0009]** In some embodiments, after the predefined number of groups has been displayed, a next-page link that when clicked by a user is configured to perform the aforementioned operations for linked documents that have not been requested is displayed in the second window.

**[0010]** In some embodiments, in response to a user clicking on the next-page link, the aforementioned operations are performed for the linked documents that have not been previously requested.

**[0011]** In some embodiments, the number of linked documents that have not been previously requested is determined and the number of pages required to display groups of chunks for the linked documents that have not been previously requested is calculated based on a predefined number of linked documents per page. Page links corresponding to each page are then displayed in the second window.

**[0012]** In some embodiments, in response to a user clicking on a respective page link corresponding to a respective page of the second window, the requesting and the determining operations are repeated until all pages up to and including the respective page has the predefined number of groups or until there are no more linked documents that have not been previously requested.

**[0013]** In some embodiments, page links to pages that are no longer required to display groups of chunks for the linked documents are removed.

**[0014]** In some embodiments, a predefined number of linked documents are requested from the respective data sources in parallel.

**[0015]** In some embodiments, prior to receiving the linked documents from respective data sources, placeholders for each requested linked document are displayed in the second window.

**[0016]** In some embodiments, in response to determining that the respective linked document does not include chunks that match at least one of the one or more search terms, a placeholder corresponding to the respective linked document is removed.

**[0017]** In some embodiments, the predefined number of linked documents are requested in a predetermined order.

**[0018]** In some embodiments, the predetermined order is selected from the group consisting of: the order that the one or more hyperlinks to the linked documents appear in the document as displayed in the first window and the order that the one or more hyperlinks to the linked documents appear in a document object model for the document.

**[0019]** In some embodiments, the second window includes a search box for receiving user-provided search terms.

**[0020]** In some embodiments, the first window is a web browser window, and wherein the second window is a sidebar of the first window.

**[0021]** In some embodiments, the document is a first web page associated with a first website, wherein at least one of the linked documents is a second web page that is separate and distinct from the first web page.

**[0022]** In some embodiments, the document is an HTML document and the one or more hyperlinks include URLs to the linked documents.

**[0023]** Some embodiments provide a system, a computer readable storage medium including instructions, and a method for highlighting a location of a document chunk within a document. A portion of a first document is displayed in a first window and one or more chunks extracted from a second document are displayed in a second window, wherein the first document includes one or more hyperlinks to the second document. In response to receiving a selection of a user-selected chunk in the second window, a first element type defined in the second document that includes the user-selected chunk is identified. At least a portion of the second document is displayed in the first window. A second element type of the second document as displayed in the first window that has the same element type as the first element type and that includes the user-selected chunk is identified. The display of the first window and the second window is updated by: displaying a portion of the second document that includes the second element type including the user-selected chunk in the first

window, highlighting the second element type that includes the user-selected chunk in the first window, and highlighting the user-selected chunk in the second window.

**[0024]** In some embodiments, the first document is a first web page associated with a first website and the second document is a second web page that is separate and distinct from the first web page.

**[0025]** In some embodiments, one or more chunks are extracted from the second document in response to a search request including one or more search terms submitted by a user through the computer system.

**[0026]** In some embodiments, the one or more search terms are highlighted in the second window in a manner distinct from that of highlighting the user-selected segment.

**[0027]** In some embodiments, the first document is an HTML document, and wherein the one or more hyperlinks include at least one URL to the second document.

**[0028]** In some embodiments, prior to receiving the selection of the user-selected chunk in the second window and in response to a cursor hovering over the user-selected chunk in the second window, a portion of the first document that includes a URL to the second document including the user-selected chunk is identified and the URL of the first document is highlighted in the first window.

**[0029]** In some embodiments, prior to receiving the selection of the user-selected chunk in the second window and in response to a cursor hovering over the user-selected chunk in the second window, a displayed portion of the first document is updated in the first window to include a portion of the first document that includes the URL to the second document including the user-selected chunk.

**[0030]** In some embodiments, the first window is a web browser window and the second window is a sidebar of the first window.

**[0031]** Some embodiments provide a system, a computer readable storage medium including instructions, and a method for ranking hyperlinks of a document. A portion of a document is displayed in a first window, the document including one or more hyperlinks to linked documents at respective data sources. One or more chunks in the document are identified, wherein a respective chunk includes at least one of the one or more hyperlinks.

The one or more chunks are ranked based at least in part on properties of the one or more hyperlinks in the one or more chunks and properties of the one or more chunks. One or more linked documents are obtained from the respective data sources in accordance with respective rankings of the one or more chunks that include respective hyperlinks to the one or more linked documents. At least a portion of one of the one or more linked documents is displayed in a second window.

**[0032]** In some embodiments, the properties of a respective hyperlink are selected from the group consisting of: a target of the respective hyperlink, the number of words in anchor text for the respective hyperlink, and the meaning of the words in anchor text for the respective hyperlink.

**[0033]** In some embodiments, the properties of a respective chunk are selected from the group consisting of: a ratio of raw text to hyperlink anchor text in the respective chunk and a location of the respective chunk in the document.

**[0034]** In some embodiments, prior to obtaining the one or more linked documents from the respective data sources, a search request including one or more search terms provided by a user in a search box displayed in the second window is received.

**[0035]** In some embodiments, at least a portion of a respective linked document in the second window is displayed as follows. It is determined whether the respective linked document includes chunks that match at least one of the one or more search terms. In response to determining that the respective linked document includes chunks that match at least one of the one or more search terms, at least a subset of the chunks are displayed as a respective group in the second window.

**[0036]** In some embodiments, the document is a first web page associated with a first website, wherein at least one of the linked documents is a second web page that is separate and distinct from the first web page.

**[0037]** In some embodiments, the document is an HTML document and the one or more hyperlinks include URLs to the linked documents.

**[0038]** In some embodiments, at least a subset of the one or more chunks include nested chunks, and rankings of child chunks are aggregated into rankings of corresponding parent chunks.

**[0039]** Some embodiments may be implemented on either the client side or the server side of a client-server network environment.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0040]** The aforementioned features and advantages of the invention as well as additional features and advantages thereof will be more clearly understood hereinafter as a result of a detailed description of preferred embodiments when taken in conjunction with the drawings.

**[0041]** Figure 1 is a block diagram of an exemplary computer system that includes a front end, a search server including a query engine, a cache engine, an index database, and a stream engine, and one or more data sources in accordance with some embodiments.

**[0042]** Figure 2 is a flowchart illustrative of how the front end processes user-provided search keywords in accordance with some embodiments.

**[0043]** Figure 3 is a flowchart illustrative of how the query engine generates search criteria for the user-provided search keywords in accordance with some embodiments.

**[0044]** Figure 4 is a flowchart illustrative of how the cache engine produces a set of candidate document identifiers for the user-provided search keywords in accordance with some embodiments.

**[0045]** Figure 5 is a flowchart illustrative of how the stream engine processes candidate documents retrieved from different data sources in accordance with some embodiments.

**[0046]** Figure 6 is a flowchart illustrative of how the cache engine processes the candidate documents coming out of the stream engine in accordance with some embodiments.

**[0047]** Figure 7 is a flowchart illustrative of how the query engine identifies relevant chunks within the candidate documents in accordance with some embodiments.

**[0048]** Figure 8A is a flowchart illustrative of how the stream engine generates semantic data models for different types of documents in accordance with some embodiments.

**[0049]** Figure 8B is a flowchart illustrating a first embodiment of how the query engine identifies a relevant chunk within a node stream representing a candidate document.

**[0050]** Figure 8C is a flowchart illustrating a second embodiment of how the query engine identifies a relevant chunk within a node stream representing a candidate document.

**[0051]** Figure 9A is a flowchart illustrative of how the stream engine processes multiple candidate documents to identify candidate chunks in accordance with some embodiments.

**[0052]** Figure 9B is an exemplary HTML document to be processed by the stream engine as shown in Figure 9A in accordance with some embodiments.

**[0053]** Figure 10A is a block diagram illustrative of how a query mediator coordinates the query engine and the stream engine to identify chunks within a node stream representing a candidate document in accordance with some embodiments.

**[0054]** Figure 10B is a flowchart illustrative of how the stream engine divides the node stream into multiple sub-streams using a filter model in accordance with some embodiments.

**[0055]** Figure 11A is an exemplary XML document to be processed by the stream engine and the query engine in accordance with some embodiments.

**[0056]** Figure 11B is an exemplary XQuery to be applied to the XML document in accordance with some embodiments.

**[0057]** Figure 11C is a table of input sequences defined by the query engine in accordance with some embodiments.

**[0058]** Figure 11D is a flowchart illustrative of how the query engine processes node sub-streams at different input sequences in accordance with some embodiments.

**[0059]** Figure 11E is a block diagram illustrative of how a node stream corresponding to the XML document is divided into multiple node sub-streams by a finite state machine in accordance with some embodiments.

**[0060]** Figure 11F is a block diagram illustrative of the input sequences and their associated node sub-streams after the first candidate chunk in the XML document is processed in accordance with some embodiments.

**[0061]** Figure 11G is the search result of applying the XQuery to the node sub-streams derived from XML document in accordance with some embodiments.

**[0062]** Figure 12A is a flowchart illustrative of a first process of identifying one or more documents, each document having one or more chunks that satisfy user-specified search keywords, in accordance with some embodiments.

**[0063]** Figure 12B is a flowchart illustrative of a second process of identifying one or more document, each document having one or more chunks that satisfy user-specified search keywords, in accordance with some embodiments.

**[0064]** Figures 12C through 12J are screenshots of a graphical user interface on a computer display illustrative of features associated with the processes as shown in Figures 12A and 12B in accordance with some embodiments.

**[0065]** Figure 13A is a flowchart illustrative of a first process of identifying within a document one or more chunks that satisfy user-specified search keywords in accordance with some embodiments.

**[0066]** Figure 13B is a flowchart illustrative of a second process of identifying within a document one or more chunks that satisfy user-specified search keywords in accordance with some embodiments.

**[0067]** Figures 13C through 13G are screenshots of a graphical user interface on a computer display illustrative of features associated with the processes as shown in Figures 13A and 13B in accordance with some embodiments.

**[0068]** Figure 14 is a flowchart illustrative of a process of modeling a document and identifying within the document one or more chunks that satisfy user-specified search keywords in accordance with some embodiments.

**[0069]** Figure 15 is a flowchart illustrative of a process of customizing a search query based on user-specified search keywords in accordance with some embodiments.

**[0070]** Figure 16A is a flowchart illustrative of a process of displaying and re-using search results based on user instructions in accordance with some embodiments.

**[0071]** Figures 16B through 16J are screenshots of a graphical user interface on a computer display illustrative of features associated with the process as shown in Figure 16A in accordance with some embodiments.

**[0072]** Figure 17A is a flowchart illustrative of a process of finding and replacing text strings in connection with a set of search results based on user instructions in accordance with some embodiments.

**[0073]** Figure 17B is a flowchart illustrative of a process of finding and replacing text strings within a set of documents based on user instructions in accordance with some embodiments.

**[0074]** Figures 17C through 17E are screenshots of a graphical user interface on a computer display illustrative of features associated with the processes as shown in Figures 17A and 17B in accordance with some embodiments.

**[0075]** Figure 18A is a flowchart illustrative of a first process of narrowing search results based on user instructions in accordance with some embodiments.

**[0076]** Figure 18B is a flowchart illustrative of a second process of narrowing search results based on user instructions in accordance with some embodiments.

**[0077]** Figures 18C through 18D are screenshots of a graphical user interface on a computer display illustrative of features associated with the processes as shown in Figures 18A and 18B in accordance with some embodiments.

**[0078]** Figure 19 is a flowchart illustrative of a process of alternatively processing document node streams in accordance with some embodiments.

**[0079]** Figure 20 is a flowchart illustrative of a process of semantically and contextually annotating documents of different structures in accordance with some embodiments.

**[0080]** Figure 21A is a flowchart illustrative of a first process of screening matching chunks within a document based on predefined criteria in accordance with some embodiments.

**[0081]** Figure 21B is an exemplary HTML document illustrative of the process as shown in Figure 21A in accordance with some embodiments.

**[0082]** Figure 21C is a flowchart illustrative of a second process of screening matching chunks within a document based on predefined criteria in accordance with some embodiments.

**[0083]** Figure 21D is a screenshot of a graphical user interface on a computer display illustrative of features associated with the processes as shown in Figures 21A and 21B in accordance with some embodiments.

**[0084]** Figure 22A is a flowchart illustrative of a process of identifying contents matching a search request within a plurality of inter-related documents in accordance with some embodiments.

**[0085]** Figures 22B through 22D are screenshots of a graphical user interface on a computer display illustrative of features associated with the process as shown in Figure 22A in accordance with some embodiments.

**[0086]** Figure 23 is a block diagram of an exemplary document search server computer in accordance with some embodiments.

**[0087]** Figure 24 is a block diagram of an exemplary client computer in accordance with some embodiments.

**[0088]** Figure 25 is a block diagram of an exemplary system in accordance with some embodiments.

**[0089]** Figure 26 is a block diagram of an exemplary client computer in accordance with some embodiments.

**[0090]** Figure 27 is a block diagram of an exemplary server in accordance with some embodiments.

**[0091]** Figure 28 is a flowchart of a method for requesting and displaying chunks based on search terms, according to some embodiments.

**[0092]** Figure 29 is a flowchart of a method for displaying page numbers for linked documents that have not been searched yet, according to some embodiments.

**[0093]** Figure 30 is a screen shot of a web browser window including a search application and a first document, according to some embodiments.

**[0094]** Figure 31 is a screen shot the web browser window including the search application and the first document after a user initiated a search using the search application, according to some embodiments.

**[0095]** Figure 32 is a screen shot of the web browser window including chunks identified by the search application, according to some embodiments.

**[0096]** Figure 33 is a screen shot of the web browser window illustrating a cursor hovering over a chunk in the search application, according to some embodiments.

**[0097]** Figure 34 is a screen shot of the web browser window illustrating the cursor hovering over another chunk in the search application, according to some embodiments.

**[0098]** Figure 35 is a screen shot of the web browser window illustrating the cursor hovering over another chunk in the search application, according to some embodiments.

**[0099]** Figure 36 is a screen shot of the web browser window including the search application and a second document after a user clicked on a chunk in the search application, according to some embodiments.

**[00100]** Figure 37 is a screen shot of the web browser window illustrating the cursor hovering over a chunk in the search application, according to some embodiments.

**[00101]** Figure 38 is a screen shot of the web browser window including the search application and the first document after a user clicked on a next page link, according to some embodiments.

**[00102]** Figure 39 is a screen shot of the web browser window including chunks identified by the search application for a second page of linked documents, according to some embodiments.

**[00103]** Figure 40 is a flowchart of a method for highlighting chunks of linked documents in the web browser, according to some embodiments.

**[00104]** Figure 41 is a flowchart of a method for ranking hyperlinks to linked documents, according to some embodiments.

**[00105]** Figure 42 is a flowchart of a method for displaying chunks in the search application, according to some embodiments.

**[00106]** Like reference numerals refer to corresponding parts throughout the several views of the drawings.

## DESCRIPTION OF EMBODIMENTS

**[00107]** Reference will now be made in detail to embodiments, examples of which are illustrated in the accompanying drawings. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the subject matter presented herein. But it will be apparent to one skilled in the art that the subject matter may be practiced without these specific details. In other instances, well-known methods, procedures, components, and circuits have not been described in detail so as not to unnecessarily obscure aspects of the embodiments.

**[00108]** Figure 1 is a block diagram of an exemplary computer system 100 that includes a front end 15, a search server 20, and one or more data sources 70 in accordance with some embodiments. The front end 15 is a software application configured to receive and process input from a user 10 such as search keywords and present search results to the user 10. The search server 20 further includes a query engine 30, a cache engine 40, an index database 50, and a stream engine 60. The data sources 70 include storage devices such as file systems on hard drives accessible to the computer system 100 and remote web servers on the Internet.

**[00109]** At runtime, the front end 15 forwards the user-provided search keywords to the search server 20 in the form of a search query. In response, different components within the search server 20 work in concert to identify a set of candidate documents that matches the search keywords and retrieve the contents of the candidate documents from their respective locations at local and/or remote data sources 70. The different components within the search server 20 then search within the retrieved document contents for chunks that match the search keywords and return the identified chunks to the front end 15 in the form of search results.

**[00110]** In this application, a document is generally referred to as a data entity that has textual content, such as a Microsoft Office document, a plain-text document, a PDF

document, an email or text message, a web page, etc. A "candidate chunk" within a document is a portion of the document that is semantically and contextually regarded as a unit of textual content by one skilled in the relevant art. For example, within a Word document, a sentence, a paragraph, a table, a figure's caption, the document's title, header, and footer are candidate chunks. Similarly, a slide within a PowerPoint document, a bullet point within the slide, and a cell or a row within an Excel spreadsheet are also candidate chunks. A "chunk" or more specifically a "relevant chunk" served as part of the search results is a candidate chunk that satisfies the search keywords in accordance with predefined search criteria, e.g., if the candidate chunk includes at least one instance of one of the search keywords.

**[00111]** Figure 2 is a flowchart illustrative of how the front end 15 processes user-provided search keywords in accordance with some embodiments. After receiving the search keywords (201), the front end 15 generates a search query using the search keywords (203). Depending on the type of data to be processed by the search server 20, the search query can be written in different query languages such as structured query language (SQL) for relational databases or XQuery for XML data sources. The front end 15 then submits the search query to the query engine 30 within the search server 20 (205) for further processing.

**[00112]** Figure 3 is a flowchart illustrative of how the query engine 30 generates search criteria for the user-provided search keywords in accordance with some embodiments. After receiving the search query (302), the query engine 30 analyzes the query (304) and generates optimized search criteria (306). In some embodiments, the query engine 30 also generates one or more path filters from the search query (308). The path filters are derived from the user-provided search keywords and search options. The stream engine 60 employs the path filters to exclude document content that is not part of any candidate chunks. A more detailed description is provided below in connection with Figures 10 and 11. The query engine 30 submits both the search criteria and the path filters to the cache engine 40 (310).

**[00113]** In some embodiments, the query engine 30 generates an optimized execution plan for the query according to the capabilities of other components within the search server 20. For example, if the search query contains a predicate limiting the search to documents at the local hard drive that have been updated within the last two days, the query engine 30 has two options. One option is that the query engine 30 keeps the predicate to itself and waits to apply the predicate to the candidate chunks. In this case, the search server 20 (especially the

stream engine 60) may have processed more candidate documents than necessary. The other option is that the query engine 30 pushes the predicate down to the file system managing the local hard drive through the index database 50. In this case, only candidate documents that have been updated within the last two days are processed and the stream engine 60 is relieved from additional, unnecessary workload.

**[00114]** Figure 4 is a flowchart illustrative of how the cache engine 40 produces a set of candidate document identifiers for the user-provided search keywords in accordance with some embodiments. After receiving the search criteria from the query engine (401), the cache engine 40 submits the search criteria to the index databases 50. In some embodiments, the index databases include both a local index database and a remote index database. The local index database manages index information of documents at the local hard drive and the remote index database manages index information of documents at a remote document server. In some embodiments, the remote index database refers to the index database of a third-party search engine.

**[00115]** For given user search criteria, the cache engine 40 may search the local index database (403) if the user is looking for documents at the local hard drive or the remote index database (405) if the user is submitting a search request to a third-party search engine or both. From these index databases 50, the cache engine 40 receives respective search results (407), e.g., in the form of a set of document references such as URIs, and identifies a candidate document identifier within each search result (409). Note that a candidate document is a document that matches the search query at the document level, but may or may not at the chunk level. For example, a PowerPoint document that has slide A matching one search keyword and slide B matching another search keyword is a candidate document, but does not have any chunk matching the search query. In some embodiments, a universal resource identifier (URI) is used as a document identifier. Thus, documents at the local hard drive and remote web servers can be referenced in the same manner.

**[00116]** In some embodiments, the search results returned by the index databases 50 are ordered by the corresponding candidate documents' relevancy to the search query. Many well-known algorithms for determining a document's relevancy to a search query can be found in the classic book entitled "Automatic Information Organization and Retrieval" by G. Salton, McGraw-Hill, New York, 1968, which is incorporated here by reference in its entirety.

**[00117]** In some embodiments, a candidate document's relevancy is at least in part ranked by the past user activities on the candidate document. For example, a candidate document that has been recently accessed by the user, such as browsing, copying and updating, is given a higher rank than another candidate document that has never been accessed by the user before. In one embodiment, a candidate document's ranking score is determined by combining the following two pieces of information:

- The frequency of a search keyword in the document – For each keyword, the index database 50 may keep information such as a total count of occurrences of the keyword in a number of documents and a per-document count of the occurrences of the keyword. By combining the frequencies of different search keywords within the same document, a basic ranking score of the document is computed using a generic inverse frequency algorithm.

- The personalized usage weight of the document – A respective number of points are assigned to each operation the user applies to the document. For example, the preview operation of a particular document is given two points, the re-use of content from the previewed document is given three points, and the re-use of a specific chunk within the document is given four points. The total points assigned to the document, when compared against the total points allocated for the corresponding document type, yields a personalized ranking score for the document, which may be combined with the aforementioned basic ranking score to generate a customized ranking score for the document.

**[00118]** In some embodiments, a document's relevancy to a search query is not solely determined at the document level but is, at least in part, determined at the chunk level. For example, the index database 50 may maintain information about the locations of candidate chunks within each document as well as the distinct ranking information of different candidate chunks within the same document relative to different search keywords, thereby making it possible to return the relevant chunks within the same document in an order consistent with their respective chunk-level ranking scores.

**[00119]** The cache engine 40 submits a set of candidate document identifiers and path filters, if any, generated by the query engine 30 to the stream engine 60 for further processing (411).

**[00120]** For illustration, the aforementioned processes in connection with Figures 2 through 4 are collectively referred to as the "downstream processes 25" as shown in Figure 1. The input to the downstream processes is a search request including one or more search keywords and its output is a set of candidate document identifiers that identify candidate documents satisfying the search keywords. For example, a document is deemed to be a candidate document if it includes at least one instance of each search keyword. But the fact that each search keyword has a match in a candidate document does not guarantee that the candidate document has a chunk relevant to the search request.

**[00121]** As noted above, identifying a chunk within a document requires semantic information about the document. Such information is not available in the index database 50. To find out whether a candidate document has any relevant chunk, the search server 20 needs to retrieve the document and analyze the document's structure and content to understand the relationship between the document's structure and the document's content. The processes of retrieving a document, determining whether the document has any relevant chunks, and identifying and returning these relevant chunks to the requesting user are collectively referred to as the "upstream processes 35" as shown in Figure 1.

**[00122]** Figure 5 is a flowchart illustrative of how the stream engine 60 processes candidate documents retrieved from data sources 70 in accordance with some embodiments.

**[00123]** After receiving the candidate document identifiers and optional path filters from the cache engine (502), the stream engine 60 starts retrieving the candidate documents identified by the document identifiers from respective data sources, including retrieving some candidate documents from local data sources (504) and/or retrieving some candidate documents from remote data sources (506). In some embodiments, local data sources include any storage devices affiliated with the computer system 100, such as hard disk and CD/DVD drives, and remote data sources include any storage devices that can be accessed by the computer system 100 through wired and/or wireless network, such as a web server on the Internet and/or a network storage device on the Intranet.

**[00124]** In some embodiments, a specific user instruction may limit the document search to local or remote data sources. As shown in Figure 16B, if the user specifies that the type of the documents to be searched are Word documents, the stream engine 60 will retrieve only Word candidate documents from the local data source such as the local file system. For

each candidate document identifier, the stream engine 60 submits a request for the corresponding candidate document to the file system and waits for the file system to return the candidate document. But if the user clicks the checkbox next to a web source such as "Source A," the stream engine 60 will retrieve the candidate documents identified by Source A from their respective remote web hosts. For example, if the candidate document is an HTML document hosted by a web server, the stream engine 60 submits an HTTP request to the web server for the HTML document and waits for an HTTP response including the HTML document from the web server. In some embodiments, the user instruction may explicitly or implicitly request that candidate documents be retrieved from both local and remote data sources.

**[00125]** In some embodiments, the stream engine 60 submits multiple requests for different candidate documents in parallel or sequentially to the respective targeted data source(s). The candidate documents are then retrieved from the respective data sources and processed by the stream engine 60 in parallel or sequentially. For illustration, the following description in connection with Figure 5 focuses on a single candidate document. But this by no means limits the present application to processing documents one by one sequentially. As will become more apparent in connection with Figures 9A through 9B below, it is more efficient to process multiple candidate documents from different data sources in parallel in some embodiments.

**[00126]** Referring again to Figure 5, the stream engine 60 performs the following operations on each candidate document retrieved from a data source:

**[00127]** 1. Convert the candidate document into a node stream (508);

**[00128]** To reduce the computer system 100's response latency, the stream engine 60 starts converting immediately after receiving a portion of the candidate document, without waiting for the entire candidate document to be retrieved. A more detailed description of the conversion is provided below in connection with Figure 8A.

**[00129]** 2. Identify candidate chunks in the node stream (510);

**[00130]** As noted above, a candidate document includes one or more candidate chunks. A candidate chunk within the document, if identified as satisfying the user-specified search keywords, is usually more relevant to the user's search interest and therefore more useful to

the user. A more detailed description of this operation is provided below in connection with Figures 8A and 9A.

**[00131]** 3. Apply the optional path filters to the node stream (512); and

**[00132]** For a user-specified search query, certain portions of the node stream are potentially relevant and other portions are completely irrelevant. It could be an efficiency gain if these irrelevant portions are excluded from further consideration. The path filters generated by the query engine (operation 308 in Figure 3) can be used to divide the node stream into multiple node sub-streams, thereby eliminating the irrelevant portions of the node stream. In some embodiments, this procedure is optional if the query engine 30 generates no path filter. A more detailed description of the conversion is provided below in connection with Figures 10A-10B and 11A-11G.

**[00133]** 4. Submit the node stream (or sub-streams) to the cache engine (514).

**[00134]** After performing the operations above, the stream engine 60 submits the node stream or sub-streams to the cache engine 40. As will be explained below in connection with Figure 6, the cache engine 40 may or may not do anything depending on whether it needs to index the document represented by the node stream. If it does, the cache engine 40 invokes the index database 50 to generate new indexes or update existing indexes for the document. Otherwise, the cache engine 40 simply forwards the node stream or sub-streams to the query engine 30 for further processing, which is provided below in detail in connection with Figures 8A-8C and 11A-11G.

**[00135]** Figure 6 is a flowchart illustrative of how the cache engine 40 processes the candidate documents coming out of the stream engine 60 in accordance with some embodiments.

**[00136]** After receiving the node stream or sub-streams corresponding to a candidate document (601), the cache engine 40 performs different operations based on the type and status of the candidate document as well as its destination. For example, if the candidate document is a Word document found in the local hard drive of the computer system 100 and has not been indexed or its indexes are deemed stale, the cache engine 40 will request that the index database 50 generate new indexes or update existing indexes for the candidate document (603). Depending on whether the document is requested by an end user through

the front end 15 or a software agent monitoring the index database 50, the cache engine 40 may or may not return the node stream to the query engine 30 for further processing (605).

**[00137]** If the candidate document is an HTML document at a remote web server, which is identified through a third-party document source, it may be optional to index the HTML document. If so, the node stream or sub-streams will be returned to and further processed by the query engine 30 to determine whether it has any relevant chunk (605).

**[00138]** In sum, in some embodiments the cache engine 40 plays a relatively lightweight role in the upstream processes 35 especially if the candidate document is retrieved from a remote data source to satisfy an end user's search request and the computer system 100 does not intend to index the document. Therefore, some of the embodiments below assume that the stream engine 60 directly communicates with the query engine 30 for clarity.

**[00139]** Figure 7 is a flowchart illustrative of how the query engine 30 identifies relevant chunks within the candidate documents in accordance with some embodiments.

**[00140]** Upon receipt of the node stream or sub-streams (e.g., path filtering at the stream engine 60) (702), the query engine 30 traverses the node stream and compares the candidate document's content with the user-specified search keywords. If a match is found, the query engine 30 identifies the candidate chunk within the document corresponding to the match as one relevant chunk (704) and returns the identified chunk to the front end 15 to be displayed to the end user (706).

**[00141]** In some embodiments (as shown Figure 7), the query engine 30 returns any relevant chunk to the front end 15 as soon as the chunks are identified and this process repeats until the query engine 30 completely traverses the candidate document (708). In some other embodiments, the query engine 30 defers returning any chunk to the front end 15 until a more specific relevant chunk is found in the node stream. A more detailed description of these two approaches is provided below in connection with Figures 8B and 8C, respectively.

**[00142]** As noted above, candidate documents arriving at the stream engine 60 are each converted into a node stream. The node stream is an instance of a data model of the corresponding candidate document. For example, the XQuery data model of an XML

document is a tree of nodes. The types of the nodes that appear at different hierarchical levels of the tree include: document, element, attribute, text, namespace, processing instruction, and comment. Any node in the tree has a unique node identity. The data model not only preserves the XML document's entire content but also has metadata derived from sources such as XML tags for identifying candidate chunks subsequently.

**[00143]** Unfortunately, not all candidate documents are structured like an XML document. For example, a plain-text document is completely unstructured such that it does not have any metadata embedded therein defining a hierarchical structure for the document. Without any pre-processing, a node stream corresponding to the plain-text document loses the semantic information intrinsic in the content distribution of the document such that it is difficult to identify any chunk such as paragraph, headline, or title, within the node stream to satisfy a search query. PDF documents have similar problems that make it challenging to find relevant chunks within a PDF document.

**[00144]** Between the structured XML documents and the unstructured plain-text documents are semi-structured documents such as HTML documents. Unlike the plain-text document, an HTML document has a hierarchical structure defined by metadata embedded in the HTML document. But the metadata in the HTML document usually does not have a deterministic relationship with the content data as the metadata in an XML document has. The same HTML tag can be used purely for web page layout purpose at one location while carrying a semantic connotation at another location within the same document.

**[00145]** To expedite the upstream processes and accommodate more types of documents in the future, it is desired to have a unified approach such that different types of documents are processed in the same manner.

**[00146]** Figure 8A is a flowchart illustrative of how the stream engine 60 generates semantic data models for different types of documents in accordance with some embodiments.

**[00147]** After receiving the raw data of a candidate document, the stream engine 60 transforms the raw data into an instance of a data model of structured or semi-structured data (801). In some embodiments, this operation is straightforward if the candidate document is already a structured document like a Microsoft Office 2007 document. In some other embodiments, this operation is necessary if the candidate is a plain-text document without

any structure-related metadata. In this case, the stream engine 60 may insert metadata into the document that defines a hierarchical structure for the document's content.

**[00148]** Based on the class of the raw data (803), the stream engine 60 then performs different sets of operations to the data model instance generated previously. For example, as noted above, the candidate document may be classified into one of three categories:

- unstructured documents (805-A) such as plain-text and PDF;
- semi-structured documents (805-B) such as HTML and RTF; and
- structured documents (805-C) such as XML and Office 2007.

**[00149]** For an unstructured document, the stream engine 60 applies a set of natural language and common formatting based heuristic rules to separate text within the document into separate candidate chunks (807). For example, one heuristic rule for identifying paragraphs stipulates that any two text segments separated by symbols such as an end-of-line (EOL) character or a blank line correspond to at least two separate paragraphs. Another heuristic rule stipulates that a text segment matching a predefined text pattern is deemed to be a candidate chunk. Consider the following text segment that has two hyphens, one at the start of a new line:

- This is a bullet list.
- What about a page number?

In this case, each line by itself may be a candidate chunk (although it may or may not be deemed to be a paragraph). The fact that the two lines have the same text pattern, i.e., a hyphen at the start of a new line followed by a text string, may indicate that the entire text segment is a candidate chunk at one level of the document's hierarchical structure and each line is also a candidate chunk at a lower level of the hierarchical structure.

**[00150]** Similarly, for a semi-structured document, the stream engine 60 has another set of heuristic rules based on the type of the semi-structured document (809). For a node stream corresponding to an HTML document, the stream engine 60 identifies candidate chunk break nodes within the node stream both dynamically and statically.

**[00151]** For example, the <p> tag defines a paragraph within the HTML document and it is deemed to be a candidate chunk break node. Whenever the <p> tag appears in an HTML

document, the subsequent document segment following this <p> tag and before another <p> tag is identified as a candidate chunk.

**[00152]** Note that there are many ways of identifying chunk break nodes within a semi-structured document known to one skilled in the art. In some embodiments, the stream engine 60 applies different sets of customized heuristic rules to different types of documents. For a structured document or a semi-structured document for which there is no customized solution, the stream engine 60 assumes that there is a strongly-deterministic relationship between the document's content and the document's metadata and a generic set of rules is applied to the data model instance to identify possible candidate chunks in the candidate document.

**[00153]** By traversing the node stream, the stream engine 60 generates a data model instance for the candidate document that includes semantic annotation (811). Subsequently, the semantically-annotated node stream is fed into the query engine 30. The query engine 30 then applies a search query to the node stream to identify among the candidate chunks any that satisfy the search query.

**[00154]** As noted above in connection with Figure 7, the query engine 30 does not have to wait until it traverses the entire node stream before returning any relevant chunk to the front end 15. Below are two embodiments of how the query engine 30 returns identified chunks after a respective condition is met and before the entire node stream is traversed.

**[00155]** Assume that the search query has two keywords, "raining" and "data," and the exemplary candidate document is as follows:

<c0>
It's **raining** outside ...
<c1>
For XML-based **data** management,
**Raining Data** is your choice.
</c1>
</c0>

**[00156]** Figure 8B is a flowchart illustrating a first embodiment of how the query engine identifies a relevant chunk within a node stream representing a candidate document.

**[00157]** The query engine 30 starts the search after receiving a node stream corresponding to the candidate document above (821). If no more nodes are in the node stream (823, no), the query engine 30 assumes that it has completely traversed the node stream and the search stops (825). Otherwise, the query engine 30 processes the next node in the stream (827).

**[00158]** Before any further processing, the query engine 30 checks whether it is in the middle of accumulating nodes (829). In some embodiments, the query engine 30 begins accumulating nodes after it encounters the chunk break node of the first candidate chunk in the node stream. In this example, the chunk break node of the first candidate chunk is the <c0> tag, which is the first node in the stream, and the accumulation has not started yet (829, no).

**[00159]** Next, the query engine 30 checks whether the node is a text node (839). Since the <c0> tag is not a text node (839, the query engine 30 updates the current node path to be "/c0" (841) and checks whether the current node is part of a candidate chunk (843). Because the <c0> tag is the chunk break node of the first candidate chunk (843, yes), the query engine 30 marks the current node path as corresponding to a candidate chunk (845) and then starts node accumulation immediately (847).

**[00160]** Following the <c0> tag node, the next node to be processed by the query engine 30 is a text node including "It's raining outside …" In this case, because the accumulation has already started (829, yes), the query engine checks if the text node is part of a relevant chunk (831). But since no search keyword has been matched so far (831, no), the query engine 30 accumulates the text node (837). Because this is a text node (839, yes), the query engine 30 then checks whether it is in a candidate chunk (849).

**[00161]** In this case, the text node is in a candidate chunk (849, yes). The query engine applies the search query to the text node (851). But because only the keyword "raining" finds a match in the text string, which is a partial match of the search query, no relevant chunk has been found yet (853, no) and the query engine 30 returns to process the next node in the sub-stream (823). In some embodiments, the query engine 30 records the partial match result for subsequent use.

**[00162]** When the query engine 30 receives the second text node including the text string "For XML-based data management," it repeats the same processing steps 827 through

853 described above. In this case, because the two text nodes in combination match both keywords, a relevant chunk and its associated node path "/c0/c1" are identified (855). Next, the query engine 30 processes the third text node including the text string "Raining Data is your choice." Because the third node is already in a relevant chunk (831, yes), the query engine 30 checks whether the relevant chunk is completed (833). In some embodiments, a chunk is completed if the query engine encounters a node including the end tag of a candidate chunk, e.g., </c0> or </c1>.

**[00163]** In this case, because the query engine 30 has not seen any end tag yet (833, no), the process continues and the second and third text nodes in combination also match the two keywords because both the second and third text nodes are within the second candidate chunk (<c1>, </c1>), which is a descendent of the first candidate chunk (<c0>, </c0>). In some embodiments, if there is a hierarchical relationship between multiple relevant chunks, the query engine 30 first completes the relevant chunk at the lowest level, which is also referred to as the more specific relevant chunk, and then outputs this more specific relevant chunk to the front end 15 (835). In this example, the more specific relevant chunk is

<c1>
For XML-based **data** management,
**Raining Data** is your choice.
</c1>

**[00164]** Note that the query engine 30 does not necessarily stop after outputting the more specific relevant chunk (835). Rather, the query engine 30 proceeds to the next node that includes the </c0> tag. As a result, the less specific relevant chunk (as will be described below in connection with Figure 8C) is the next relevant chunk to be output.

**[00165]** In some embodiments, the query engine 30 outputs this relevant chunk to the front end 15. As a result, the front end 15 may ultimately display two relevant chunks to the end user. Alternatively, the front end 15 may compare the two relevant chunks before displaying them and choose only one of them, e.g., the more specific one above or the second broader one, to be displayed. In some other embodiments, the query engine 30 may choose not to output the second relevant chunk to the front end 15 if it determines that the first one is sufficient to satisfy the end user's search interest.

**[00166]** Figure 8C is a flowchart illustrating a second embodiment of how the query engine 30 identifies a relevant chunk within a node stream representing a candidate document. This embodiment is similar to the embodiment described above in connection with Figure 8C except that, after a relevant chunk is identified, the query engine 30 immediately starts outputting nodes in the identified chunk (895) without waiting for the completion of the relevant chunk (835 in Figure 8B). Moreover, the query engine 30 also outputs subsequent nodes within the same relevant chunk (877), if there are any, without waiting for the completion of the relevant chunk (835 in Figure 8B).

**[00167]** Using the same exemplary candidate document above, the query engine 30 outputs the relevant query when it encounters the second text node including the text string "For XML-based data management" because both search keywords have matches in the relevant chunk. Although this relevant chunk might not be as satisfactory as the more specific one, the response latency of this second embodiment is usually shorter than the response latency of the first embodiment.

**[00168]** As described above in connection with Figure 5, the stream engine 60 receives one or more candidate document identifiers such as URIs from the cache engine 40. For each URI, the stream engine 60 submits a request to a respective data source to retrieve the corresponding candidate document hosted by the data source. If multiple requests are submitted to different data sources within a short time period or even in parallel, the requested candidate documents may arrive at the stream engine 60 simultaneously or nearly so.

**[00169]** In some embodiments, a candidate document such as a web page at a remote web server is divided into multiple data packets at the respective data source and transmitted back to the stream engine 60 one packet at a time. But due to network traffic jams, the data packets from a single data source may arrive at the stream engine 60 out of their original transmission order and the data packets from different data sources may arrive at the stream engine 60 at different rates. The query engine 30, however, usually requires that the data packets of a particular candidate document be analyzed sequentially to identify relevant chunks therein and present them to the end user. This is especially true if a text node that satisfies a search query is larger than the maximum size of a packet and therefore has to be allocated into multiple data packets for network transmission.

**[00170]** As a result, such a deadlock situation often occurs: on the one hand, the stream engine 60 is waiting for a data packet from a first data source to support the query engine 30's operation; on the other hand, the data packet cannot arrive at the stream engine 60 on time due to network delay. At the same time, multiple data packets from a second data source may have arrived at the stream engine 60, but they are postponed from further processing although they might contain a relevant chunk. If this issue is not appropriately resolved, it would significantly increase the computer system's response latency, causing a less satisfactory user experience to the end user.

**[00171]** Figure 9A is a flowchart illustrative of how the stream engine 60 processes multiple candidate documents to identify candidate chunks in accordance with some embodiments. For illustration, assume that the stream engine 60 receives two URIs, UA and UB, from the cache engine 40, each identifying a candidate document at a respective data source. In reality, the stream engine 60 may receive N URIs and therefore process N node streams at the same time, N being an integer number varying from a few to a few hundred.

**[00172]** Initially, whenever it has bandwidth for processing more URIs (902, yes), the stream engine 60 checks whether there are any URI available for processing (904). If not (904, no), the stream engine 60 processes existing node streams (912). In this example, both UA and UB are available (904, yes). The stream engine 60 chooses one of them (906), e.g., UA, and checks the availability of the corresponding data source (908). If the data source is not available (908, no), the stream engine 60 then returns to process the next URI (902). Otherwise (908, yes), the stream engine 60 generates a node stream for UA (910) and then returns to process the next URI (902). At the end, for each candidate document, the stream engine 60 generates a node stream to manage incoming data packets corresponding to the document.

**[00173]** In some embodiments, the stream engine 60 checks the availability of a data source repeatedly until a predefined condition is met, e.g., the time elapsed from the first check to the last check is beyond a threshold level. If no, the stream engine 60 assumes that the corresponding document is not available and devotes its resources to processing other available candidate documents. Note that the stream engine 60 may perform the same or similar exercise repeatedly for each data source from which it has already received data packets. If the stream engine 60 fails to receive any data packet from a data source for a predefined time period, the stream engine 60 may assume that this data source is no longer

available and free any resources allocated for this data source and the corresponding node stream. By doing so, the overall response latency is below a level of reasonable tolerance.

**[00174]** In this example, assume that the stream engine 60 chooses to work on one of the two available node streams (902), e.g., the UA node stream, and the first data packet has arrived (916). The stream engine 60 processes the data packet (920), such as verifying its accuracy, extracting the raw data of the candidate document from the data packet, and converting the raw data into one or more nodes in the UA node stream. Next, the stream engine 60 parses the next node in the UA node stream (922) to identify candidate chunks within the node stream.

**[00175]** For each node in the UA node stream, the stream engine 60 determines if it corresponds to a new candidate chunk (926) or is within an existing candidate chunk (928) until finishing the last one in the node stream (924). In either case (926, yes) (928, yes), the stream engine 60 accumulates the node into the candidate chunk (930) and then determines whether it has reached the end of the corresponding candidate chunk. If so (932, yes), the stream engine 60 sends the candidate chunk to the query engine 30 for further processing (934), e.g., determining whether the candidate chunk is a chunk relevant to the user-specified search keywords.

**[00176]** In some embodiments, after sending the candidate chunk to the query engine 30, the stream engine 60 returns to parse the next one in the node stream (922) and repeats the aforementioned operations until it exhausts the last one in the node stream. In other words, the stream engine 60 and the query engine 30 may proceed in parallel and independently. This mechanism or the like can be very efficient if the computer system 100 has enough resources, e.g., multiple processors (including co-processors) and/or a large amount of memory, or if different components within the computer system 100, e.g., the stream engine 60 and the query engine 30, operate on separate threads and there is a carefully-maintained thread boundary between the two.

**[00177]** In some other embodiments, the stream engine 60 pauses after passing one candidate chunk to the query engine 30 (934) and resumes processing the node stream after it receives feedback from the query engine 30 (936). This mechanism or the like may be more feasible if the computer system 100 has limited resources, e.g., a single processor and/or limited memory. In this case, the stream engine 60 and the query engine 30 share the same

thread. As a result, the computer system 100 may only need a small amount of memory to have a reasonably efficient performance. A more detailed description of this feedback-based scheme is provided below in connection with Figures 10A-10B and 11A-11G.

**[00178]** As noted above, a candidate chunk is semantically and contextually a unit within a candidate document. The process described above in connection with Figure 8A may annotate multiple nodes in a node stream, each annotated node corresponding to a candidate chunk. These candidate documents may be associated with different levels of a hierarchical data model of the candidate document. In other words, a small candidate chunk can be a descendant of a large candidate chunk.

**[00179]** Figure 9B is an exemplary HTML document to be processed by the stream engine as shown in Figure 9A in accordance with some embodiments. From applying the corresponding heuristic rules to this HTML document, the stream engine 60 identifies nine candidate chunks 942 through 958. Note that the first node within each candidate chunk is highlighted in Figure 9B. For example, the first node of the candidate chunk 942 is the <table> tag 960 and the first node of the candidate chunk 956 is the  tag 974. The candidate chunk 956 and the candidate chunk 958 are at the same level in the hierarchical data model, both of which are descendants of the larger candidate chunks such as the candidate chunk 954.

**[00180]** When applying the process in Figure 9A to the HTML candidate document in Figure 9B, the stream engine 60 receives the node including the <table> tag 960 and a new candidate chunk 942 is found (924, yes). Subsequently, the stream engine 60 receives the node including the <td> tag 962 and another new candidate chunk 944 is found (924, yes). When the stream engine 60 receives the  tag 976, the first complete candidate chunk 956 is found (930, yes) and the stream engine 60 sends the candidate chunk 956 to the query engine 30 (932). Similarly, when the stream engine 60 reaches the  tag 980, the second complete candidate chunk 958 is found (930, yes) and sent to the query engine 30 (932). When the stream engine 60 reaches the </td> tag 982, the third complete candidate chunk 954 is found (930, yes) and sent to the query engine 30 (932). Note that the candidate chunk 954 is the parent of the two candidate chunks 956 and 958 and the candidate chunk 954 does not have any content outside the two descendant candidate chunks 956 and 958. As will be explained below, the query engine 30 identifies the candidate chunk 954 as the relevant

chunk if the two descendant candidate chunks 956 and 958 in combination satisfy the user-specified search keywords.

**[00181]** Assume that the stream engine 60 has processed the last node in the UA node stream, which is one of multiple data packets occupied by a large paragraph in the corresponding candidate document, and the stream engine 60 has not received the last of the multiple data packets yet. In this case, because there are no more nodes in the UA node stream (922, no), the stream engine 60 returns to process the next URI (902). But as noted above, there are no more URIs available (904, no) because the stream engine 60 receives only two URIs from the cache engine 40 and it has already generated a node stream for each URI. The stream engine 60 then has to choose between the UA node stream and the UB node stream (912).

**[00182]** If the stream engine 60 chooses one of the two node streams, e.g., the UA node stream, and for some reason the next data packet associated with the UA node stream has not arrived at the stream engine 60 after a certain time (918, no), the stream engine 60 then returns to perform operation 902. In some embodiments, the stream engine 60 does not randomly choose the next available node stream. Rather, it compares the available node streams and selects one node stream that has one or more data packets waiting to be processed (912). By doing so, the stream engine 60 effectively reduces the risk of running into the deadlock situation described above, which blocks the query engine 30 from identifying and serving relevant chunks from a different node stream.

**[00183]** For example, after finishing the last node in the UA node stream, the stream engine 60 may choose the UB node stream (912) and start searching for candidate chunks within the UB node stream until (i) the UB node stream is completed (914, no) or (ii) there is a network traffic jam with the UB node stream (924, no). In either case, the stream engine 60 repeats the same process described above to work on the UA node stream if it has newly received data packets and there is still time for processing the node stream for a given response latency threshold.

**[00184]** In some embodiments, as noted above, a feedback mechanism (936, Figure 9A) is established between the stream engine 60 and the query engine 30. The description above illustrates the activities on the stream engine side. The description below in connection with Figures 10 and 11 focuses on the query engine side, in particular, how the

query engine 30 works in concert with the stream engine 60 to identify relevant chunks in response to a search request.

**[00185]** Figure 10A is a block diagram illustrative of how a query mediator coordinates the query engine 30 and the stream engine 60 to identify chunks within a node stream representing a candidate document in accordance with some embodiments.

**[00186]** As described above, upon receiving a search query, the query engine 30 may generate one or more path filters (308, Figure 3), the path filters are passed down to the stream engine 60 by the cache engine 40 (411, Figure 4), and the stream engine 60 then applies the path filters to a node stream (512, Figure 5). Figure 10A depicts these processing steps in a slightly different manner.

**[00187]** Upon receiving the search query, the query engine 30 performs query processing 1010 to define a set of input sequences 1015 for the search query. The set of input sequences 1015 further defines one or more path filters, which are used to build a filter model 1020. In some embodiments, as described below in connection with Figure 10B, the filter model 1020 is the same or similar to a deterministic finite state machine (FSM).

**[00188]** In addition to defining the path filters, the query engine 30 iterates the input sequences 1015 and their associated node sub-streams to identify relevant chunks. Initially, because the query engine 30 has not received anything from the stream engine 60, a data request is submitted to the query mediator 1025. The query mediator 1025 is a user-configurable tool through which the user can, e.g., specify the maximum number of nodes in memory at any given time and control the rate of node stream consumption by the query engine 30.

**[00189]** In some embodiments, as the query engine 30 iterates each input sequence 1015 and its associated node sub-stream, it determines whether a desired node is currently in memory. If not, the query engine 30 asks the query mediator 1025 for the desired node until one of the predefined conditions is met. These conditions include: (i) another context node for the input sequence is available; (ii) another fragment or content node of the current context node is available; and (iii) the current context node is complete. A more detailed description of context nodes is provided below in connection with Figures 11A through 11G.

**[00190]** In response to the data request, the query mediator 1025 triggers the stream engine 60 for further conversion of raw data into the node stream 1030. As a result, more nodes are submitted to the filter model 1020. The filter model 1020 feeds these nodes into the finite state machine it builds using the path filters to accumulate those nodes matching the path filters in their respective sub-streams until one of the predefined conditions is satisfied. By then, the query mediator 1025 passes the control back to the input sequences 1015 and therefore the query engine 30, which analyzes the node sub-streams to identify relevant chunks.

**[00191]** In sum, this feedback mechanism between the stream engine 60 and the query engine 30 ensures that a minimum number of nodes are stored in the computer system 100's memory and processed by the query engine 30 to fulfill the search query, and that this process is accomplished without loss of any raw data.

**[00192]** Figure 10B is a flowchart illustrative of how the stream engine 60 divides the node stream into multiple sub-streams using a filter model in accordance with some embodiments.

**[00193]** Using the path filters provided by the query engine 30, the stream engine 60 generates a finite state machine (1034). The input to the finite state machine is a node stream corresponding to the raw data of a candidate document and the output is one or more node sub-streams, each node sub-stream including a set of nodes that may be potentially relevant to the search query. Thus, the finite state machine effectively filters out nodes that are deemed to be completely irrelevant to the search query and reduces the amount of data to be handled by the query engine 30. Next, the stream engine 60 receives the next node in the node stream (1036) and compares the node with the finite state machine (1038) to determine if the node belongs to one or more node sub-streams associated with the path filters.

**[00194]** In some embodiments, the finite state machine performs different operations in accordance with different comparison results. For example, the finite state machine may: (i) perform a transition operation (1040-A) and move itself from the current state to a different one that is associated with the node (1042); (ii) perform a maintenance operation (1040-B) and stay at the current state (1044); or (iii) perform a null operation (1040-C) and discard the node as irrelevant to the search query (1046). In the last case, the finite state machine may also stay at the current state.

**[00195]** After performing a transition/maintenance operation, the stream engine 60 accumulates the node into a respective node sub-stream (1048). Depending on whether the node is a context node (1050-A) or a content node (1050-B), the stream engine 60 may insert the node into the context node sub-stream (1052) or insert the node into a node sub-stream that is associated with the current context node (1054). A more detailed description of this accumulation operation is provided below in connection with Figure 11E. Next, the stream engine 60 determines whether the node stream is completed (1056). If so (1056, yes), the stream engine 60 sends the node sub-streams to the query engine 30 for further process (1058). Otherwise (1056, no), the stream engine 60 returns to process the next node in the node stream (1036).

**[00196]** To further explain the feedback mechanism between the stream engine 60 and the query engine 30, Figures 11A through 11G illustrate in detail how a candidate document is processed using the feedback mechanism.

**[00197]** Figure 11A is an exemplary XML document 1100 to be processed by the stream engine 60 and the query engine 30 in accordance with some embodiments. The XML document 1100 includes a list of books 1102 through 1108, each book being identified by its publication year (the "year" attribute in the <book> tag), its title (the pair of <title> and </title> tags), its author (the pair of <author> and </author> tags) including first name (the pair of <first> and </first> tags) and last name (the pair of <last> and </last> tags), its publisher (the pair of <publisher> and </publisher> tags), and price (the pair of <price> and </price> tags).

**[00198]** Figure 11B is an exemplary XQuery 1110 to be applied to the XML document 1100 in accordance with some embodiments. The XQuery 1110 searches for any book in the XML document "bib.xml" whose publisher is Addison-Wesley and whose publication year is later than 1991. The XQuery 1110 requires that the search results be returned in the form of a new XML document including a new list of the books matching the two search criteria, each book in the new XML document only including the book's title and its publication year as an attribute in the <book> tag.

**[00199]** Figure 11C is a table 1115 of the five input sequences defined by the query engine 30 in accordance with some embodiments. Based on the XQuery 1110, the query engine 30 defines five input sequences, each input sequence corresponding to one tag or tag

attribute within the XML document 1100. Note that the publication year attribute "@year" appears twice in the XQuery 1110, one in the where clause and the other in the return clause, and corresponds to two separate input sequences. The five input sequences each have an associated node sub-stream labeled "Node Sub-Stream (0)" through "Node Sub-Stream (4)" and correspond to a respective path filter as shown in the table 1115.

**[00200]** Different input sequences are associated with different portions of the XQuery 1110 and therefore have different modes. For example, the <book> tag associated with the input sequence "Node Sub-Stream (0)" appears in the for-in clause, but not the return clause. Thus, the input sequence "Node Sub-Stream (0)" is presumed to provide context for the search process and serve in the "Context" mode, and the nodes in the corresponding node sub-stream are referred to as "context node sub-stream."

**[00201]** Similarly, the content of the <publisher> tag associated with the input sequence "Node Sub-Stream (1)" is compared with "Addison-Wesley" in the where clause of the XQuery 1110. Thus, the input sequence "Node Sub-Stream (1)" is presumed to provide content for the search process and serve in the "Content" mode, and the nodes in the corresponding node sub-stream are therefore referred to as "content node sub-stream." The <title> tag associated with the input sequence "Node Sub-Stream (4)" appears in the return clause. Thus, the input sequence "Node Sub-Stream (4)" is presumed to provide both context and content for the search process and serve in the "All" mode. In some embodiments, an input sequence in the "All" mode has two node sub-streams.

**[00202]** The "Parent" column in the table 1115 indicates whether an input sequence is a child of another input sequence. In this example, the input sequence associated with the for-in clause provides basis for the other input sequences associated with the other parts of the XQuery 1110. Any node in one of the other four input sequences corresponds to a specific node in the input sequence "Node Sub-Stream (0)," which is therefore deemed to be the parent input sequence of the other four input sequences.

**[00203]** Figure 11D is a flowchart illustrative of how the query engine 30 processes node sub-streams at different input sequences in accordance with some embodiments. This flowchart provides more details of the information flow shown in the block diagram of Figure 10A.

**[00204]** The query engine 30 initializes the stream engine 60 (1120) and processes the search query (1122) to define input sequences, path filters, and a finite state machine that is used for generating one or more node sub-streams. The query engine 30 then starts iterating the next node sub-stream (1124). In this example, the query engine 30 begins with the context node sub-stream of Node Sub-Stream (0).

**[00205]** If the context node sub-stream has no context node (1126, no), the query engine 30 then requests more context nodes from the stream engine 60 (1128, 1130). Consequently, more data packets are retrieved (1132) and parsed (1134) to provide more nodes, including context nodes and content nodes, to the query engine 30.

**[00206]** Once a new context node is present in the node sub-stream of Node Sub-Stream (0) (1126, yes), the query engine 30 applies the search query to the context node and its associated nodes in other node sub-streams (1136). If the search criteria are satisfied (1138, yes), a relevant chunk has been identified and there is no need to apply the search query to the remaining portion of the relevant chunk. Thus, the query engine 30 needs to quickly reach the end of the relevant chunk through round-tripping the content nodes in different node streams (1140). After finishing the content nodes, if the end of the chunk has not been reached (1142, no), the query engine 30 may request the stream engine 60 to process more data packets (1146).

**[00207]** If the search criteria are not met (1138, no), a relevant chunk has not been identified, and the query engine 30 sends a request to the query mediator to retrieve one or more content nodes and re-apply the search query. If the stream engine 60 has more nodes or node fragments (1144, yes), they will be parsed and submitted to the query engine 30 (1134). Otherwise (1144, no), the query engine 30 may request the stream engine 60 to process more data packets (1146).

**[00208]** As shown in Figure 11C, the XQuery 1110 defines five input sequences and therefore five path filters. The stream engine 60 uses these path filters to build a finite state machine, which, as shown in Figure 10B, is to divide the original node stream corresponding to the XML document 1100 into five node sub-streams. The finite state machine has an initial state, which can be any one of the five input sequences.

**[00209]** Figure 11E is a block diagram illustrative of how the node stream is divided into multiple node sub-streams by the finite state machine in accordance with some

embodiments. From the start state (1150), the finite state machine receives a node including the <bib> tag. Because this tag is not relevant to any input sequence, the finite state machine discards the node. After receiving a node including the <book> tag, the finite state machine makes a transition to the state corresponding to Node Sub-Stream (0) and adds the node into the corresponding context node stream (1152). Next, the node including the publication year attribute is processed and added into the two respective node sub-streams corresponding to Node Sub-Stream (2) and Node Sub-Stream (3) (1154).

**[00210]** Upon receiving a node including the <title> tag, the finite state machine makes another transition to the state corresponding to Node Sub-Stream (4). As noted above in connection with Figure 11C, the input sequence Node Sub-Stream (4) serves in the "All" mode. Thus, besides adding the node including the <title> tag into the corresponding node sub-stream (1156), the finite state machine adds everything enclosed by the pair of (<title>, </title>) tags into the same node sub-stream or a separate node sub-stream (1158). For example, if there is a pair of (<subtitle>, </subtitle>) tags within the pair of (<title>, </title>) tags, they will be added into the respective node sub-stream because, as explained above, the XQuery 1110 requires the return of each matching book's title, including its subtitle, if present.

**[00211]** Similarly, the node including the <publisher> tag is added into the node sub-stream corresponding to Node Sub-Stream (1) (1160) and the textual portion within the pair of (<publisher >, </publisher>) tags is extracted by a text() function and added into the same or a separate node sub-stream (1162). This textual portion is required by the XQuery 1110 to check whether the book is published by the publisher Addison-Wesley.

**[00212]** Figure 11F is a block diagram illustrative of the input sequences and their associated node sub-streams after the first candidate chunk in the XML document is processed in accordance with some embodiments.

**[00213]** The node sub-stream "Node Sub-Stream (0)" is first populated with a context node "<book>" (1164). Next, the node sub-streams "Node Sub-Stream (2)" and "Node Sub-Stream (3)" are each populated with a content node "1994" (1166, 1168). For the node including the <title> tag, the stream engine 60 inserts into the node sub-stream "Node Sub-Stream (4)" both the <title> tag (1170) and the data descending from the <title> tag (1172). For the node including the <publisher> tag, the stream engine 60 is only interested in its

content and therefore populates the node sub-stream "Node Sub-Stream (1)" with the textual portion of the <publisher> tag (1174).

**[00214]** Figure 11G is the search result of applying the XQuery 1110 to the node sub-streams derived from XML document 1100 in accordance with some embodiments. The search result is also an XML document 1180 that includes two books 1182 and 1184 that satisfy the XQuery 1110. As shown in Figure 11F, the node sub-streams corresponding to the five input sequences include all information necessary for generating this resulting XML document 1180.

**[00215]** Thus far, detailed descriptions of document-processing schemes in response to a search request, including the downstream processes 25 and the upstream processes 35, are provided above. These document-processing schemes can be used to implement various applications to satisfy different user needs. For illustration, embodiments of representative applications are provided below.

**[00216]** One application of the invention is to improve a user's experience with the search results generated by search engines. Although a document identified by the search results is relevant to the search keywords, the document may not include any relevant chunks because the search engines treat the entire document, not a chunk within the document, as the basic unit to be compared with the search keywords. Thus, one aspect of the invention is to identify and display relevant chunks within a document in response to a search request.

**[00217]** Figure 12A is a flowchart illustrative of a first process of identifying one or more documents, each document having one or more chunks that satisfy user-specified search keywords, in accordance with some embodiments.

**[00218]** A computer identifies multiple resource identifiers (1201), each resource identifier corresponding to a document at a respective data source. In some embodiments, a resource identifier is a URL, which identifies a web page at a remote web server. In some embodiments, the resource identifiers are part of search results produced by a server computer such as a search engine in response to one or more search keywords provided by an end user from a client computer.

**[00219]** For at least one of the resource identifiers, the computer retrieves the corresponding document from the respective document source (1203). If the document is a

web page hosted by a web server, the computer submits an HTTP request to the web server and the web server returns the document in an HTTP response. Within the retrieved document, the computer identifies a chunk that satisfies the user-specified search keywords (1205) and displays the identified chunk and a link to the identified chunk within the document to the user (1207).

**[00220]** Figure 12B is a flowchart illustrative of a second process of identifying one or more document, each document having one or more chunks that satisfy user-specified search keywords, in accordance with some embodiments.

**[00221]** A client computer submits one or more user-specified search keywords to a server computer (1211). In some embodiments, the server computer is one or more third-party search engines. The client computer receives a set of search results from the server computer (1213), each search result identifying a document located at a respective document source that satisfies the search keywords in accordance with a first set of predefined criteria.

**[00222]** For each identified document, the client computer retrieves the document from the corresponding document source (1215), identifies a chunk within the document that satisfies the search query in accordance with a second set of predefined criteria (1217), and displays the identified chunk and a link to the identified chunk within the document (1219). In some embodiments, the two sets of predefined criteria are different. For example, the first set of criteria requires that all the search keywords be found within a document, but not necessarily within a chunk. In contrast, the second set of criteria is satisfied only if all the search keywords are found within a chunk.

**[00223]** Figures 12C through 12J are screenshots of a graphical user interface on a computer display illustrative of features associated with the processes as shown in Figures 12A and 12B in accordance with some embodiments.

**[00224]** The graphical user interface includes one or more document links, each document link having one or more associated chunks identified within the corresponding document as satisfying one or more user-specified search keywords. In some embodiments, each chunk has an associated chunk link and includes terms matching each of the user-specified search keywords. The matching terms may also be highlighted in the chunk in a visually distinguishable manner (such as in different colors, font types or combination thereof). In response to a user selection of a chunk's chunk link, the corresponding document

is displayed in a window on the computer display and at least a portion of the chunk is highlighted in the window.

**[00225]** In some embodiments, each document link has an associated chunk page-link icon for searching chunks within documents that are referenced by the corresponding document. In response to a user selection of a document link's associated chunk page-link icon, one or more referenced document links are displayed on the computer display, with each referenced document link having one or more associated chunks identified within a corresponding referenced document as satisfying the user-specified search keywords.

**[00226]** In some embodiments, each document link has an associated hide-chunk icon. In response to a user selection of a document link's associated hide-chunk icon, the chunks associated with the document link and their associated chunk links disappear from the computer display.

**[00227]** In some embodiments, chunks associated with a respective document link are displayed in an order consistent with their relative relevancy to the user-specified search keywords. In some other embodiments, chunks associated with a respective document link are displayed in an order consistent with their relative locations within the corresponding document.

**[00228]** As shown in Figure 12C, through an application 1220 (e.g., a web browser window), a user submits three search keywords 1221 from a client computer to a content provider such as a search engine. In this example, the application 1220 provides four different search options for the user to choose. They are:

- "Best Match" option 1226-A – This search option allows the application 1220 to adaptively select one or more chunks satisfying one or more of the user-specified search keywords according to predefined criteria. In some embodiments, the "Best Match" option is the default option if the user does not expressly choose a different one. A more detailed description of this search option is provided below in connection with Figures 21A through 21D.

- "Match All" option 1226-B – This search option limits the search results to relevant chunks that satisfy each of three user-specified search keywords. Thus, a candidate chunk that only includes "einstein" and "bohr," but not

"debate," should not be in the search results responsive to the "Match All" option, but may be in the search results responsive to the "Best Match" option. As shown in Figure 12C, the user expressly chooses the "Match All" option.

- '"Exact' Match" option 1226-C – This search option further limits the search results to relevant chunks that not only satisfy each of three user-specified search keywords, but also include an exact match of the search keywords as a string. Examples of "exact"-matching chunks are shown in Figures 12F and 12G, respectively. Note that this option is different from the string-match approach, which is variant-sensitive. Under the string-match approach, "einstein bohr debates" does not match "Einstein-Bohr debate." But according to the '"Exact' Match" option, the two sets of terms do match each other as this search option ignores any non-word characters such as white space, punctuation, etc., and only requires that the three terms appear in the same order and have no intervening terms.

- "Match Any" option 1226-D – This search option allows the application 1220 to identify and display any chunk that satisfies at least one of the user-specified search keywords. Thus, the search results responsive to any of the three options above are a subset of the search results responsive to the "Match Any" option, an example of which is depicted in Figures 12I and 12J.

**[00229]** The content provider returns a search result 1225 to the client computer and the search result 1225 includes an abbreviated document segment identified by the search engine as satisfying the search keywords. The client computer retrieves a document identified in the search result 1225 (an HTML web page in this example) from a web server and identifies one or more chunks 1229-A through 1229-C therein that satisfy the search keywords 1221, each chunk having an associated link 1231 to the chunk in the original web page. In some embodiments, each of the chunks 1229-A through 1229-C are different from the abbreviated document segment because it is a semantically and contextually consistent unit within the document without abbreviation.

**[00230]** In some embodiments, after retrieving a candidate document, the application 1220 generates a search query using the search keywords and applies the search query to the retrieved document to identify relevant chunks within the document.

**[00231]** In some embodiments, the terms that match the search keywords in the identified chunk are ordered differently from the user-specified search keywords. For example, the term "debate" appears between "Bohr" and "Einstein" in the chunk 1229-B of Figure 12C.

**[00232]** In some embodiments, the terms that match the search keywords in the identified chunk are separated from one another by at least one term not matching any of the search keywords. For example, the three terms appearing in the last sentence of the chunk 1229-A are separated from one another by many other words. Unlike the conventional string search, an identified chunk may or may not include an exact match of the search keywords as a string. Rather, the search process according to some embodiments of the invention includes tokenization of the search keywords in a text string into atoms and subsequent search in the token space according to the atoms, which is variant-agnostic by, e.g., ignoring things like grammatical tense, punctuation, white space, casing, diacritics, etc. in the search keywords. For example, in the screenshot of Figure 12C, "Einstein Bohr debate" and "einstein bohr debating" are deemed to be identical according to some embodiments of the invention.

**[00233]** In some embodiments, an identified chunk includes an identical instance of the search keywords appearing as a phrase. But, as noted above, although the instance is the same as the result of a string search, the search keywords are not identified collectively as a text string within the chunk.

**[00234]** In some embodiments, different terms matching different search keywords in the identified chunk are highlighted in different manners such as different colors, different foreground/background patterns, different font types/sizes, or a combination thereof. In Figure 12C, the three terms appearing in each chunk are highlighted using large, italic, and underlined font. In some embodiments, the three terms are further distinguished from one another using a unique style for each term. For example, the three terms may have three completely different styles such as Courier New for "**Einstein**," Arial for "**Bohr**," and Monotype Corsiva for "***debate***." In some other embodiments, the three terms may have different background colors, such as gray for "Einstein," green for "Bohr," and yellow for "debate." In yet some other embodiments, the different manners may be combined to further distinguish different search keywords appearing in the same chunk.

**[00235]** In some embodiments, one or more sponsored links (1227, Figure 12C) are identified to be associated with at least one of the search keywords and displayed adjacent the identified chunk.

**[00236]** As shown in Figure 12C, there are a chunk page-link icon 1223 and a hide-chunk icon 1224 below the search result 1225. In response to a user selection of the chunk page-link icon 1223, the computer retrieves documents that are referenced by the document identified by the search result 1225 and therefore have page links in the document. For each retrieved document, the computer identifies chunks within the document that satisfy the search keywords 1221 by apply the same "chunking" process that has been applied to the document identified by the search result 1225.

**[00237]** Figure 12D is a screenshot illustrative of the search results after a user selection of the chunk page-link icon 1251, including a link 1253-A, 1253-B to a respective document and a set of relevant chunks 1255-A, 1255-B identified within the corresponding document. The terms that match the search keywords are similarly highlighted in the relevant chunks. Note that a user can repeat this process by clicking the chunk page-link icons 1254-A, 1254-B associated with the respective documents. In some embodiments, the application 1220 applies its default search option, e.g., "Best Match" option 1226-A, for performing the task associated with the chunk page-link icon 1251. In some other embodiments, the user can override the default search option by expressly selecting another option.

**[00238]** Figure 12E is a screenshot illustrative of the search results after the user clicks the hide-chunk icons (1224, Figure 12C) associated with the respective search results. In this example, the relevant chunks associated with a search result disappear from the web browser window and the hide-chunk icons become show-chunk icons 1257A, 1257B. The relevant chunks are displayed again in the web browser window after a user selection of the show-chunk icons.

**[00239]** In some embodiments, multiple relevant chunks are identified within a candidate document and these chunks are displayed in an order consistent with their relative locations within the document. Figure 12F is a screenshot that includes multiple relevant chunks, each one satisfying the two search keywords "Bohr-Einstein" and "debates." These chunks are listed in the web browser window in accordance with their relative locations in the

web page such that the first chunk 1233-A that appears first in the web page is displayed above the other ones and the last chunk 1233-B that appears below the other chunks is displayed at the bottom of the web browser window.

**[00240]** In some embodiments, multiple relevant chunks are identified within a candidate document and these chunks are displayed in an order consistent with their relative relevancy to the search keywords. Figure 12G is another screenshot that includes the same set of relevant chunks. Assume that the chunk 1233-B is more relevant than the chunk 1233-A. The more relevant chunk 1233-B is displayed above the other less relevant chunks including the chunk 1233-A. For illustrative purposes, the two screenshots in Figures 12F and 12G are generated using the "'Exact' Match" option 1226-C. Each chunk 1233-A, 1233-B includes at least one instance of the two search keywords as a string (ignoring the casing difference). The aforementioned chunk-ordering schemes or the like are equally applicable to the other search options.

**[00241]** In some embodiments, in response to a user selection of the link to an identified chunk, at least a portion of the identified document is displayed in a document view window and the displayed portion includes, at least partially, the identified chunk. Figure 12H is a screenshot of the web browser window after a user click of the chunk link 1235. A document view window 1237 is displayed next to the search results. The document view window 1237 displays a portion of the document that includes the relevant chunk and the displayed portion includes at least part of the relevant chunk 1239 within the document. In this example, the relevant chunk 1239 is highlighted in the document view window. Sometimes, the terms matching the search keywords in the relevant chunk 1239 are processed such that they are visually distinguishable over the rest of the identified chunk, e.g., using different colors or font types.

**[00242]** In some embodiments, for each relevant chunk in the identified document, the computer inserts a pair of unique chunk virtual delimiters into the identified document. This pair of chunk virtual delimiters uniquely defines the scope of the relevant chunk within the identified document, but is invisible to the user when the identified document is being rendered by an application. In response to a user request to view the relevant chunk 1239 in the document view window 1237, the computer can quickly locate the scope of the relevant chunk 1239 within the document by looking for the corresponding pair of chunk virtual delimiters and then highlight the chunk in the document view window appropriately.

**[00243]** In some embodiments, the HTML tag  can be introduced into a candidate document for forming chunk virtual delimiters. For example, the following chunk in an HTML document

```
<p>This is a candidate chunk.</p>
```

can be re-defined as:

```
<span id="chunk-1"><p>This is a candidate chunk.</p></span>
```

**[00244]** The HTML tag  has no effect on the appearance of the chunk in a web browser window because it has no associated style information. But the pair of chunk virtual delimiters (, ) uniquely identifies the chunk's location in the document, which a web browser application can rely upon to highlight the chunk's existence by, e.g., altering its background color. Note that the HTML tag  is not the only choice of a suitable invisible anchor element. In some other embodiments, it is possible to use one or more document-unique, chunk-unique identifiers or the like within the document as chunk virtual delimiters to achieve the same or similar effect.

**[00245]** In some embodiments, for at least one of the resource identifiers, after the corresponding document is retrieved from the respective document source, no relevant chunk that satisfies each of the search keywords is identified therein. This scenario happens if the terms matching the search keywords are distributed in different chunks within the document. In this case, the web browser window displays a link to search for chunks that satisfy any of the search keywords within the document. In response to a user selection of the link to search for chunks that satisfy any of the search keywords within the document, the retrieved document is re-processed, and as a result, one or more chunks that satisfy at least one of the search keywords is identified in the document. Accordingly, these chunks are displayed to the end user.

**[00246]** Figure 12I is a screenshot that includes a search result 1241 that satisfies all the search keywords "Einstein" and "big bang." Because no relevant chunk is found in the web page, the web browser window provides a link 1243 to "re-chunk" the web page to search for any chunk matching any search keywords. Figure 12J is another screenshot after the user click of the link 1243. Note that at least five chunks are identified in the document, three chunks 1245 including the keyword "Einstein" and two other chunks 1247 including the

keywords "big bang." But no chunk satisfies all the search keywords. In some embodiments, the same set of chunks can be identified in the document through a user selection of the "Match Any" option 1226-D.

**[00247]** Another application of the invention is to identify and display within a document relevant chunks satisfying user-specified search keywords while the user is browsing the document. Conventionally, a user visiting a web page may be only interested in the content of a particular paragraph therein. To find the paragraph, the user-specified text string has to exactly match the one in the paragraph. Otherwise, the paragraph can not be easily located in the document if the user can provide a few search keywords but is mistaken about their exact sequence in the paragraph. Such issues with the conventional approach have been solved by the application described below.

**[00248]** Figure 13A is a flowchart illustrative of a first process of identifying within a document one or more chunks that satisfy user-specified search keywords in accordance with some embodiments.

**[00249]** A computer displays a portion of a document to a user (1302). Upon receiving a user-specified text string that includes multiple search keywords, the computer identifies a chunk within the document that satisfies the search keywords (1304) and displays the identified chunk to the user (1306). In some embodiments, the identified chunk is not within the initially displayed portion of the document. To locate the chunk, the computer generates a search query using the search keywords and applies the search query to the document to identify the chunk. In some embodiments, the terms that match the search keywords are either ordered differently from the search keywords in the user-specified text string or separated from one another by at least one term not matching any of the search keywords.

**[00250]** Figure 13B is a flowchart illustrative of a second process of identifying within a document one or more chunks that satisfy user-specified search keywords in accordance with some embodiments.

**[00251]** While a user is browsing a document through a computer, the computer receives multiple user-specified search keywords (1312). The search keywords have a first sequence. Within the document, the computer identifies at least one chunk that satisfies the search keywords (1314) and displays a portion of the document including the identified

chunk (1316). In some embodiments, the search keywords are highlighted in the identified chunk and have a second sequence that is different from the first sequence.

**[00252]** Figures 13C through 13G are screenshots of a graphical user interface on a computer display illustrative of features associated with the second process as shown in Figures 13A and 13B in accordance with some embodiments.

**[00253]** Figure 13C is a screenshot of a web page covering Bohr-Einstein debates at www.wikipedia.org. Assuming that a visitor of this web page is interested in learning about the experimental apparatus developed by George Gamow, the visitor can enter a few search keywords relating to this topic in the input field 1322 and then click the "Chunk Page" icon 1323.

**[00254]** Figure 13D is a screenshot of the web page including the identified chunk 1326 that satisfies the user-specified search keywords 1324, i.e., "gamow" and "experiment." In this example, the relevant chunk 1326 is actually not a paragraph, but a caption of a figure in the document. The sentence 1327 including the two keywords is read as follows: "George *Gamow*'s make-believe experimental apparatus for validating the thought *experiment* ..." Although the two keywords are separated from each other by other terms, the figure caption is identified nonetheless because the two keywords happen to be within the same chunk.

**[00255]** Figure 13E is a screenshot illustrative of another embodiment of the invention in response to a user selection of the "Chunk Page" icon at the top of the web browser window. In this example, the left side of the web browser window displays the relevant chunks 1325 identified within the web page. If the web page has multiple relevant chunks, the user can easily get an overview of these chunks from the left side of the web browser. The right side of the web browser is a document view window that displays the portion of the document including the relevant chunk 1326. Thus, this document view window provides more contexts for each relevant chunk to the user.

**[00256]** In some embodiments, like the examples described above in connection with Figures 12C through 12J, different terms in the identified chunk that match different search keywords are highlighted in different manners such as different colors, different foreground/background styles, different font types, or a combination thereof.

**[00257]** In some embodiments, multiple relevant chunks are identified within a document, each one appearing at a respective location in the document. In this case, the web browser window displays, at least partially, the chunk that appears above the other chunks in the document and its associated context.

**[00258]** Figure 13F is a screenshot of another web page at www.wikipedia.org. In response to the user-specified search keywords 1328 "cosmic," "background," and "radiation," the first relevant chunk 1330 in the web page that matches the three search keywords is identified and displayed in a visually distinguishing manner. A scroll down of the web page displays additional relevant chunks identified in the web page.

**[00259]** Sometimes, the first relevant chunk shown in Figure 13F is not necessarily the most relevant one. In some embodiments, after identifying multiple chunks within the document, the web browser assigns to each chunk a ranking metric indicative of its relevancy to the search keywords and displays in a prominent location, at least partially, the chunk that has the highest ranking metric.

**[00260]** Figure 13G is a screenshot of the same web page shown in Figure 13F. But the relevant chunks are now displayed in an order consistent with their relevancy to the search keywords. In this case, the relevant chunk 1332 is a section heading, which is presumably more relevant than the chunk 1330 shown in Figure 13F.

**[00261]** In some embodiments, if there is no chunk within the document that satisfies each of the search keywords, the web browser, or more specifically, the "Chunk Page" toolbar application, may relax its search criteria to look for any chunks in the document that satisfy any of the search keywords and display them to the user. In other words, this feature is similar to the one described above in connection with Figures 12I and 12J.

**[00262]** Another application of the invention is to identify relevant chunks within unstructured or semi-structured documents. It has been a particular challenge to identify chunks within an HTML web page because the HTML syntax allows its user to produce the same or similar web page layout using very different metadata.

**[00263]** Figure 14 is a flowchart illustrative of a process of modeling a document and identifying within the document one or more chunks that satisfy user-specified search keywords in accordance with some embodiments.

**[00264]** A computer identifies a document in response to a search request from a user (1401). The document includes content data and metadata, and the search request includes one or more search keywords. In some embodiments, the document is a semi-structured document, e.g., an HTML web page. The content data refers to the document's content such as a paragraph, a table, or a list of bullet items, etc. The metadata specifies how the content data should be rendered through an application, e.g., a web browser window.

**[00265]** The computer generates a hierarchical semantic model of the content data of the document by applying heuristics to the metadata of the document (1403). In some embodiments, the generation of the hierarchical semantic model includes identifying one or more candidate chunks in the document, each candidate chunk corresponding to a respective subset of the document. As noted above, the HTML web page shown in Figure 9B has a hierarchical semantic model, which includes a set of HTML tags at different levels.

**[00266]** In some embodiments, a first subset of the document associated with a first candidate chunk encompasses a second subset of the document associated with a second candidate chunk. For example, as shown in Figure 9B, both the candidate chunks 956 and 958 are within the candidate chunk 954, which is, in turn, within the candidate chunk 952. There is no overlapping between the candidate chunk 956 and the candidate chunk 958.

**[00267]** In some embodiments, the heuristics stipulates that a subset of the document is identified as a candidate chunk if the subset of the document has at least one instance of predefined metadata. For example, the candidate chunks 956 and 958 are identified because each begins with the <p> paragraph tag.

**[00268]** In some embodiments, the heuristics stipulates that a subset of the document is deemed to be a candidate chunk if the subset of the document has at least two instances of predefined metadata. For example, two or more instances of the <li> tag appearing in a web page one after another are collectively identified as a candidate chunk.

**[00269]** The computer identifies a chunk within the document by sequentially scanning the hierarchical semantic model (1405). The identified chunk includes a subset of the content data that satisfies the search keywords and the corresponding metadata. The computer returns the identified chunk to the requesting user (1407).

**[00270]** In some embodiments, assume that there are two search keywords, a first search keyword and a second search keyword. While sequentially scanning the semantic model, the computer first identifies some content data that is in the first candidate chunk and precedes the second candidate chunk as satisfying the first search keyword (e.g., "It's **raining** outside …") and then identifies content data in the second candidate chunk that satisfies the second search keyword (e.g., "For XML-based **data** management"). Because both search keywords are matched, the first candidate chunk is chosen to be the identified chunk and returned to the requesting client.

**[00271]** In some embodiments, the computer does not return the first chunk immediately after finding a match for the search keyword. Rather, the computer continues scanning the model until identifying content data in the second candidate chunk that also satisfies the first search keyword (e.g., "Raining Data is your choice"). In this case, the second candidate chunk is returned as the relevant chunk that is more specific than the first one.

**[00272]** In some embodiments, while sequentially scanning the hierarchical semantic model, the computer identifies content data that satisfies the first search keyword in one candidate chunk and content data that satisfies the second search keyword in another candidate chunk. For example, assume that the search keywords are "CAD" and "job listings." As shown in Figure 9B, the candidate chunk 956 includes the search keyword "CAD" and the candidate chunk 958 includes the search keyword "job listings." In this case, the computer chooses the candidate chunk 954, which is the parent of the chunks 956 and 958 in the hierarchical semantic mode, as the identified chunk. Note that there is no other content data or metadata within the candidate chunk 954 besides the two candidate chunks 956 and 958.

**[00273]** Another application of the invention is to transform the user-specified search keywords into a finely-tuned query. Sometimes, the user-specified search keywords may include a special character (e.g., "%") or sequence of characters (e.g., "Jan 22 2008"). This special character or sequence of characters, if interpreted appropriately, can help to find the relevant chunks more efficiently.

**[00274]** Figure 15 is a flowchart illustrative of a process of customizing a search query based on user-specified search keywords in accordance with some embodiments.

**[00275]** After receiving a search keyword provided by a user (1502), the computer selects an archetype for the search keyword (1504). The computer identifies one or more search results in accordance with the archetype (1506) and returns at least one of the search results to the user (1508).

**[00276]** In some embodiments, the archetype has an enumerable set of instances and the search keyword is associated with one of the instances. For example, if the user-specified search keyword is "Tuesday," a possible archetype would be "week," of which "Tuesday" represents one of the seven members in the archetype.

**[00277]** In some embodiments, after selecting the archetype, the computer identifies at least one query operator for the selected archetype, constructs a search query using the query operator, and then executes the search query against one or more data sources. For example, for the "week" archetype, the computer may generate a search query that looks for chunks including not only the keyword "Tuesday," but any of the seven days within a week such as "Sunday," "Monday," etc.

**[00278]** In some embodiments, the query operator has a scope and the search query is constructed to limit search results within the scope. For example, assume that the search phrase is "discount of 10%." It is likely that the user is not only interested in chunks having the phrase "discount of 10%," but also chunks having similar phrases, e.g., "discount of 15%." Alternatively, the user may be mistaken when entering the phrase and the candidate document actually has no chunk including the phrase "discount of 10%," but does have chunks including the phrase "discount of 20%." In this case, the computer may generate a search query for discount within the scope of 0% to 100%. As a result, more relevant chunks are identified.

**[00279]** In some embodiments, the query operator has a pattern and the search query is constructed to limit search results including the pattern. For example, the user-specified phrase "Jan 22 2008" indicates a date pattern. If so, the computer may generate a search query accordingly to search for any chunk having the date pattern.

**[00280]** In some embodiments, after selecting the archetype and before identifying the search results, the computer solicits user instructions in connection with the archetype, constructs the search query in accordance with the user instructions, and executes the search query against the data sources. For example, if the user-specified search keyword includes

the special character "%," the computer may display a user interface through which the user may specify the scope or range associated with that special character, which is then built into the search query.

**[00281]** In some embodiments, based on the user instructions, the computer may generate feedback to the user instructions and then receive more user instructions in connection with the archetype and the feedback. Note that this process may repeat for multiple loops until the user submits a search query execution request, which suggests that the user is satisfied with the customized search query.

**[00282]** Another application of the invention is not only to display relevant chunks identified within a document but also to re-use them for different purposes. For example, when a user composes a Word document using Microsoft Office, the user may like to view a slide in a PowerPoint document and, if needed, generate a copy of the slide in the Word document. Currently, there is no convenient way to do so other than opening the PowerPoint document in a separate window, manually searching for the slide in the window, and manually copying the slide and pasting it into the Word document.

**[00283]** Figure 16A is a flowchart illustrative of a process of displaying and re-using search results based on user instructions in accordance with some embodiments.

**[00284]** A computer displays an application user interface (1601). The application user interface includes a document authoring window and a search results window. In response to a search request including one or more user-specified search keywords, the computer displays in the search results window a set of search results in a text-only display format (1603). In some embodiments, each search result includes a chunk within a respective document that satisfies the search keywords. In response to a user request to view a chunk, the computer launches a document display window in the application user interface and displays therein a portion of the corresponding document that includes the chunk in its native display format (1605). In response to a user request to duplicate a segment of the corresponding document in the document authoring window, the computer generates therein an instance of the segment of the corresponding document in its native display format (1607).

**[00285]** Figures 16B through 16J are screenshots of a graphical user interface on a computer display illustrative of features associated with the process as shown in Figure 16A in accordance with some embodiments.

**[00286]** The application user interface includes a document authoring window and a search results window. A set of search results associated with one or more user-specified search keywords is displayed in the search results window in a text-only display format and each search result includes one or more chunks identified within a respective document as satisfying the user-specified search keywords. In response to a user request to duplicate a chunk within a document in the document authoring window, an instance of the chunk is displayed in the document authoring window in the document's native display format. In some embodiments, two chunks identified within two different documents have different native display formats.

**[00287]** In some embodiments, each chunk in the search results window has an associated chunk link. In response to a user selection of a respective chunk link, a document display window is displayed in the application user interface and a portion of the corresponding document that includes the corresponding chunk is displayed in the document display window in the document's native display format.

**[00288]** In some embodiments, each chunk includes terms that match the user-specified search keywords an associated chunk link. Different terms matching different search keywords are highlighted in the search results window in a visually distinguishable manner.

**[00289]** In some embodiments, the chunks identified within a document are displayed in the search results window in an order consistent with their relative relevancy to the user-specified search keywords. In some other embodiments, the chunks identified within a document are displayed in the search results window in an order consistent with their relative locations within the corresponding document.

**[00290]** Figure 16B is a screenshot of the Microsoft Office 2007 Word application user interface 1611. The main region of the user interface is occupied by a document authoring window 1613. Above the document authoring window 1613 is an add-in 1615 to Microsoft Office 2007. The add-in 1615 includes a keyword(s) input field 1615-A into which the user enters search keywords, a document type selection field 1615-B through which the user selects the types of candidate documents to be searched, and a web source field 1615-C including multiple document sources through which the user can search and re-use documents identified by the respective document sources.

**[00291]** In some embodiments, the set of search results includes a first chunk within a first document having a first content type and a second chunk within a second document having a second content type, wherein the first content type is different from the second content type. Different search keywords in the search results window are highlighted in different manners.

**[00292]** Figure 16C is a screenshot including a search results window 1625 and the search phrases 1621 "Einstein general relativity." In this example, the user limits the document search to two types of documents 1623, Word and PowerPoint. As described above in connection with Figure 1, this search limit is passed down from the front end 15 (the add-in 1615 in this example) to the query engine 30 and then to the cache engine 40. Thus, the cache engine 40 only looks for Word and PowerPoint documents in the index database 50. In this example, one chunk 1627 from a PowerPoint document and another chunk 1629 from a Word document are shown in the search results window 1625.

**[00293]** Note that each chunk in the search results window has an associated content type, which may be different from the document type of the corresponding document that includes the chunk. For example, a Word document may include a PowerPoint slide or an Excel spreadsheet. If the PowerPoint slide is identified to be the relevant chunk, the content type of the relevant chunk is PowerPoint, not Word, although the PowerPoint slide is within a Word document. Similarly, if a row in the Excel spreadsheet is identified to be the relevant chunk and the content type of the relevant chunk is therefore Excel, not Word. These chunks may or may not be displayed depending upon the embodiment.

**[00294]** In some embodiments, in response to a user request to duplicate the first chunk from the search results window into the document authoring window, the computer generates therein an instance of a first segment of the first document, including the first chunk, in its native display format. In response to a user request to duplicate the second chunk from the search results window into the document authoring window, the computer generates therein an instance of a second segment of the second document, including the second chunk, in its native display format. Sometimes, the first document and the second document have different native display formats.

**[00295]** Figure 16D is a screenshot including a PowerPoint slide 1633 in the document authoring window and the slide 1633 corresponds to the relevant chunk 1627 in Figure 16C.

To duplicate this slide 1633 in the document authoring window, the user first selects the checkbox 1631 next to the text-only version of the slide in the search results window and then clicks the duplicate icon 1635 at the top of the search results window.

**[00296]** Figure 16E is another screenshot including not only the PowerPoint slide 1633 but also a paragraph 1643, which corresponds to the relevant chunk 1629 in Figure 16C. To duplicate this paragraph 1643 in the document authoring window, the user first selects the checkbox 1641 next to the text-only version of the paragraph in the search results window and then clicks the duplicate icon 1645 at the top of the search results window.

**[00297]** Note that a PowerPoint document and a Word document are deemed to have different native display formats. But relevant chunks in the search results window are displayed in a text-only format regardless of whether these chunks are identified within a PowerPoint document, a Word document, a plain-text document or even a PDF document. But when a chunk is duplicated into the document authoring window, the computer tries to display the chunk in its native format. Note that a chunk found in a plain-text or PDF document will be customized to a native display format associated with the document authoring window. In other words, if the document authoring window is a Word document authoring window, the chunk is displayed in the Word document's native display format.

**[00298]** In some embodiments, the user may like to display a relevant chunk in its native display format before re-producing the chunk in the document authoring window. For example, in response to a first user selection of the first chunk, the computer launches a first document display window in the application user interface and displays therein a first document that includes the first chunk in its native display format. In response to a second user selection of the second chunk, the computer launches a second document display window in the application user interface and displays therein a second document that includes the second chunk in its native display format.

**[00299]** In some embodiments, the application user interface allows multiple document display windows associated with different document types to exist simultaneously. In some other embodiments, at one time, the application user interface only allows one document display window associated with a document type, e.g., by closing the first document display window before launching the second document display window in response to the second user selection of the second chunk.

**[00300]** In some embodiments, in response to a user request to view the chunk, the computer generates an empty region in the application user interface by shrinking the document authoring window and then occupies the empty region with the document display window in the application user interface.

**[00301]** In some embodiments, the portion of the corresponding document in the document display window includes more information about the search keywords than the chunk in the search results window, such as the location of the search keywords in the corresponding document or the textual contents adjacent to the search keywords in the corresponding document.

**[00302]** Figure 16F is a screenshot including a document display window 1653 in the process of being rendered within the application user interface in response to a user selection of the link 1651. Note that the link 1651 is next to a chunk identified within a PowerPoint document. As shown in Figure 16G, the corresponding slide 1657 is displayed in the document display window and its location 1659 is highlighted in the document display window.

**[00303]** After viewing a chunk in the document display window, the author may want to duplicate the chunk in the document authoring window as well. As shown in Figures 16H-16J, respectively, in response to a user request to copy and paste a segment 1657 of the first document from the first document display window into the document authoring window, the computer generates therein an instance 1661 of the segment of the first document in its native display format; in response to a user request to copy and paste a segment 1663 of the second document display window into the document authoring window, the computer generates therein an instance 1665 of the segment 1663 of the second document in its native display format. This process is similar to the process described above in connection with Figures 16D and 16E.

**[00304]** In some embodiments, the document display window is a preview-only window of the corresponding document (e.g., a PDF document). The user cannot modify the document through the preview-only window. In some other embodiments, the document display window itself is a document authoring window, which may be another instance of the document authoring window (see, e.g., Figure 16I) or may be different from the original

document authoring window (see, e.g., Figure 16G). Sometimes, the search keywords in the document display window are also highlighted.

**[00305]** Another application of the invention is to replace one text string with another text string among a set of documents without having to open any of them. For example, a user may like to change the name of a subject from A to B within many documents of different types that cover the subject. In some cases, the user may like to limit the change to certain types of documents or certain locations within the documents. Currently, the user has to open each document one by one and manually apply the change. This is not only time-consuming but also error-prone.

**[00306]** Figure 17A is a flowchart illustrative of a process of finding and replacing text strings in connection with a set of search results based on user instructions in accordance with some embodiments.

**[00307]** A computer receives a user request to replace a first text string with a second text string in a first document and a second document (1702). The first text string in the first document has a first content type and the first text string in the second document has a second content type, which is different from the first content type. The computer substitutes the second text string for the first text string in the first document and the second document (1704). The replacing second text string in the first document has the first content type and the replacing second text string in the second document has the second content type.

**[00308]** Figure 17B is a flowchart illustrative of a process of finding and replacing text strings within a set of documents based on user instructions in accordance with some embodiments.

**[00309]** After receiving a search request that includes one or more user-specified search keywords (1710), a computer identifies a first document and a second document (1712), each document having at least one chunk that satisfies the search keywords. A first text string in the first document has a first content type and the first text string in the second document has a second content type, which is different from the first content type. After receiving a user request to replace the first text string with a second text string (1714), the computer substitutes the second text string for the first text string in the first document and the second document (1716). The replacing second text string in the first document has the

first content type and the replacing second text string in the second document has the second content type.

**[00310]** Figures 17C through 17E are screenshots of a graphical user interface on a computer display illustrative of features associated with the processes as shown in Figures 17A and 17B in accordance with some embodiments.

**[00311]** Figure 17C is a screenshot including a search assistant window 1722, which occupies the space in the application user interface previously occupied by the document display window (see, e.g., Figure 16J). In some embodiments, the search assistant window 1722 is activated by a user selection of the search assistant icon 1720. The search assistant window 1722 includes three tabs, "Search Options," "History," and "Replace." The "Replace" tab allows a user to replace one text string 1724 ("Albert Einstein" in this example) with another text string 1726 ("A. Einstein" in this example) by clicking the "Update Content" button 1727.

**[00312]** In some embodiments, the "Replace" tab provides additional options 1728 for the user to choose. For example, the user can limit the replacement to the selected search results in the search results window or relevant chunks in the identified documents, which documents result from a search and display of chunks that satisfy user-specified search keywords. Note that the text string 1724 to be replaced does not have to be related to the user-specified search keywords. They can be the same or overlapping (as is the case in Figure 17C) or completely different.

**[00313]** In some embodiments, the user can broaden the scope of the replacement to be the identified documents including, but not limited to, the relevant chunks. In some other embodiments, the user can further expand the scope to cover all the documents whether or not they have a relevant chunk.

**[00314]** In some embodiments, the "Replace" tab also allows the user to specify the locations within a document at which the replacement may happen. For example, Figure 17C depicts target options 1729 that include multiple locations, each having an associated checkbox. Thus, the user can stipulate that the first text string at one or more user-specified locations in the first and second documents be replaced by the second text string by checking the respective checkboxes. As a result, the computer substitutes the second text string for the first text string at the user-specified locations in the first document and the second document,

respectively. Possible locations within a document include one or more selected from the group consisting of title, paragraph, table, header, footer, slide, spreadsheet, and all.

**[00315]** In some embodiments, after identifying the first document and the second document, the computer displays a first chunk from the first document and a second chunk from the second document, each chunk including at least one instance of the first text string. The instances of the first text string within the first and second chunks are displayed in a text-only display format. As described above, a PowerPoint document and a Word document are identified as having chunks satisfying the search phrase "Einstein general relativity." The two relevant chunks are displayed in a text-only display format and different matching terms therein are highlighted in different colors.

**[00316]** In some embodiments, the first and second documents may have different document type. Note that a document's document type is relevant to the document's distinct appearance when the document is rendered through its native application. For example, the first text string in the first document may have a first appearance when the first document is rendered by its native application and the first text string in the second document may have a second appearance that is different from the first appearance when the second document is rendered by its native application.

**[00317]** In this example, the Word document and the PowerPoint document have different document types because their contents have different appearances when rendered by Microsoft Office. Sometimes, a document's suffix may uniquely identify its document type, e.g., a document with the suffix ".docx" is a Microsoft Office 2007 Word document. Sometimes, a document's suffix cannot uniquely identify its document type, e.g., documents like "hello.c" and "hello.java" are probably both plain-text documents and therefore have the same document type.

**[00318]** Figure 17D is a screenshot after the update is completed 1730. In some embodiments, replacing one text string with another text string does not trigger an update of the chunks in the search results window. Thus, the instances 1732, 1734 of the old text string "Albert Einstein" still appear in the search results window. To view the replacing text string, the user has to perform a new search for the replacing text string.

**[00319]** As shown in Figure 17E, in response to a new search request including search keywords 1740 "Einstein general relativity," the computer updates the chunks in the search

results window, and as a result, "Albert Einstein" is replaced with "A. Einstein." Note that the instances 1742, 1744 of the replacing second text string within the first and second chunks are also displayed in the text-only display format.

**[00320]** In some embodiments, after substituting the second text string for the first text string, the computer also replaces the displayed instances of the first text string within the first and second chunks in the search results window with respective instances of the second text string.

**[00321]** In some embodiments, the first document includes an original second text string that has a content type different from the replacing second text string. For example, the Word document may include a PowerPoint slide that has the phrase "A. Einstein," but not the phrase "general relativity." Assuming that the user limits the replacement to the chunks in the search results window, after the update, when the Word document is rendered by Microsoft Office, the second text string has at least two different appearances, one being a Word appearance and the other being a PowerPoint appearance.

**[00322]** Note that the methodology enabling the application of text string finding-and-replacement can be used for implementing other document-editing features such as undoing or reversing last N editing operations (including addition, deletion, and modification) applied to a set of documents and redoing or repeating N editing operations (including addition, deletion, and modification) applied to the set of documents. The set of documents may be located at the same data source or distributed across multiple data sources.

**[00323]** Another application of the invention is to refine search results using different search keywords. For example, after conducting one search using a set of search keywords, a user may like to conduct another search among the documents (or chunks) identified by the first search using another set of search keywords.

**[00324]** Figure 18A is a flowchart illustrative of a first process of narrowing search results based on user instructions in accordance with some embodiments.

**[00325]** After receiving a first user request including a first set of search keywords (1801), a computer identifies a first set of chunks within multiple documents (1803). Each chunk includes terms matching the first set of search keywords. The computer displays at least a portion of the first set of chunks (1805), including highlighting the terms matching the

first set of search keywords in the displayed portion in a first manner. After receiving a second user request to search among the documents for documents that satisfy a second set of search keywords (1807), the computer identifies a second set of chunks within the documents (1809). Each chunk includes terms matching the second set of search keywords. The computer displays at least a portion of the second set of chunks (1811), including highlighting the terms matching the second set of search keywords in the displayed portion in a second manner that is different from the first manner.

**[00326]** Figure 18B is a flowchart illustrative of a second process of narrowing search results based on user instructions in accordance with some embodiments.

**[00327]** After receiving a first user request including a first set of search keywords (1821), a computer identifies multiple documents (1823). Each document includes at least one chunk that satisfies the first set of search keywords. After receiving a second user request to search among the chunks in the identified documents for chunks that satisfy a second set of search keywords (1825), the computer identifies a subset of the chunks (1827). Each chunk in the subset satisfies the second set of search keywords.

**[00328]** Note that a user can repeat any of the two processes above for many times by providing different sets of search keywords for each search step until a predefined condition is met, e.g., the chunks of the user's interest have been found or no chunk is identified. At any time, the user can roll back the search process to a previously-identified set of chunks and try a different set of search keywords that has not been used previously.

**[00329]** Figures 18C through 18D are screenshots of a graphical user interface on a computer display illustrative of features associated with the processes as shown in Figures 18A and 18B in accordance with some embodiments.

**[00330]** The graphical user interface includes a first set of search results displayed in a text-only display format, each search result including one or more chunks identified within a respective document as satisfying a first set of search keywords. In response to a user request to search among the identified chunks for chunks that satisfy a second set of search keywords, the first set of search results is replaced by a second set of search results. Each search result in the second set includes one or more chunks identified within a respective document as satisfying both the first set of search keywords and the second set of search keywords. In some embodiments, two chunks identified within two different documents have

different native display formats. In some embodiments, the second set of search keywords includes at least one search keyword that is not present in the first set of search keywords.

**[00331]** In some embodiments, terms matching the first set of search keywords and terms matching the second set of search keywords within a respective chunk are highlighted in a visually distinguishable manner.

**[00332]** In some embodiments, the chunks identified within a respective document as satisfying the first set of search keywords are displayed in an order consistent with their relative relevancy to the first set of search keywords, and the chunks identified within a respective document as satisfying both the first set of search keywords and the second set of search keywords are displayed in an order consistent with their relative relevancy to the second set of search keywords. In some other embodiments, the chunks identified within a respective document as satisfying any of the first and second sets of search keywords are displayed in an order consistent with their relative locations within the corresponding document.

**[00333]** Figure 18C is a screenshot including a first set of relevant chunks 1833 identified within a PowerPoint document as satisfying the search keyword 1831 "A. Einstein." In some embodiments, the chunks 1833 are ordered by their respective relevancy to the search keywords 1831. In this example, the chunk 1835-B has a relative lower ranking metric when compared with the other chunks above (e.g., 1835-A) and is therefore displayed at the bottom of the search results window. In some embodiments, if the subset of chunks includes a first chunk and a second chunk, the computer displays the first chunk ahead of the second chunk in response to the first user request and displays the second chunk ahead of the first chunk in response to the second user request.

**[00334]** Figure 18D is a screenshot including a second set of relevant chunks 1843 identified within the PowerPoint document as satisfying the search keyword 1841 "gravitation." Note that the second set of search keywords 1841 can be completely different from the first set of search keywords 1831. In this example, the user has selected the checkbox next to the "Search Within Results" icon 1847. Accordingly, the search for the second set of chunks is limited to the documents identified as having chunks that satisfy the search keywords 1831. In this case, it is possible that the second set of chunks includes at least one chunk that is not included in the first set of chunks. In some embodiments, the

search for the second set of chunks is further limited to the chunks 1833 that are identified by the first search.

**[00335]** In some embodiments, the second set of chunks includes at least one chunk that is included in the first set of chunks. For example, the chunks 1845-A, 1845-B in Figure 18D are the same as the respective chunks 1835-A, 1835-B in Figure 18C. In some embodiments, the chunks 1835-A, 1835-B are displayed in an order consistent with their relevancy to the first set of search keywords 1831 in the first set of chunks and the chunks 1845-A, 1845-B are displayed in an order consistent with their relevancy to the second set of search keywords 1841 in the second set of chunks

**[00336]** In some embodiments, the terms in the chunks 1843 matching the first set of search keywords 1831 and the terms in the chunks 1843 matching the second set of search keywords are highlighted in different manner (e.g., different colors, font type, etc.). In this example, the matching terms are displayed using larger, italic, and underlined font.

**[00337]** At any time, if the user is unsatisfied with the identified chunks 1843, the user can bring back the previously-identified chunks by clicking the "Previous" link 1849-A and restart the search process by entering a different set of search keywords. Similarly, the user can skip some search results by clicking the "Next" link 1849-B.

**[00338]** Another application of the invention is to minimize the response latency by alternatively processing different node streams to identify the relevant chunk within a node stream as quickly as possible.

**[00339]** Figure 19 is a flowchart illustrative of a process of alternatively processing document node streams in accordance with some embodiments.

**[00340]** The computer identifies a first candidate document at a first data source and a second candidate document at a second data source in response to a request from a user (1902). The request includes one or more keywords. In some embodiments, the request is a search including one or more search keywords. The computer generates a first node stream for the first candidate document and a second node stream for the second candidate document using data packets received from the respective first and second data sources (1904). The computer alternatively processes the first node stream and the second node stream until a candidate chunk is identified therein (1906). In some embodiments, the candidate chunk

includes a set of nodes within a respective data source. Optionally, the computer returns the candidate chunk as a relevant chunk to the user if the candidate chunk satisfies the keywords (1908). Note that the first data source and the second data source may or may not be the same one. For example, they may be two different web servers. Thus, each candidate document can be an HTML web page.

**[00341]** In some embodiments, the computer submits an HTTP request to the first data source and receives an HTTP response from the first data source. The HTTP response may include multiple data packets corresponding to the first candidate document. After receiving one of the data packets from the first data source, the computer extracts one or more nodes from the data packet and inserts the one or more nodes into the first node stream. Sometimes, the computer may extract only a node fragment from the data packet if the node is too large to fit in a single data packet. In this case, the computer then forms a node by combining the node fragment with another node fragment, which may be extracted from a previous data packet, and insert the formed node (if the node is now complete) into the first node stream.

**[00342]** In some embodiments, after processing nodes currently in the first node stream, the computer waits for more nodes to appear in the first node stream. If no new node appears in the first node stream for a first amount of time, the computer may suspend processing the first node stream and switch to process nodes currently in the second node stream and identify the candidate chunk in the second node stream, if there is any one.

**[00343]** In some embodiments, after processing the nodes currently in the second node stream, the computer may switch back to process nodes currently in the first node stream if no new node appears in the second node stream for the first amount of time and identify the candidate chunk in the first node stream, if there is any one.

**[00344]** In some embodiments, the computer may discard processing results associated with one of the first node stream and the second node stream if no new node appears in the node stream for a second amount of time, which should be no less than and preferably longer than the first amount of time. For example, if there is a network traffic jam and the computer has not received any data packet from a remote data source for a relatively long period of time, the computer can stop working on the corresponding node stream and use the resources associated with the node stream for other purposes, e.g., processing another node stream.

**[00345]** Note that the HTTP-related example above is for illustrative purposes. The process is equally applicable to any communication protocol in which response latency is a concern, such as other TCP/IP based network protocols, file transfer protocol (FTP), or the like.

**[00346]** Another application of the invention is to provide a unified data model for documents having different structure types such as a strictly-structured XML document, a semi-structured HTML web page, and an unstructured plain-text document. This unified data model simplifies the process of identifying relevant chunks therein in response to a set of search keywords.

**[00347]** Figure 20 is a flowchart illustrative of a process of semantically annotating documents of different structures in accordance with some embodiments.

**[00348]** After retrieving a document from a data source (2001), the computer generates a customized data model (e.g., a hierarchical data mode) for the document in accordance with its structure type (2003). In some embodiments, the structure type can be structured, semi-structured, and unstructured. The computer identifies one or more candidate chunks within the customized data model in accordance with a set of heuristic rules associated with the structure type (2005). Optionally, the computer selects one of the candidate chunks that satisfies one or more search keywords and returns it to an end user as a relevant chunk (2007).

**[00349]** In some embodiments, the data source is a web server and the document is an HTML web page that includes multiple pairs of HTML tags. In this case, the computer identifies a first subset of the HTML web page between a first pair of HTML tags as a first candidate chunk if the first pair of HTML tags satisfies one of the set of heuristic rules. If necessary, the computer recursively identifies a second subset of the HTML web page within the first subset of the HTML web page between a second pair of HTML tags as a second candidate chunk if the second pair of HTML tags satisfies one of the set of heuristic rules.

**[00350]** In some embodiments, for a plain-text document, the computer generates the data model by heuristically inserting metadata such as XML tags into the data model. The document contents following different XML tags are identified to be different candidate chunks if they have predefined textual patterns. For example, a paragraph separated by blank

lines is a candidate chunk and a sentence following a hyphen is also a candidate chunk if it is deemed to be part of a list of items.

**[00351]** Another application of the invention is to adaptively select matching chunks from a plurality of candidate chunks identified within a candidate document in response to a search request so as to improve the usability of the chunks to the end user.

**[00352]** As noted above in connection with Figure 12C, because the "'Exact' Match" and "Match All" options require all the search keywords find their matches in a chunk, they may ignore a chunk that, although highly relevant, fails to satisfy one of the search keywords. Alternatively, these two search options may return a chunk that, although satisfying all the search keywords, is too long to retain the benefits an ideal chunk should offer, e.g., being both precise and efficient in locating the information of the user's search interest. The latter case is especially true if the candidate document has a hierarchical data model and the search keywords spread over multiple layers of the data model.

**[00353]** On the other hand, the "Match Any" option accepts any chunk that satisfies at least one search keyword. This could end up with returning too many short chunks to a user, which is equally frustrating because the user has to review many remotely matching chunks before locating the information of the user's search interest or concluding that no such information is in the document.

**[00354]** Fortunately, the "Best Match" option, as will be described below, can successfully avoid the issues associated with these more polarized search options by screening out chunks that are potentially more distractive and presenting only chunks that satisfy a set of carefully-chosen criteria to the user.

**[00355]** Figure 21A is a flowchart illustrative of a first process of screening matching chunks within a candidate document based on predefined criteria in accordance with some embodiments. In this application, a "matching chunk" is defined as a candidate chunk that matches at least one search keyword. Certainly, a matching chunk could be an all-match if it matches all the search keywords and even an exact-match if it matches the search keywords in exactly the same order.

**[00356]** Assume that a set of matching chunks within the candidate document have been identified and they are fed into a computer in an order consistent with their respective

locations in the document. The computer begins the adaptive process by checking if there is any more matching chunk to be further processed (2102). If so (2102, yes), the computer receives the next matching chunk (2104) and checks if the matching chunk meets the current minimum matching level set for the document (2106).

**[00357]** In some embodiments, a matching chunk is characterized by one or more attributes such as its matching level to the corresponding search request and its length. For example, the matching level of a matching chunk may be the total count of unique search keywords found within the chunk and the chunk's length may be the total count of words or non-white-space characters in the chunk. Initially, the computer assigns a minimum matching level, e.g., one unique keyword per chunk, and a range of accepted chunk length, e.g., 50-70 words per chunk, to the candidate document.

**[00358]** If the matching level of the next matching chunk is below the minimum matching level (2106, no), the computer invalidates the matching chunk (2110) and proceeds to the next one in the pipeline. If the matching level of the next matching chunk is above the minimum matching level (2106, yes), the computer checks whether the chunk's length is within the range of accepted chunk length (2108). If the length of the chunk is outside the scope of accepted chunk length (2108, no), either too long or too short, the computer repeats the same procedure of invalidating the matching chunk (2110) and proceeds to the next one in the pipeline.

**[00359]** Otherwise (2108, yes), the computer inserts the matching chunk into a respective queue in accordance with the chunk's match level (2112). In some embodiments, matching chunks having different total counts of unique search keywords are put into separate queues. In some other embodiments, matching chunks having different sets of unique search keywords are grouped into separate queues. In either case, the computer calculates the current total count of matching chunks within the different queues (2113).

**[00360]** If the total count of matching chunks is greater than a predefined threshold, e.g., 10 chunks per document, the computer updates the document's current minimum matching level (2114) by, e.g., increasing the minimum matching level by one. As a result, at least one queue of matching chunks has a matching level less than the updated minimum matching level. In some embodiments, the computer invalidates the entire queue of matching chunks, re-determines the current total count of matching chunks, and repeats this procedure

until the total count of matching chunks is less than the threshold. Certainly, the computer should not invalidate any matching chunk if the total count of matching chunks is less than the predefined threshold.

**[00361]** After updating the current minimum matching level, the computer checks whether the current minimum matching level has reached the maximum matching level associated with the search request (2116). In some embodiments, the maximum matching level is defined by identifying a best-matching chunk such as an all-match chunk or an exact-match chunk. If true (2116, yes), the computer outputs all the best-matching chunks it has accumulated in one or more queues to the user (2118). By doing so, the computer effectively reduces the latency by serving the presumably most relevant chunks to the user while continuously processing the other matching chunks. Otherwise (2116, no), the computer proceeds to the next one in the pipeline. In some embodiments, the operations 2116, 2118 are optional and the computer postpones returning any chunk to the user until after processing all the matching chunks.

**[00362]** At the end of the aforementioned process, the computer should filter out most, if not all, the distractive chunks that are presumably of little interest to the user and is now ready to serve the remaining matching chunks in the queues to the user. Assuming that the computer has queued multiple groups of matching chunks (2120, yes), it begins with serving a group of currently best-matching chunks to the user (2122). After that, the computer checks if the total count of matching chunks that have been served exceeds the predefined threshold or not (2124). If so (2124, yes), the computer stops the process of serving any more matching chunks even if there are additional queues of un-served matching chunks. By keeping the total count of served matching chunks below the threshold, the computer can avoid overwhelming the user with too many chunks in the search results view. Otherwise (2124, no), the computer repeats the process of serving the group of second best-matching chunks until the predefined threshold is met. In some embodiments, the computer stops serving any matching chunk if no more matching chunks are left in any queue (2120, no). This may occur even if the total count of served matching chunks has not reached the predefined threshold.

**[00363]** In some embodiments, the matching chunks identified within a document having a hierarchical data model are queued in an order such that a descendant matching chunk always precedes its ancestor matching chunks if they appear in the same queue. This

ordering guarantees that the computer first serve the more refined descendant matching chunk before encountering any of the ancestor matching chunks because, as noted above, the serving process proceeds from perfect-matching chunks to less perfect ones. After serving the more refined descendant matching chunk, the computer also invalidates all the ancestor matching chunks in the same queue since none of them are presumably more relevant than the descendant chunk.

**[00364]** According to the aforementioned process, the matching chunks are served in an order consistent with their relevancy to the search request, which may be different from the order of the chunks' locations in the document. For example, a best-matching chunk served before the other matching chunks may be located at the end of the document and vice versa. In some embodiments, the computer may apply a two-phase process to ensure that the matching chunks be served in an order consistent with their locations in the candidate document:

- Phase One – The computer screens the matching chunks as described above, including assigning a monotonically increasing chunk identifier to each matching chunk based on the matching chunk's location in the document and invalidating any chunk and its ancestors that fail to meet any of the predefined criteria, without serving any chunk to an end user.

- Phase Two – The computer sorts the surviving matching chunks within different queues in accordance with their respective chunk identifiers such that the first matching chunk to be served is located above the other matching chunks in the same document and outputs the matching chunks in this new sorted order.

**[00365]** Note that there are many other approaches of outputting chunks in an order consistent with their locations in the document. For example, the computer may generate a chunk linked-list during initial data model generation or matching chunk screening process such that each chunk includes a reference to the next chunk in the document. After the screening process, the computer can output the result matching chunks in an order consistent with their locations in the document by navigating the chunk linked-list and skipping invalidated chunks.

**[00366]** Figure 21B is an exemplary HTML document 2130 illustrative of the process as shown in Figure 21A in accordance with some embodiments. For illustration, the HTML document 2130 includes five matching chunks, each chunk having a unique chunk ID "cid."

**[00367]** Assume that there are seven user-specified search keywords, "*Scintillating Examples of the Best Match Algorithm.*" Further assume that the predefined threshold of total chunk count is two (2), the range of accepted chunk length is 30-200 characters, and the initial minimum matching level is one keyword per chunk. The five matching chunks, each satisfying at least one of the seven search keywords, are fed into the computer in the order (as represented by their chunk IDs) of #2, #3, #1, #5, #4.

**[00368]** According to the flow chart shown in Figure 21A, chunks #2 and #3 are both placed in Queue 4, which contains the chunks matching four search keywords, although the two chunks do not have the same four search keywords. Chunk #1 is placed in Queue 6, which contains the chunks matching six search keywords. Since three chunks have been placed into different queues, exceeding the threshold, the computer updates the current minimum matching level from "one keyword per chunk" to "four keywords per chunk."

**[00369]** Although containing four matching keywords, chunk #5 is nonetheless invalidated because its length (26 characters) is outside the range of accepted chunk length. In contrast, chunk #4, which is a parent of chunk #5, is placed in Queue 4 for containing the same four matching keywords and being longer than 30 characters.

**[00370]** After processing all the matching chunks, the computer begins outputting the matching chunks within different queues. In this example, the computer outputs the chunks in an order consistent with their respective relevancy to the search request. Thus, chunk #1 in Queue 6 is first served to the user. As noted above, the export of chunk #1 also causes the invalidation of chunks #2 and #3 in Queue 4 because they are descendants of chunk #1. Because Queue 5 is empty, the computer proceeds to Queue 4, which has only chunk #5 left for output. Finally, the computer stops the process after examining the queues of matching chunks with a matching level no less than the current minimum matching level.

**[00371]** Figure 21C is a flowchart illustrative of a second process of screening matching chunks within a document based on predefined criteria in accordance with some embodiments.

**[00372]** A computer identifies within a document multiple matching chunks in response to a search request from a user (2142). In some embodiments, the search request includes one or more search keywords and each of the multiple matching chunks matches at least one of the search keywords. The computer partitions the matching chunks into multiple groups (2144). The matching chunks within a respective group have an associated matching level to the search request. In some embodiments, the partition is a queuing process wherein chunks containing the same number of matching keywords are placed in the same queue. The computer returns one or more groups of the matching chunks to the user in an order consistent with their respective matching levels to the search request (2136). In some embodiments, the computer displays a respective relevancy indicator adjacent each of the returned matching chunks, indicating the relevancy between the corresponding matching chunk and the search request. The relevancy indicator can be formed using image, text, number or the like to give the user an intuitive impression as to the matching chunk's proximity to the search keywords.

**[00373]** In some embodiments, each of the search keywords has an associated weight indicative of its relevance to the user's search interest. Different search keywords may have the same weight or different weights. Some of the search keywords may even have an associated weight of zero. For instance, in the example described above in connection with Figure 21B, the keyword "the" may be given a weight of zero and therefore have no impact on the search results.

**[00374]** In some embodiments, the matching level of a respective group of matching chunks is, at least partially, determined by summing the weights of unique search keywords within one of the matching chunks. For example, the matching level of a respective group of matching chunks may be, at least partially, determined by the number of unique search keywords within one of the matching chunks. If all the search keywords (including "the") are given the same weight, chunks #2 and #3 would have the same matching level and therefore be put in the same group.

**[00375]** In some embodiments, to partition the matching chunks into multiple groups, the computer selects one of the matching chunks, determining the chunk's matching level and length, and invalidates the chunk if its matching level is less than a minimum matching level or if its length is outside a predefined range of acceptable chunk length. If the selected matching chunk satisfies all the criteria including the minimum matching level and the

predefined range of acceptable chunk length, the computer inserts the chunk into one of the groups of matching chunks. As noted above, the length of the matching chunk can be the total word count of the textual content of the matching chunk, or alternatively, the total character count of the textual content of the matching chunk after white-space normalization.

**[00376]** In some embodiments, after selecting a matching chunk that satisfies all the criteria, the computer compares the chunk's matching level to the matching level of a respective group of matching chunks until identifying a group of matching chunks whose matching levels are the same or similar to the selected chunk's matching level and then adds the chunk to the group of matching chunks.

**[00377]** In some embodiments, after placing a matching chunk within a group or exporting a matching chunk to the end user, the computer checks whether there are any chunks within the same group that are descendants of the newly-placed or newly-exported matching chunk in a hierarchical data model of the document. If so, the computer then invalidates the descendant matching chunks from the group of matching chunks because they are redundant chunks from the user's perspective.

**[00378]** In some embodiments, after inserting one matching chunk into a group of matching chunks, the computer determines a total count of matching chunks whose matching levels are no less than the minimum matching level and updates the current minimum matching level if the total count of matching chunks is greater than a predefined count threshold. Additionally, the computer may invalidate at least a subset of one of the groups of matching chunks whose matching levels are less than the updated minimum matching level.

**[00379]** In some embodiments, if there are multiple groups of matching chunks (e.g., Queue 6 and Queue 4 in the example shown in Figure 21B), the computer selects among the groups of matching chunks a group of matching chunks that has a highest matching level (e.g., Queue 6) and returns the selected group of matching chunks to the user. If there are still groups of matching chunks left, the computer then returns to select a group of matching chunks having a next highest matching level (e.g., Queue 4) until the total count of the returned matching chunks is not less than a predefined count threshold.

**[00380]** Figure 21D is a screenshot of a graphical user interface on a computer display illustrative of features associated with the processes as shown in Figures 21A and 21B in accordance with some embodiments. In this example, the search keywords box includes five

search keywords 2150, "*distance between earth and moon*," and the "Best Match" search option is chosen for selecting matching chunks.

**[00381]** Based on these search keywords, it is not difficult to appreciate that the user is probably interested in knowing the spatial distance between the earth and the moon. But as shown in Figure 21D, the search result 2154 provided by a generic search tool is not satisfactory because it has nothing to do with the answer expected by the user although all the four search keywords are present in the search result (note that the term "and" is treated as a stop-word with no weight).

**[00382]** In contrast, a process according to some embodiments of the invention identifies multiple matching chunks within the same document, 2152-A through 2152-C, different chunks having different numbers of search keywords. In this example, the matching chunks are ordered by their matching levels to the search keywords. Therefore, the matching chunk 2152-A appears before the other two chunks because it includes at least one instance of each of the four search keywords, which is essentially an all-match chunk. But this chunk does not have the answer to the user's question either. Actually, it is the second matching chunk 2152-B that, although having no match for the search keyword "between," has the answer to the user's question, that is, the phrase 2156 "*distance from the Earth to the Moon is 384,403 km.*" Thus, the user receives a satisfactory answer to his or her question from the matching chunks without visiting any of the candidate documents. Note that the same matching chunk 2152-B would have been ignored by the "Match All" and "'Exact' Match" options because it does not have the keyword "between."

**[00383]** Another application of the invention is to search a set of inter-related documents for contents matching a search request. This application is different from the conventional search tools, which always treat the Internet as the search space and perform all the searches in the entire search space no matter how irrelevant most of the documents in the space are to the user-specified search keywords. Consequently, many documents identified by the conventional search tool, although have nothing to do with the user's search interest, end up occupying prominent spots in the search results window. If a user is allowed to narrow the search space to a small set of user-specified documents, it is possible for a computer to produce more relevant search results at a fraction of the cost wasted by the conventional search tools.

**[00384]** Figure 22A is a flowchart illustrative of a process of identifying contents matching a search request within a plurality of inter-related documents in accordance with some embodiments. In this application, a first document is inter-related to a second document if the first document includes a document link that either directly references the second document or indirectly references the second document by referencing a third document that directly or indirectly references the second document. The first document is also inter-related to the second document if they are both directly or indirectly referenced by a third document. As such, different documents referenced by respective document links within an HTML web page are referred to as "inter-related documents." In this case, the HTML web page is called "primary document" and the documents referenced by the web page are called "secondary documents."

**[00385]** A computer receives a request to search one or more secondary documents (2201). At least one of the secondary documents is associated with a primary document. The computer searches at least a subset of the secondary documents for documents that satisfy the search request (2203) and identifies at least one secondary document that satisfies the search request (2205).

**[00386]** In some embodiments, the computer first displays the primary document (e.g., a web page) on a display device (e.g., a computer monitor) before receiving the search request from a user. The primary document includes one or more document links, each document link referencing one of the secondary documents. After identifying the secondary document, which may be another web page or the like, the computer displays at least a portion of the identified secondary document to the user. The displayed portion of the secondary document preferably includes one or more search keywords in the search request.

**[00387]** In some embodiments, the computer locates within the identified secondary document one or more chunks that satisfy the search request using the approaches as described above and displays one or more of the identified chunks to the user.

**[00388]** In some embodiments, the primary document includes many document links pointing to a large number of secondary documents, many of which may have nothing to do with the user's search interest. For example, many web pages include links to boilerplate-type secondary documents such as "About Us," "Contact Us," "Sitemap," "Disclaimer," etc. Searching out these secondary documents rarely returns any useful search results. Thus, in

some embodiments, rather than searching all the secondary documents referenced by the primary document, the user is allowed to select a subset of secondary documents to be searched by identifying document links associated with the user-selected secondary document.

**[00389]** For example, each of the subset of secondary documents can be selected by a respective mouse click of the corresponding document link in the primary document. Alternatively, the computer defines a region in the primary document using an input device and then identifies document links within the defined region as the user-selected document links. For example, the computer presses down a mouse's button at a first location and drags the mouse from one location to another location until releasing the mouse's button at a second location. By doing so, the user-selected region is a rectangle area defined by the first location and the second location and all the document links falling into this rectangle area are document links to secondary documents to be further searched in response to a search request.

**[00390]** In some embodiments, the computer searches both the primary and secondary documents for chunks that satisfy the search request, and as a result, identifies at least one chunk in the primary document and at least one chunk in one of the secondary documents, both chunks satisfying the search request. The chunks associated with the primary and secondary documents are visually separated by a bar such that it is intuitive for a user to distinguish chunks identified within the primary document and chunks identified within the secondary documents.

**[00391]** In some embodiments, the search of secondary documents is a recursive process. In response to a user request to search a secondary document, the computer recursively retrieves the secondary document and documents referenced by this secondary document. Thus, the search results may not only include chunks identified within the primary document but also chunks within a secondary document that is indirectly referenced by the primary document.

**[00392]** Figures 22B through 22D are screenshots of a graphical user interface on a computer display illustrative of features associated with the process as shown in Figure 22A in accordance with some embodiments.

**[00393]** Figure 22B is a screenshot of a web browser window rendering a web page identified by the URL 2211 http://www.rainingdata.com/products/index.html. There are two user-specified search keywords 2213 "tigerlogic xdms" in the search box. The screenshot depicts at least chunks 2217-A through 2217-D that match the two search keywords. The web page includes many document links. Some of the document links (e.g., links 2219) are likely to be related to the search keywords 2213 and others (e.g., links 2220) probably have nothing to do with the search keywords 2213. In this example, the user avoids searching secondary documents associated with the links 2220 by either mouse-clicking the links or defining a rectangle region covering the links.

**[00394]** After a user mouse-click of the "Chunk Page Links" icon 2215, the computer generates a plurality of chunks identified within the primary document and the secondary documents identified by the links 2219 as shown in the screenshot of Figure 22C. Note that the search results 2221 associated with the primary document (including the chunks 2217-A through 2217-C) are separated from the search results 2225 and 2229, which are associated with the two secondary documents identified by the two links 2219, each including a respective set of matching chunks 2227's and 2231's. Figure 22D is another screenshot that only depicts the search results from the secondary documents, nothing from the primary document.

**[00395]** Figure 23 is a block diagram of an exemplary document search server 2300 computer in accordance with some embodiments.

**[00396]** The exemplary document search server 2300 typically includes one or more processing units (CPU's) 2302, one or more network or other communications interfaces 2310, memory 2312, and one or more communication buses 2314 for interconnecting these components. The communication buses 2314 may include circuitry (sometimes called a chipset) that interconnects and controls communications between system components. The document search server 2300 may optionally include a user interface, for instance a display and a keyboard. Memory 2312 may include high speed random access memory and may also include non-volatile memory, such as one or more magnetic disk storage devices. Memory 2312 may include mass storage that is remotely located from the CPU's 2302. In some embodiments, memory 2312 stores the following programs, modules and data structures, or a subset or superset thereof:

- an operating system 2316 that includes procedures for handling various basic system services and for performing hardware dependent tasks;
- a network communication module 2318 that is used for connecting the document search server 2300 to other servers or computers via one or more communication networks (wired or wireless), such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on;
- a system initialization module 2320 that initializes other modules and data structures stored in memory 2312 required for the appropriate operation of the document search server 2300;
- a query engine 2322 for processing a user-driven search query and preparing relevant chunks in response to the search query;
- a cache engine 2324 for identifying candidate documents in response to the search query;
- a stream engine 2326 for retrieving candidate documents and identifying candidate chunks therein; and
- an index database 2328 for storing index information of a number of candidate documents 2330 accessible to the document search server 2300.

**[00397]** Figure 24 is a block diagram of an exemplary client computer 2400 in accordance with some embodiments.

**[00398]** The exemplary client computer 2400 typically includes one or more processing units (CPU's) 2402, one or more network or other communications interfaces 2410, memory 2412, and one or more communication buses 2414 for interconnecting these components. The communication buses 2414 may include circuitry (sometimes called a chipset) that interconnects and controls communications between system components. The client computer 2400 may include a user input device 2410, for instance a display and a keyboard. Memory 2412 may include high speed random access memory and may also include non-volatile memory, such as one or more magnetic disk storage devices. Memory 2412 may include mass storage that is remotely located from the CPU's 2402. In some embodiments, memory 2412 stores the following programs, modules and data structures, or a subset or superset thereof:

- an operating system 2416 that includes procedures for handling various basic system services and for performing hardware dependent tasks;
- a network communication module 2418 that is used for connecting the client computer 2400 to the document search server 2300 or other computers via one or more communication networks (wired or wireless), such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on;
- a system initialization module 2419 that initializes other modules and data structures stored in memory 2412 required for the appropriate operation of the client computer 2400;
- a web browser 2420 for retrieving and displaying candidate documents including web pages from remote web servers;
- a search toolbar 2425 attached to the web browser 2420 for identifying relevant chunks within the retrieved candidate document and displaying the relevant chunks;
- one or more applications 2430 such as Microsoft Office Word application 2431, Microsoft Office PowerPoint application 2433, Microsoft Office Excel application 2435, etc.; and
- an add-in application 2437 attached to the Microsoft Office applications for displaying relevant chunks associated with user-specified search keywords and re-using the relevant chunks based on user instructions.

**Multiple Window User Interface**

**[00399]** As described above with respect to Figures 12-13, the chunks and the document are displayed in a single document displayed in a web browser window. For example, consider the web browser window illustrated in Figure 13E. The web browser window displays a parent document that includes a search box 1340, a frame 1342 including a chunk list, and a frame 1344 including the document that includes the chunks. In other words, the document displayed in the frame 1344 is integrated with the parent document. In some cases, content providers may not desire any modifications to the presentation of their documents. For example, a content provider may object to other websites loading the content provider's document into a frame of a document hosted by other websites. Thus, some embodiments provide a multiple window interface for displaying chunks of documents. In these embodiments, a first window displays the content of the document on which chunks are

identified and a second window displays a search application that identifies the chunks of documents that are relevant to search terms provided by users.

**[00400]** In some embodiments, the second window is a sidebar (or toolbar) of a web browser window. In these embodiments, the first window is the web browser window and the sidebar, including the user interface for the search application, is displayed adjacent to the document in the same web browser window. Note that the discussion below describes embodiments in which the second window is a sidebar (or toolbar) of a web browser and embodiments in which the second window is a separate and distinct window from the web browser window. However, it is noted that either of these embodiments may be used in place of the other. For example, in embodiments described with respect to a sidebar, the sidebar may be replaced with a second window that is separate and distinct from the web browser window, and vice versa.

**[00401]** Figure 25 is a block diagram of an exemplary system 2500 in accordance with some embodiments. The system 2500 includes data sources 2502-2503 and a client computer system 2530 coupled to each other through a network 2520. The network 2520 can generally include any type of wired or wireless communication channel capable of coupling together computing nodes. This includes, but is not limited to, a local area network, a wide area network, or a combination of networks. In some embodiments, the network 2520 includes the Internet. Also note that although only two data sources and one client computer system are illustrated in Figure 25, the system 2500 may include any number of data sources and client computer systems.

**[00402]** The data source 2502 includes one or more documents 2510 including one or more hyperlinks 2511. Similarly, the data source 2503 includes one or more documents 2512 including one or more hyperlinks 2513. As illustrated in Figure 25, the document 2510-1 located on (i.e., hosted by) the data source 2502 includes a hyperlink 2511-1 that links to the document 2512-1 located on (i.e., hosted by) the data source 2503. In some embodiments, the documents 2510 and 2512 are HTML documents. In some embodiments, the data sources 2502-2503 are web servers that host one or more documents. These documents may be stored on the web servers or in a database of the web servers.

**[00403]** The client computer system 2530 includes a web browser 2532 configured to request documents from data sources and present the documents to a user of the client

computer system 2530. In some embodiments, the client computer system 2530 includes a search application 2534 configured to identify and display chunks associated with a document displayed in a window of a user interface for the client computer system 2530 in accordance with search terms provided by a user of the client computer system 2530, as described below with respect to Figures 28-42.

**[00404]** In some embodiments, the data sources 2502-2503 provide the functionality of the search application 2534 on the server-side. In these embodiments, the data sources 2502-2503 provide a search interface that allows users on clients to enter in search terms. For example, the search interface may be included in one or more documents of the data source 2502. A user may enter and submit search terms into the search interface to search for chunks associated with a document being displayed on the user interface of the client computer system 2530 that include at least one of the search terms. The data source 2502 transmits the search terms provided by the user and a document identifier (e.g., a URL) of the document being displayed in the user interface of the client computer system 2530 to one or more servers 2540 including the search application 2534. The servers 2540 include the search application 2534 configured to identify chunks associated with the document being displayed in the user interface of the client computer system 2530 and to return the chunks to the data sources 2502 in accordance with the search terms provided by the user of the client computer system 2530. The data source 2502 then returns a new document that includes a list of identified chunks and a modified version of the document that facilitates navigation to chunks associated with the document (e.g., see Figure 12H). For example, the new document may include code (e.g., JavaScript) that can determine whether a cursor is hovering over an identified chunk or whether the user has clicked on an identified chunk. In response to this determination, the code may then perform an appropriate action (e.g., identifying a hyperlink associated with the chunk or requesting the linked document and highlighting the identified chunk in the linked document, as described herein). In some embodiments, the data sources 2502-2503 include the search application 2534. In these embodiments, the search application 2534 performs the aforementioned operations on the data sources 2502-2503, respectively.

**[00405]** In some embodiments, a user of the client computer system 2530 uses the search application 2534 to search for chunks in linked documents in accordance with search terms provided by the user. For example, the user may request the document 2510-1 from the data source 2502. The user then enters one or more search terms into the search application

2534 and initiates a search of linked documents corresponding to at least a subset of the hyperlinks in the document 2510-1. The search application 2534 requests the linked documents corresponding to at least the subset of the hyperlinks in the document 2510-1 from the document sources specified by the hyperlinks. When the linked documents are received by the search application 2534, the search application 2534 determines whether the linked documents include at least one of the one or more search terms. If so, the search application 2534 identifies chunks within the linked documents that include at least one of the one or more search terms and presents at least a subset of the identified chunks to the user of the client computer system 2530. These embodiments are described in more detail below.

**[00406]** Figure 26 is a block diagram illustrating the client computer system 2530, according to some embodiments. The client computer system 2530 typically includes one or more processing units (CPU's) 2602, one or more network or other communications interfaces 2604, memory 2610, and one or more communication buses 2609 for interconnecting these components. The communication buses 2609 may include circuitry (sometimes called a chipset) that interconnects and controls communications between system components. The client computer system 2530 includes a user interface 2605 comprising a display device 2606 and input devices 2608 (e.g., keyboard, mouse, touch screen, keypads, etc.). Memory 2610 includes high-speed random access memory, such as DRAM, SRAM, DDR RAM or other random access solid state memory devices; and may include non-volatile memory, such as one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid state storage devices. Memory 2610 may optionally include one or more storage devices remotely located from the CPU(s) 2602. Memory 2610, or alternately the non-volatile memory device(s) within memory 2610, comprises a computer readable storage medium. In some embodiments, memory 2610 stores the following programs, modules and data structures, or a subset thereof:

- an operating system 2612 that includes procedures for handling various basic system services and for performing hardware dependent tasks;
- a communication module 2614 that is used for connecting the client computer system 2530 to other computers via the one or more communication interfaces 2604 (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on;

- a user interface module 2616 that receives commands from the user via the input devices 2608 and generates user interface objects in the display device 2606;
- a web browser 2618 that requests documents from document sources (e.g., the document sources 2502-2503), parses and renders the documents on the display device 2606, wherein the web browser 2618 optionally includes the search application 2534 as described below.

**[00407]** Each of the above identified elements may be stored in one or more of the previously mentioned memory devices, and corresponds to a set of instructions for performing a function described above. The set of instructions can be executed by one or more processors (e.g., the CPUs 2602). The above identified modules or programs (i.e., sets of instructions) need not be implemented as separate software programs, procedures or modules, and thus various subsets of these modules may be combined or otherwise re-arranged in various embodiments. In some embodiments, memory 2610 may store a subset of the modules and data structures identified above. Furthermore, memory 2610 may store additional modules and data structures not described above.

**[00408]** Although Figure 26 shows a "client computer system," Figure 26 is intended more as functional description of the various features which may be present in a client computer system than as a structural schematic of the embodiments described herein. In practice, and as recognized by those of ordinary skill in the art, items shown separately could be combined and some items could be separated.

**[00409]** Figure 27 is a block diagram illustrating the server 2540, according to some embodiments. The server 2540 typically includes one or more processing units (CPU's) 2702, one or more network or other communications interfaces 2704, memory 2710, and one or more communication buses 2709 for interconnecting these components. The communication buses 2709 may include circuitry (sometimes called a chipset) that interconnects and controls communications between system components. The server 2540 optionally may include a user interface 2705 comprising a display device 2706 and input devices 2708 (e.g., keyboard, mouse, touch screen, keypads, etc.). Memory 2710 includes high-speed random access memory, such as DRAM, SRAM, DDR RAM or other random access solid state memory devices; and may include non-volatile memory, such as one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid state storage devices. Memory 2710 may optionally include one or

more storage devices remotely located from the CPU(s) 2702. Memory 2710, or alternately the non-volatile memory device(s) within memory 2710, comprises a computer readable storage medium. In some embodiments, memory 2710 stores the following programs, modules and data structures, or a subset thereof:

- an operating system 2712 that includes procedures for handling various basic system services and for performing hardware dependent tasks;
- a communication module 2714 that is used for connecting the server 2540 to other computers via the one or more communication interfaces 2704 (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on;
- an optional user interface module 2716 that receives commands from the user via the input devices 2708 and generates user interface objects in the display device 2706; and
- the search application 2534 as described below.

**[00410]** Each of the above identified elements may be stored in one or more of the previously mentioned memory devices, and corresponds to a set of instructions for performing a function described above. The set of instructions can be executed by one or more processors (e.g., the CPUs 2702). The above identified modules or programs (i.e., sets of instructions) need not be implemented as separate software programs, procedures or modules, and thus various subsets of these modules may be combined or otherwise re-arranged in various embodiments. In some embodiments, memory 2710 may store a subset of the modules and data structures identified above. Furthermore, memory 2710 may store additional modules and data structures not described above.

**[00411]** Although Figure 27 shows a "server," Figure 27 is intended more as functional description of the various features which may be present in a set of servers than as a structural schematic of the embodiments described herein. In practice, and as recognized by those of ordinary skill in the art, items shown separately could be combined and some items could be separated. For example, some items shown separately in Figure 27 could be implemented on single servers and single items could be implemented by one or more servers. The actual number of servers and how features are allocated among them will vary from one implementation to another, and may depend in part on the amount of data traffic

that the system must handle during peak usage periods as well as during average usage periods.

**[00412]** For documents that include a large number of hyperlinks, requesting and searching the documents corresponding to all of the hyperlinks is time-consuming. Thus, in some embodiments, only a subset of the search space is searched so that the generation of search results for the complete search space is deferred until requested by a user. Furthermore, not all linked documents are relevant to the search terms. Thus, in some embodiments, linked documents that are not relevant to the search terms are eliminated from being displayed in the search application 2534. These embodiments are described below with respect to Figures 28-32 and 38-39.

**[00413]** Figure 28 is a flowchart of a method 2800 for requesting and displaying chunks based on search terms, according to some embodiments. The client computer system 2530 displays (2802) a portion of a document (e.g., the document 2510-1) in a first window (e.g., a window of the web browser 2532), wherein the document includes one or more hyperlinks (e.g., hyperlinks 2511) to linked documents at respective data sources (e.g., the documents 2512). In some embodiments, the document is a first web page associated with a first website, wherein at least one of the linked documents is a second web page that is separate and distinct from the first web page. In some embodiments, the document is an HTML document and the one or more hyperlinks include URLs to the linked documents. For example, Figure 30 is a screen shot 3000 of a web browser window 3002 illustrating a user interface for the search application 2534 in a sidebar 3004 of the web browser window 3002 and a portion of a document in a document view window 3006, according to some embodiments. As discussed above, the search application may be displayed in a second window that is separate and distinct from the web browser window 3002 and its corresponding document view window 3006. The following discussion uses both of the terms "sidebar" and "window," and either term may be substituted for the other term. As illustrated in Figure 30, the document includes a plurality of hyperlinks, including hyperlinks 3012 and 3014. The plurality of hyperlinks may include links to sections of the displayed document or links to other documents. The user interface for the search application 2534 includes a search text box 3008 and search options 3011. In some embodiments, the search options 3011 specify the scope of a search and the type of search to be performed. In some embodiments, the scope of the search may include: "Links," which searches documents identified by links of the document being displayed, "Current Page," which only searches the

current page, "Highlighted Links," which searches documents identified by highlighted links of the document being displayed, and "Current Results," which searches the current search results. In some embodiments, the search options 3011 include "Best Match," "Match All," "Exact Match," and "Match Any," as described with respect to Figure 12C.

**[00414]** Returning to Figure 25, the search application 2534 of the client computer system 2530 receives (2804) a search request including one or more search terms from a user of the client computer system 2530. In some embodiments, the second window includes a search box for receiving user-provided search terms. For example, in Figure 30, the user provides a search request including one or more search terms 3010 (e.g., "wep") in the search text box 3008.

**[00415]** In response to a search request including one or more search terms, the search application 2534 requests (2806) one or more of the linked documents from the respective data sources in parallel (e.g., assuming that the current scope of search is "Links"). In some embodiments, the search application 2534 requests a predefined number of linked documents from the respective data sources in parallel. For example, the search application 2534 may request 10 documents in parallel. In some embodiments, requesting the predefined number of linked documents from the respective data sources in parallel includes requesting at least a subset of the predefined number of linked documents prior to receiving at least one of the at least the subset of the predefined number of linked documents. For example, the search application 2534 may spawn the predefined number of threads (or processes) to request the predefined number of linked documents, wherein each spawned thread (or process) requests a distinct linked document from a corresponding data source, and wherein at least a subset of the requests are issued prior to the linked documents being returned to the search application 2534. In some embodiments, the predefined number of linked documents are requested in a predetermined order. In some embodiments, the predetermined order is selected from the group consisting of: the order that the one or more hyperlinks to the linked documents appear in the document as displayed in the first window and the order that the one or more hyperlinks to the linked documents appear in a document object model for the document. In some embodiments, the search application 2534 displays (2808) placeholders for each requested linked document in a second window of the client computer system 2530. For example, Figure 31 is a screen shot 3100 of the web browser window 3002 illustrating a plurality of placeholders for linked documents, including placeholders 3104 and 3106, displayed in the sidebar 3004 of the web browser window 3002. The search application 2534

replaces the placeholders for linked documents that include chunks matching at least one of the one or more search terms with the matching chunks. Placeholders for linked documents that do not include chunks matching the one or more search terms are removed, as described below with respect to steps 2828 and 2830.

**[00416]** Note that the requested linked documents may return to the client computer system 2530 at different times depending on multiple factors including the size of the document, the bandwidth of the connection between the client computer system 2530 and the data sources 2502-2503 (or their proxies), the latency of the connection between the client computer system 2530 and the data sources 2502-2503, the load on the data sources 2502-2503, etc. When a respective linked document is received (2810) from a respective data source, the search application 2534 determines (2812) whether the respective linked document includes chunks that match at least one of the one or more search terms. In response to determining that the respective linked document includes chunks that match at least one of the one or more search terms (2814, yes), the search application 2534 determines (2816) whether a number of groups displayed in the second window is less than a predefined number of groups. In response to determining that the number of groups displayed in the second window is less than the predefined number of groups (2818, yes), the search application 2534 displays (2820) at least a subset of the chunks as a respective group in the second window of the client computer system 2530. In other words, the search application 2534 displays at least a subset of the chunks as a respective group in the second window of the client computer system 2530 only if a number of groups displayed in the second window is less than the predefined number of groups. For example, Figure 32 is a screen shot 3200 of the web browser window 3002 illustrating a plurality of chunks, including chunks 3204 and 3206, identified by the search application 2534 and displayed in the sidebar 3004 of the web browser window 3002. As illustrated in Figure 32, the chunks are grouped so that chunks from the same linked document are in a single group. Assuming that the predefined number of groups is ten, the search application 2534 displays no more than ten groups of chunks (i.e., chunks from ten linked documents that match at least one of the one or more search terms).

**[00417]** In some embodiments, in response to determining that the respective linked document does not include chunks that match at least one of the one or more search terms (2814, yes), the search application 2534 removes (2828) a placeholder corresponding to the respective linked document and requests (2830) a linked document that has not been previously requested or processes a linked document that has been received but not yet

processed, including identifying matching chunks in the document and displaying the chunks as a new group in the second window. Note that if all of the linked documents have been previously requested and processed, the search application 2534 stops processing the linked documents and may update the total number of documents that have the matching chunks if necessary.

**[00418]** In some embodiments, in response to determining that the number of groups to be displayed in the second window is greater than the predefined number of groups (2818, no), the search application 2534 temporarily stops processing a subset of the linked documents and displays (2822) in the second window a next-page link that, when clicked by a user, is configured to repeat the operations described above for linked documents that have not been previously requested or received but not processed (e.g., the "Next" page link 3210 in Figure 32). In some embodiments, in response to determining that the number of groups displayed in the second window is greater than the predefined number of groups (2818, no), the search application 2534 re-queues the linked document for future processing by the search application 2534. In some embodiments, in response to determining that the number of groups displayed in the second window is greater than the predefined number of groups (2818, no), the search application 2534 prefetches subsequent pages of linked documents by continuing to process the linked documents as described above. When the search application 2534 receives (2824) a user selection (e.g., a click) of the next page link, the search application 2534 returns to step 2806 to request one or more linked documents that have not been previously requested, as described above. For example, Figure 38 is a screen shot 3800 of the web browser window 3002 after a user clicked on the next page link 3210, according to some embodiments. The search application 2534 may generate placeholders, including placeholders 3804 and 3806, for linked documents that have not been requested yet. For example, the search application 2534 may generate placeholders for the next ten linked documents that have not been requested yet. Figure 39 is a screen shot 3900 of the web browser window 3002 including chunks identified by the search application 2534 for a second page of linked documents, according to some embodiments. As illustrated in Figure 39, the second page of linked documents includes a plurality of chunks, including chunks 3904 and 3906, page links 3208 and 3910, and next page link 3210. If the user clicks on the next page link 3210, the search application 2534 repeats the operations of Figure 28 for linked documents that have not been previously requested (if any). Note that if the user clicks on the page link 3208 (or any page link corresponding to pages that have been

previously requested), the search application 2534 displays the chunks that were previously identified. In some embodiments, the search application 2534 uses a cached version of the page instead of requesting the linked documents again.

**[00419]** In some embodiments, in response to determining that the number of groups to be displayed in the second window is greater than the predefined number of groups (2818, no), the search application 2534 stops processing at least a subset of the linked documents and displays (2826) in the second window a respective page link corresponding to the groups displayed in the second window (e.g., the page "1" link 3208 in Figure 32).

**[00420]** In some cases, the number of hyperlinks to linked documents in the document being displayed in the web browser window 3002 may exceed the predefined number of groups. In these cases, it is desirable to allow the user to select a page link for linked documents that the search application 2534 has not requested (or searched) yet. Thus, in some embodiments, the search application 2534 generates and displays page links for linked documents that have not been requested yet. Figure 29 is a flowchart of a method 2900 for displaying page numbers for linked documents that have not been searched yet, according to some embodiments. The search application 2534 determines (2902) the number of linked documents that have not been previously requested and calculates (2904) the number of pages required to display groups of chunks for the linked documents that have not been previously requested based on a predefined number of linked documents per page. The search application 2534 then displays (2906) in the second window page links corresponding to each page. In some embodiments, the search application 2534 does not search the linked documents that have not been previously requested until a user selects a page including linked documents that have not been previously requested. In some embodiments, the search application 2534 prefetches and searches the linked documents that have not been previously requested.

**[00421]** The search application 2534 then receives (2908) a user selection (e.g., a click) of a respective page link corresponding to a respective page of the second window. In response to a user clicking on a respective page link corresponding to a respective page of the second window, the search application 2534 repeats (2910) the requesting and the determining operations (e.g., steps 2806-2814) until all pages up to and including the respective page has the predefined number of groups or until there are no more linked documents that have not been previously requested. During this process, some of the linked documents may not include chunks relevant to the one or more search terms and are removed

from the search results, as described above. The removal of linked documents may cause the total number of pages to decrease. Thus, the search application 2534 also removes (2912) page links to pages that are no longer required to display groups of chunks for the linked documents.

**[00422]** In some embodiments, after a chunk is displayed in the sidebar (i.e., the user interface of the search application 2534), the user may interact with the chunk. For example, when the user clicks on a chunk displayed in the sidebar, the search application 2534 causes the web browser to request the linked document associated with the chunk clicked by the user. However, these types of interactions with the sidebar are complicated by the fact that the chunks displayed in the sidebar are identified based on the source file of the document (e.g., the HTML source file), which may have a different structure than the document as displayed in a web browser. This problem arises because not all HTML documents are compliant with web standards. However, web browsers are able to deal with non-compliant HTML documents by making corrections to the HTML documents prior to being displayed in the web browser. Thus, there is often a mismatch between the source file of the document and the document as displayed in the web browser. These corrections may cause a skewing in the addresses of elements. For example, consider the following HTML document illustrated in Table 1:

```
<html>
    <body>
        <p>Text…</p>
        <div>Content…</div>
        <p>More text…</p>
    </body>
</html>
```

Table 1: Exemplary HTML code

**[00423]** The web browser may determine that the HTML code illustrated in Table 1 does not conform to HTML standards. For example, the web browser may decide that HTML "div" elements must be siblings at the top level of the document and performs the following corrections:

```
<html>
        <body>
                <div>
                        <p>Text…</p>
                </div>
                <div>Content…</div>
                <div>
                        <p>More text…</p>
                </div>
        </body>
</html>
```

Table 2: Exemplary HTML code

**[00424]** When identifying chunks, it is desirable to generate an address for the element including the chunk result. This address includes two components: the element type including the chunk, and the position of that element relative to other elements of the same element type. Assume that the chunk that is relevant to the one or more search terms is the chunk " More text… ." The address of the element containing "More text…" is "p2" in the exemplary HTML code illustrated in Table 1 since the element type is "p" and it is the second "p" element in the document. Note that the absolute index of the element amongst all elements may be returned instead. The address of the element containing "More text" is also "p2" in the exemplary HTML code illustrated in Table 2. However, the address for the "div" element from the document has been skewed since two more "div" elements have been introduced. In some embodiments, to correct this skewing, the search application 2534 searches the document as displayed by the web browser to identify the text included in the chunk that is also included in the element type identified by the search application 2534 in the original document. For example, the search application 2534 identifies a "p" element type that includes the text "More text" in the exemplary HTML code illustrated in Table 2.

**[00425]** Note that it is possible that there may be more than one element type that includes the same text as the chunk. Typically, finding multiple instances of the chunk does not present a problem because the chunks include the search terms requested by the user. However, a particular chunk out of the multiple instances of the chunk may be identified by counting the number of instances of the chunk from the start of the original document until the particular chunk is reached. Next, the particular chunk in the document as displayed is identified by counting the number of instances of the chunk from the start of the document as displayed. The chunk result is then returned. For example, assume that the text of the chunk is "More Text" and the element type is "p." Furthermore, assume that there are ten chunks

that match the text "More Text" and that are included in the element type "p." If the user selects the fifth chunk out of the ten chunks as identified in the search of the original document, the fifth chunk in the document as displayed that matches the text "More Text" and that is included in the element type "p" is returned.

**[00426]** Attention is now directed to Figure 40, which is a flowchart of a method 4000 for highlighting chunks of linked documents in the web browser, according to some embodiments. The search application 2534 displays (4002) a portion of a first document in a first window and one or more chunks extracted from a second document in a second window, wherein the first document includes one or more hyperlinks to the second document. Note that in this particular embodiment, both the first window and the second window may be parts of another window such as a web browser window.

**[00427]** In some embodiments, the search application 2534 receives (4004) a user interface event indicating that a cursor is hovering over a user-selected chunk in the second window. The search application 2534 identifies (4006) a portion of the first document that includes a URL to the second document including the user-selected chunk and highlights (4008) the URL (i.e., the URL to the second document) of the first document in the first window. For example, Figure 33 is a screen shot 3300 of the web browser window 3002 illustrating a cursor 3304 hovering over the chunk 3204 in the search application, according to some embodiments. The hyperlink 3306 (e.g., the URL) to the second document that includes the chunk 3204 is identified in the document displayed in the document view window 3006 (i.e., the first document). In Figure 33, the hyperlink 3306 is identified using a number (e.g., "1") corresponding to the chunk group including to the chunk 3204. Similarly, Figure 34 is a screen shot 3400 of the web browser window 3002 illustrating a cursor 3304 hovering over the chunk 3206 in the search application, according to some embodiments. Since the chunk 3206 is included in the same chunk group as the chunk 3204, the hyperlink 3306 is identified in the document view window 3006. Furthermore, Figure 35 is a screen shot 3500 of the web browser window 3002 illustrating a cursor 3304 hovering over the chunk 3504 in the search application, according to some embodiments. Since the chunk 3504 is included in a chunk group (e.g., "2") that is different than the chunk groups illustrated in Figures 32 and 33, the hyperlink 3506 is identified in the document view window 3006. In some embodiments, the search application 2534 identifies all of the hyperlinks corresponding to the one or more chunks regardless of whether the cursor is hovering over the one or more chunks. In some embodiments, the search application 2534 identifies a subset of the

hyperlinks corresponding to the one or more chunks regardless of whether the cursor is hovering over the one or more chunks.

**[00428]** In some cases, a chunk may correspond to a hyperlink that is in a portion of the document not currently being displayed in the document view window 3006. Thus, in some embodiments, when the search application 2534 receives (4004) a user interface event indicating that a cursor is hovering over a user-selected chunk in the second window (e.g., the sidebar of the web browser window 3002), the search application 2534 updates (4010) a displayed portion of the first document in the first window to include a portion of the first document that includes the URL to the second document including the user-selected chunk.

**[00429]** The search application 2534 then receives (4012) a selection (e.g., a click) of a user-selected chunk in the second window. In response to receiving a selection of a user-selected chunk in the second window, the search application 2534 identifies (4014) a first element type defined in the second document that includes the user-selected chunk. The search application 2534 then displays (4016) at least a portion of the second document in the first window and identifies (4018) a second element type of the second document as displayed in the first window that has the same element type as the first element type and that includes the user-selected chunk. The search application 2534 then updates (4020) the display of the first window and the second window by displaying (4022) a portion of the second document that includes the second element type including the user-selected chunk in the first window, highlighting (4024) the second element type that includes the user-selected chunk in the first window, and highlighting (4026) the user-selected chunk in the second window. For example, Figure 36 is a screen shot 3600 of the web browser window 3002 after a user clicked on the chunk 3204 in the search application, according to some embodiments. The search application 2534 causes the web browser to request the second document corresponding to the chunk group including the chunk 3204. The web browser then displays the second document in the document view window 3006 and the search application 2534 identifies the chunk 3604 corresponding to the chunk 3204 within the second document. Similarly, Figure 37 is a screen shot 3700 of the web browser window 3002 after a user clicked on the chunk 3206 in the search application, according to some embodiments. Since the chunk 3206 is also in the second document, the search application 2534 only needs to update the display to identify and display the chunk 3704 corresponding to the chunk 3206. In some embodiments, the chunks are highlighted in a manner that is distinct from the highlighting in Figures 33-35. For example, the highlighting illustrated in

Figures 36-37 applies a shadow effect to the chunk including the one or more search terms and labels the chunk using a chunk identifier (e.g., "B") corresponding to the chunk identifier in the sidebar 3004. Other highlighting techniques may be used. For example, the chunk may be highlighted using a different background color, a different font color, etc. In some embodiments, the search application 2534 highlights all of the chunks in a document corresponding to the user-selected chunk. In some embodiments, the search application 2534 highlights a subset of the chunks in a document corresponding to the user-selected chunk.

**[00430]** As discussed above, a document may include a large number of hyperlinks. These hyperlinks may include low-quality hyperlinks that do not produce search results. For example, these low-quality hyperlinks may include navigation links (e.g., links to other sections of the web site), login links (e.g., a link to a web page that allows a user to log into the web site), and the like, which typically do not generate any content of value. Unfortunately, there is no standard way to tag these types of hyperlinks with any semantic notion of being navigation links, login links, or the like. Although these types of low-quality hyperlinks may be easily identified by a user, it is often difficult to identify these links programmatically. Thus, some embodiments identify and eliminate low-quality links from the search process. These embodiments are described in more detail with respect to Figures 41 and 42.

**[00431]** Figure 41 is a flowchart of a method 4100 for ranking hyperlinks to linked documents, according to some embodiments. The search application 2534 displays (4102) a portion of a document in a first window, the document including one or more hyperlinks to linked documents at respective data sources. The search application 2534 then identifies (4104) one or more chunks in the document, wherein a respective chunk includes at least one of the one or more hyperlinks and ranks (4106) the one or more chunks based at least in part on properties of the one or more hyperlinks in the one or more chunks and properties of the one or more chunks. In some embodiments, the properties of a respective hyperlink are selected from the group consisting of: a target of the respective hyperlink, the number of words in anchor text for the respective hyperlink, and the meaning of the words in anchor text for the respective hyperlink. For example, hyperlinks to the same document or anchor text for hyperlinks that include only a few words may be indicative of a navigation menu. In some embodiments, the properties of a respective chunk are selected from the group consisting of: a ratio of raw text to hyperlink anchor text in the respective chunk, and a location of the respective chunk in the document. For example, a chunk with a low ratio of

raw text to hyperlink anchor text may be indicative of a navigation menu. Consider the following bulleted list of hyperlinks links:

- Home
- Contact Us

The presence of a list of very short hyperlink anchor text (e.g., links having hyperlink anchor text with less than three terms), with no intervening raw text, leads to the assessment that the links in this list are navigation links.

**[00432]** In some embodiments, the search application 2534 receives (4108) a search request including one or more search terms provided by a user in a search box displayed in the second window.

**[00433]** The search application 2534 obtains (4110) one or more linked documents from the respective data sources in accordance with respective rankings of the one or more chunks that include respective hyperlinks to the one or more linked documents and displays (4112) at least a portion of one of the one or more linked documents in a second window. Step 4112 is described in more detail with respect to Figure 42, which is a flowchart of a method 4112 for displaying chunks in the search application, according to some embodiments. The search application 2534 determines (4202) whether the respective linked document includes chunks that match at least one of the one or more search terms. In response to determining that the respective linked document includes chunks that match at least one of the one or more search terms (4204, yes), the search application 2534 displays (4206) at least a subset of the chunks as a respective group in the second window. In response to determining that the respective linked document does not include chunks that match at least one of the one or more search terms (4204, no), the search application 2534 does not display (4208) the respective document in the second window.

**[00434]** In some embodiments, the document includes nested chunks. In these embodiments, each child chunk inherits an initial ranking from its parent. For example, consider two element types in HTML: "ul" and "li". The "ul" element represents an unordered list, while the "li" element represents an individual item in the list. An unordered list is a distinct entity in a document, and therefore maintains a ranking distinct from the area in which it is nested. By contrast, a list item's ranking is directly related to that of its siblings and of the list as a whole. Thus, the "li" element inherits its initial ranking from its parent "ul" element. After the ranking of the child chunk has been assessed, ranking of the child chunk is aggregated into the ranking of the corresponding parent chunk.

**[00435]** In some embodiments, a blog page is identified based on a predefined pattern of a blog page. For example, the predefined pattern may be that blog pages have a high text-to-hyperlink anchor text ratio in the center of the document and a cluster of hyperlinks with a similar patterns (e.g., the target is the same domain, etc.) down a side. Note that other types of pages may be identified using predefined patterns.

**[00436]** Note that for the sake of clarity, the methods 2800, 2900, 4000, 4100, and 4112 were described with respect to the client computer system 2530 performing the operations. However, a person of ordinary skill in the art may modify these operations to be performed by the servers 2540.

**[00437]** The methods 2800, 2900, 4000, 4100, and 4112 may be governed by instructions that are stored in a computer readable storage medium and that are executed by one or more processors of one or more servers. Each of the operations shown in Figures 28, 29, 40, 41, and 42 may correspond to instructions stored in a computer memory or computer readable storage medium. The computer readable storage medium may include a magnetic or optical disk storage device, solid state storage devices such as Flash memory, or other non-volatile memory device or devices. The computer readable instructions stored on the computer readable storage medium are in source code, assembly language code, object code, or other instruction format that is interpreted and/or executable by one or more processors.

**[00438]** The foregoing description, for purpose of explanation, has been described with reference to specific embodiments. However, the illustrative discussions above are not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations are possible in view of the above teachings. For example, the aforementioned processes of identifying a relevant chunk within a document are by no means limited to a particular language such as English. Actually, the same processes are equally applicable to documents written in other languages and/or multi-lingual documents. The embodiments were chosen and described in order to best explain the principles of the invention and its practical applications, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated.

What is claimed is:

1. A computer-implemented method for grouping and displaying chunks, comprising:

at a computer system including one or more processors and memory storing one or more programs, the one or more processors executing the one or more programs to perform the operations of:

displaying a portion of a document in a first window, the document including one or more hyperlinks to linked documents at respective data sources;

in response to a search request including one or more search terms,

requesting one or more of the linked documents from the respective data sources in parallel;

when a respective linked document is received from a respective data source,

determining whether the respective linked document includes chunks that match at least one of the one or more search terms; and

in response to determining that the respective linked document includes chunks that match at least one of the one or more search terms, displaying at least a subset of the chunks as a respective group in a second window only if a number of groups displayed in the second window is less than a predefined number of groups.

2. The method of claim 1, wherein in response to determining that the respective linked document does not include chunks that match at least one of the one or more search terms, the method further comprises:

requesting linked documents that have not been previously requested until the number of groups displayed in the second window is the predefined number of groups or until there are no more linked documents that have not been previously requested.

3. The method of claim 1, further comprising:

after the predefined number of groups has been displayed, displaying in the second window a respective page link corresponding to the groups displayed in the second window.

4. The method of claim 1, further comprising:

after the predefined number of groups has been displayed, displaying in the second window a next-page link that when clicked by a user is configured to perform the operations of claim 1 for linked documents that have not been requested.

5. The method of claim 4, further comprising:

in response to a user clicking on the next-page link, performing the operations of claim 1 for the linked documents that have not been previously requested.

6. The method of claim 1, further comprising:

determining the number of linked documents that have not been previously requested;

calculating the number of pages required to display groups of chunks for the linked documents that have not been previously requested based on a predefined number of linked documents per page; and

displaying in the second window page links corresponding to each page.

7. The method of claim 6, further comprising:

in response to a user clicking on a respective page link corresponding to a respective page of the second window, repeating the requesting and the determining operations until all pages up to and including the respective page has the predefined number of groups or until there are no more linked documents that have not been previously requested.

8. The method of claim 7, further comprising:

removing page links to pages that are no longer required to display groups of chunks for the linked documents.

9. The method of claim 1, wherein requesting one or more of the linked documents from the respective data sources in parallel includes requesting a predefined number of linked documents from the respective data sources in parallel.

10. The method of claim 9, wherein prior to receiving the linked documents from respective data sources, the method further comprises:

displaying in the second window placeholders for each requested linked document.

11. The method of claim 10, wherein in response to determining that the respective linked document does not include chunks that match at least one of the one or more search terms, the method further comprises:

removing a placeholder corresponding to the respective linked document.

12. The method of claim 9, wherein the predefined number of linked documents are requested in a predetermined order.

13. The method of claim 12, wherein the predetermined order is selected from the group consisting of:

the order that the one or more hyperlinks to the linked documents appear in the document as displayed in the first window; and

the order that the one or more hyperlinks to the linked documents appear in a document object model for the document.

14. The method of claim 1, wherein the document is a first web page associated with a first website, and wherein at least one of the linked documents is a second web page that is separate and distinct from the first web page.

15. A system for grouping and displaying chunks, comprising:

one or more processors;

memory; and

one or more programs stored in the memory, the one or more programs comprising instructions to:

display a portion of a document in a first window, the document including one or more hyperlinks to linked documents at respective data sources;

in response to a search request including one or more search terms,

request one or more of the linked documents from the respective data sources in parallel;

when a respective linked document is received from a respective data source,

determine whether the respective linked document includes chunks that match at least one of the one or more search terms; and

in response to determining that the respective linked document includes chunks that match at least one of the one or more search terms, display at least a subset of the chunks as a respective group in a second window only if a number of groups displayed in the second window is less than a predefined number of groups.

16. A computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs comprising instructions to:

display a portion of a document in a first window, the document including one or more hyperlinks to linked documents at respective data sources;

in response to a search request including one or more search terms,

request one or more of the linked documents from the respective data sources in parallel;

when a respective linked document is received from a respective data source,

determine whether the respective linked document includes chunks that match at least one of the one or more search terms; and

in response to determining that the respective linked document includes chunks that match at least one of the one or more search terms, display at least a subset of the chunks as a respective group in a second window only if a number of groups displayed in the second window is less than a predefined number of groups.

17. A computer-implemented method for highlighting a location of a document chunk within a document, comprising:

at a computer system including one or more processors and memory storing one or more programs, the one or more processors executing the one or more programs to perform the operations of:

displaying a portion of a first document in a first window and one or more chunks extracted from a second document in a second window, wherein the first document includes one or more hyperlinks to the second document;

in response to receiving a selection of a user-selected chunk in the second window,

identifying a first element type defined in the second document that includes the user-selected chunk;

displaying at least a portion of the second document in the first window;

identifying a second element type of the second document as displayed in the first window that has the same element type as the first element type and that includes the user-selected chunk; and

updating the display of the first window and the second window by:

displaying a portion of the second document that includes the second element type including the user-selected chunk in the first window;

highlighting the second element type that includes the user-selected chunk in the first window; and

highlighting the user-selected chunk in the second window.

18. The method of claim 17, wherein the one or more chunks are extracted from the second document in response to a search request including one or more search terms submitted by a user through the computer system.

19. The method of claim 18, wherein the one or more search terms are highlighted in the second window in a manner distinct from that of highlighting the user-selected segment.

20. The method of claim 17, wherein the first document is an HTML document, and wherein the one or more hyperlinks include at least one URL to the second document.

21. The method of claim 20, wherein prior to receiving the selection of the user-selected chunk in the second window and in response to a cursor hovering over the user-selected chunk in the second window, the method further comprises:

identifying a portion of the first document that includes a URL to the second document including the user-selected chunk; and

highlighting the URL of the first document in the first window.

22. The method of claim 20, wherein prior to receiving the selection of the user-selected chunk in the second window and in response to a cursor hovering over the user-selected chunk in the second window, the method further comprises:

updating a displayed portion of the first document in the first window to include a portion of the first document that includes the URL to the second document including the user-selected chunk.

23. A system for highlighting a location of a document chunk within a document, comprising:

one or more processors;

memory; and

one or more programs stored in the memory, the one or more programs comprising instructions to:

display a portion of a first document in a first window and one or more chunks extracted from a second document in a second window, wherein the first document includes one or more hyperlinks to the second document;

in response to receiving a selection of a user-selected chunk in the second window,

identify a first element type defined in the second document that includes the user-selected chunk;

display at least a portion of the second document in the first window;

identify a second element type of the second document as displayed in the first window that has the same element type as the first element type and that includes the user-selected chunk; and

update the display of the first window and the second window by:

display a portion of the second document that includes the second element type including the user-selected chunk in the first window;

highlight the second element type that includes the user-selected chunk in the first window; and

highlight the user-selected chunk in the second window.

24. A computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs comprising instructions to:

display a portion of a first document in a first window and one or more chunks extracted from a second document in a second window, wherein the first document includes one or more hyperlinks to the second document;

in response to receiving a selection of a user-selected chunk in the second window,

identify a first element type defined in the second document that includes the user-selected chunk;

display at least a portion of the second document in the first window;

identify a second element type of the second document as displayed in the first window that has the same element type as the first element type and that includes the user-selected chunk; and

update the display of the first window and the second window by:

display a portion of the second document that includes the second element type including the user-selected chunk in the first window;

highlight the second element type that includes the user-selected chunk in the first window; and

highlight the user-selected chunk in the second window.

25. A computer-implemented method for ranking hyperlinks of a document, comprising:

at a computer system including one or more processors and memory storing one or more programs, the one or more processors executing the one or more programs to perform the operations of:

displaying a portion of a document in a first window, the document including one or more hyperlinks to linked documents at respective data sources;

identifying one or more chunks in the document, wherein a respective chunk includes at least one of the one or more hyperlinks;

ranking the one or more chunks based at least in part on properties of the one or more hyperlinks in the one or more chunks and properties of the one or more chunks;

obtaining one or more linked documents from the respective data sources in accordance with respective rankings of the one or more chunks that include respective hyperlinks to the one or more linked documents; and

displaying at least a portion of one of the one or more linked documents in a second window.

26. The method of claim 25, wherein the properties of a respective hyperlink are selected from the group consisting of:

a target of the respective hyperlink;

the number of words in anchor text for the respective hyperlink; and

the meaning of the words in anchor text for the respective hyperlink.

27. The method of claim 25, wherein the properties of a respective chunk are selected from the group consisting of:

a ratio of raw text to hyperlink anchor text in the respective chunk; and

a location of the respective chunk in the document.

28. The method of claim 25, wherein prior to obtaining the one or more linked documents from the respective data sources, the method further comprises:

receiving a search request including one or more search terms provided by a user in a search box displayed in the second window.

29. The method of claim 28, wherein displaying at least a portion of a respective linked document in the second window includes:

determining whether the respective linked document includes chunks that match at least one of the one or more search terms; and

in response to determining that the respective linked document includes chunks that match at least one of the one or more search terms, displaying at least a subset of the chunks as a respective group in the second window.

30. The method of claim 25, wherein at least a subset of the one or more chunks include nested chunks, and wherein the method further comprises:

aggregating rankings of child chunks into rankings of corresponding parent chunks.

31. A system for ranking hyperlinks of a document, comprising:

one or more processors;

memory; and

one or more programs stored in the memory, the one or more programs comprising instructions to:

display a portion of a document in a first window, the document including one or more hyperlinks to linked documents at respective data sources;

identify one or more chunks in the document, wherein a respective chunk includes at least one of the one or more hyperlinks;

rank the one or more chunks based at least in part on properties of the one or more hyperlinks in the one or more chunks and properties of the one or more chunks;

obtain one or more linked documents from the respective data sources in accordance with respective rankings of the one or more chunks that include respective hyperlinks to the one or more linked documents; and

display at least a portion of one of the one or more linked documents in a second window.

32. A computer readable storage medium storing one or more programs configured for execution by a computer, the one or more programs comprising instructions to:

display a portion of a document in a first window, the document including one or more hyperlinks to linked documents at respective data sources;

identify one or more chunks in the document, wherein a respective chunk includes at least one of the one or more hyperlinks;

rank the one or more chunks based at least in part on properties of the one or more hyperlinks in the one or more chunks and properties of the one or more chunks;

obtain one or more linked documents from the respective data sources in accordance with respective rankings of the one or more chunks that include respective hyperlinks to the one or more linked documents; and

display at least a portion of one of the one or more linked documents in a second window.

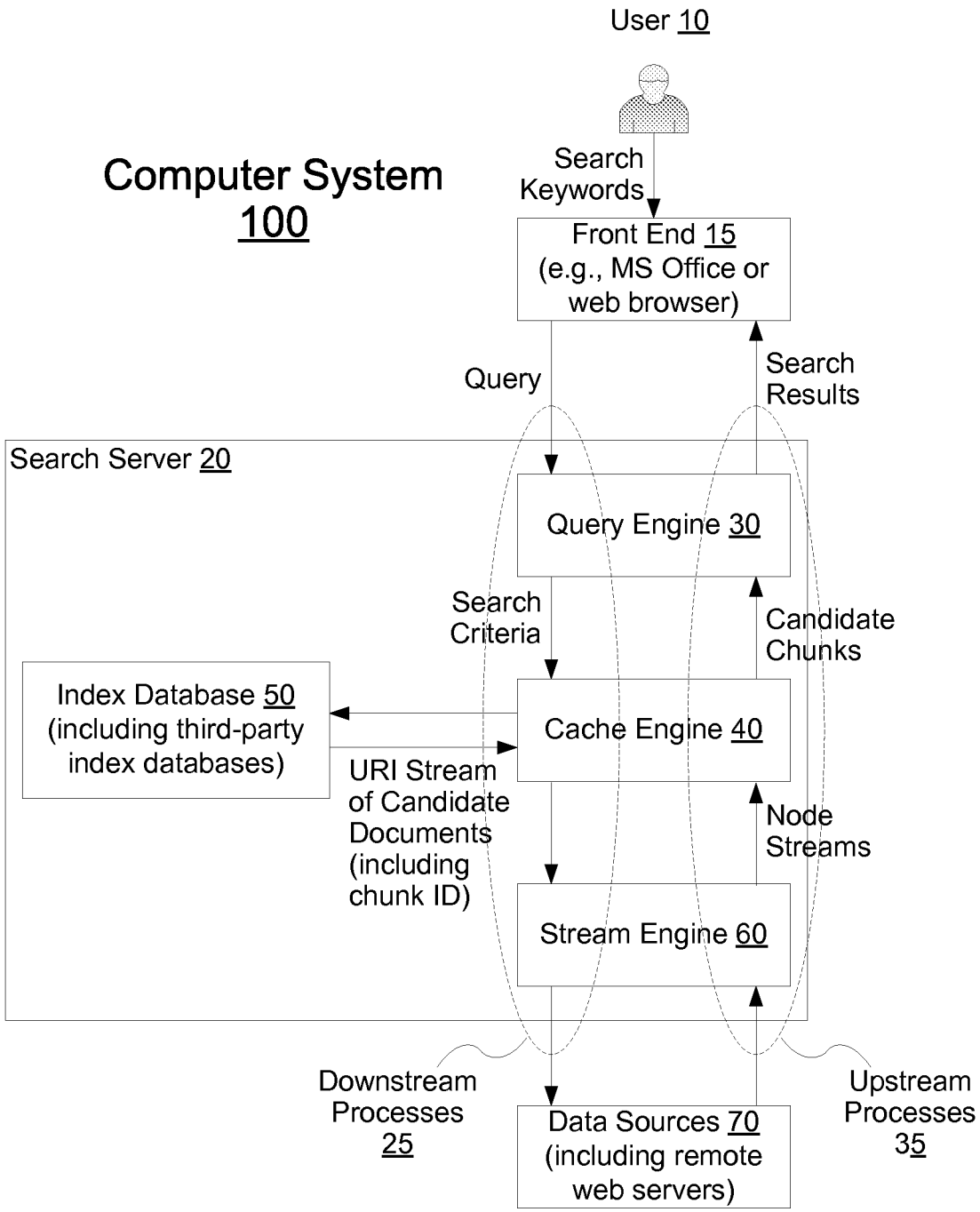

User 10
Computer System 100
Search Keywords
Front End 15 (e.g., MS Office or web browser)
Query
Search Results
Search Server 20
Query Engine 30
Search Criteria
Candidate Chunks
Index Database 50 (including third-party index databases)
Cache Engine 40
URI Stream of Candidate Documents (including chunk ID)
Node Streams
Stream Engine 60
Downstream Processes 25
Upstream Processes 35
Data Sources 70 (including remote web servers)

Figure 1

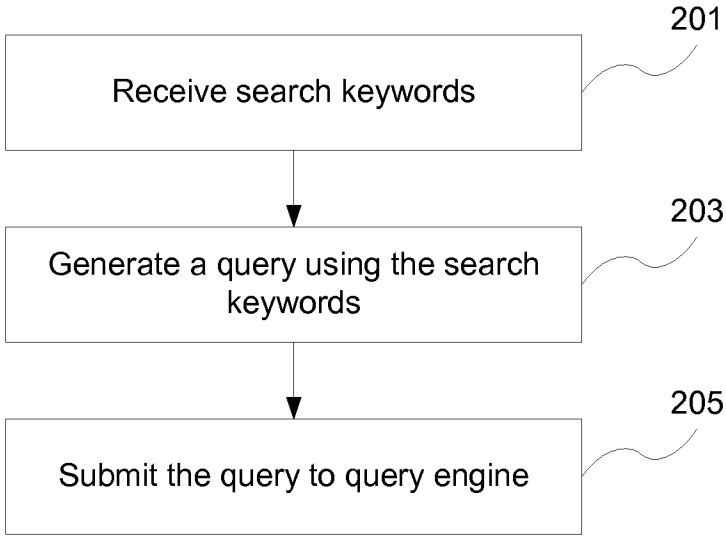



Figure 2

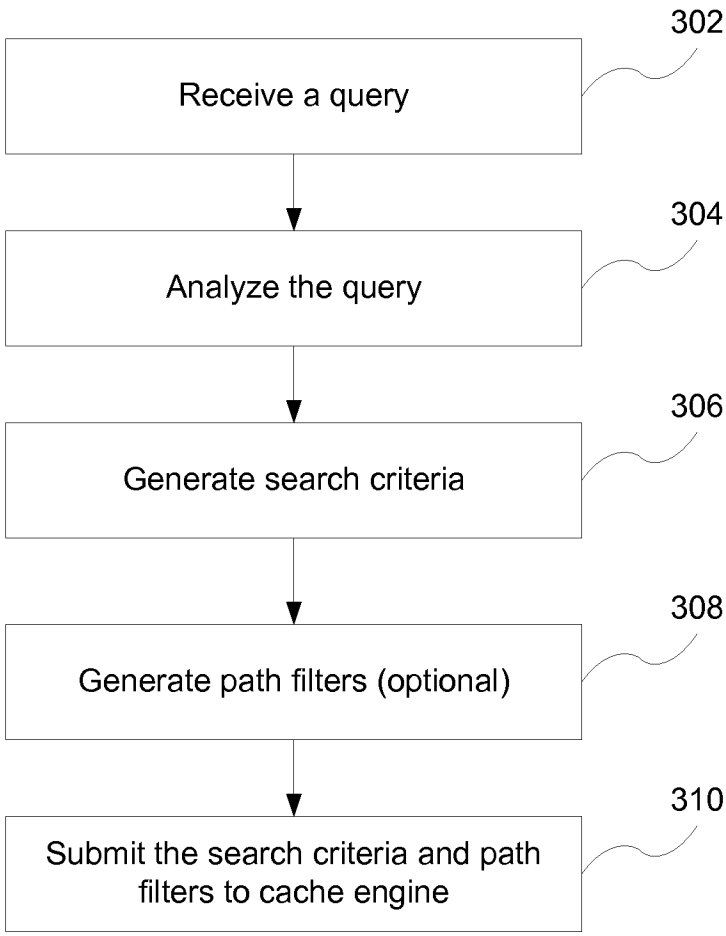



Figure 3

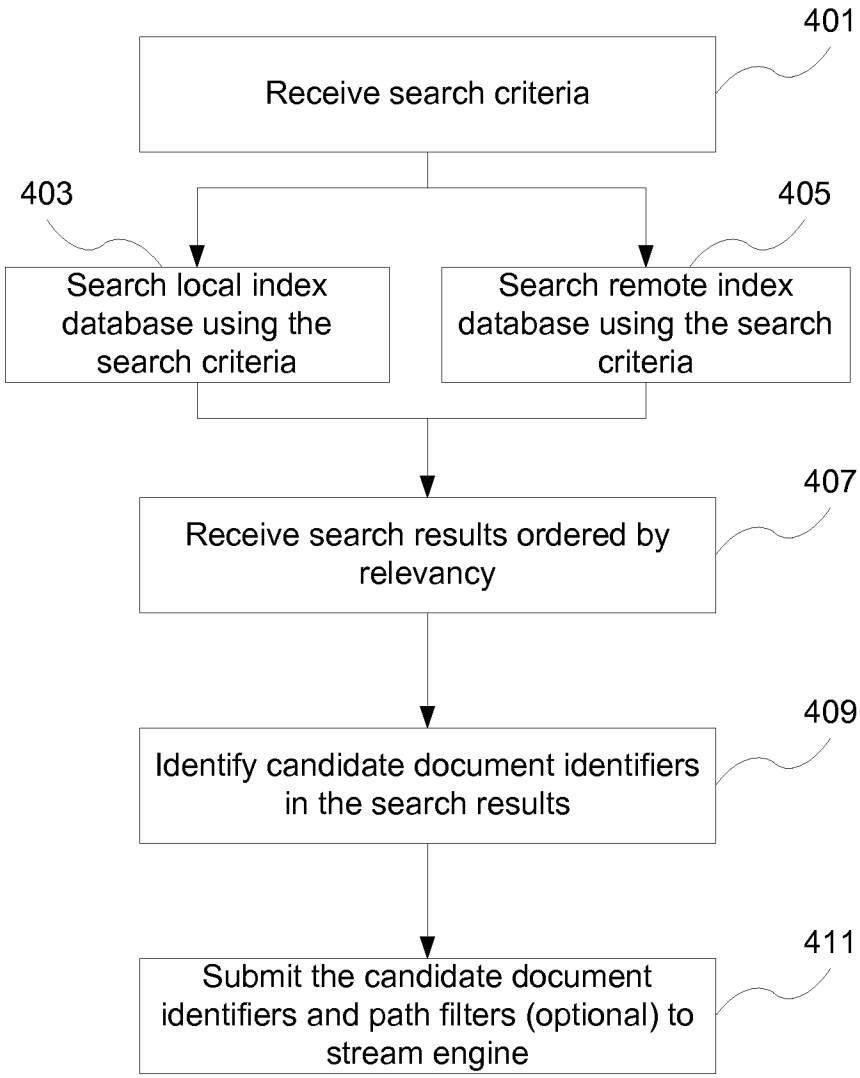

401
Receive search criteria
403
405
Search local index database using the search criteria
Search remote index database using the search criteria
407
Receive search results ordered by relevancy
409
Identify candidate document identifiers in the search results
411
Submit the candidate document identifiers and path filters (optional) to stream engine

Figure 4

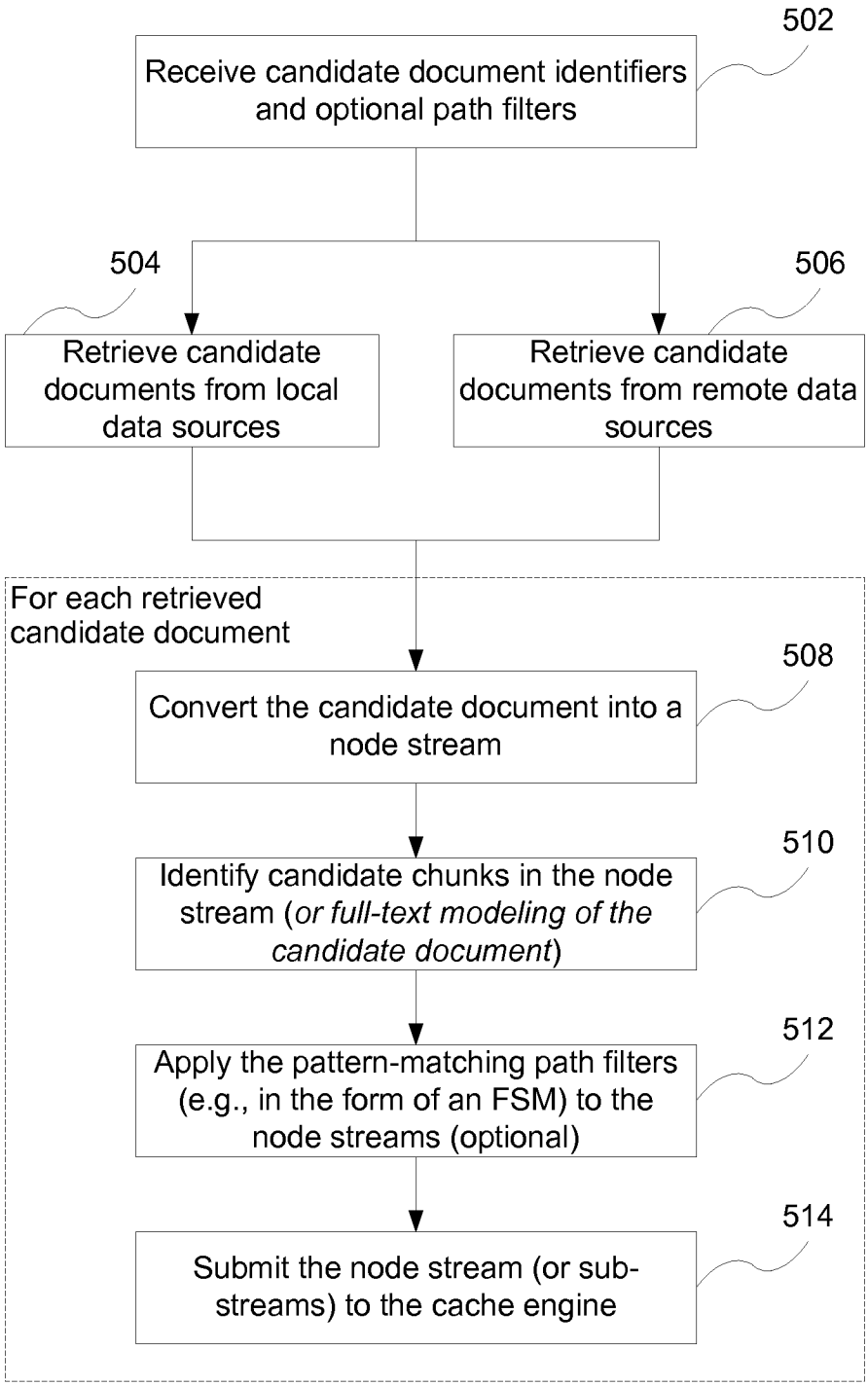

502
Receive candidate document identifiers and optional path filters
504
506
Retrieve candidate documents from local data sources
Retrieve candidate documents from remote data sources
For each retrieved candidate document
508
Convert the candidate document into a node stream
510
Identify candidate chunks in the node stream (or full-text modeling of the candidate document)
512
Apply the pattern-matching path filters (e.g., in the form of an FSM) to the node streams (optional)
514
Submit the node stream (or sub-streams) to the cache engine

Figure 5

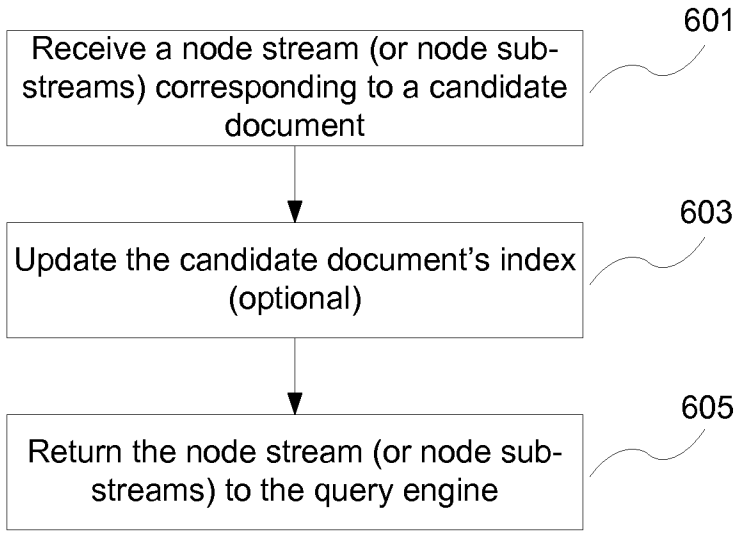



Figure 6

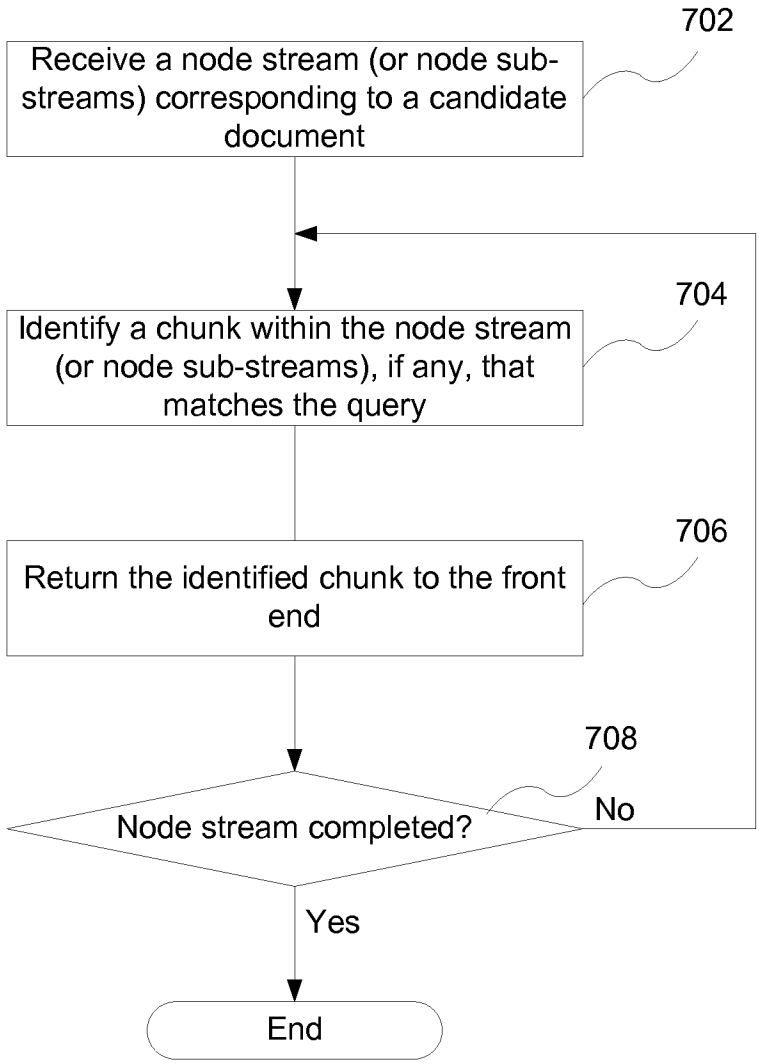



Figure 7

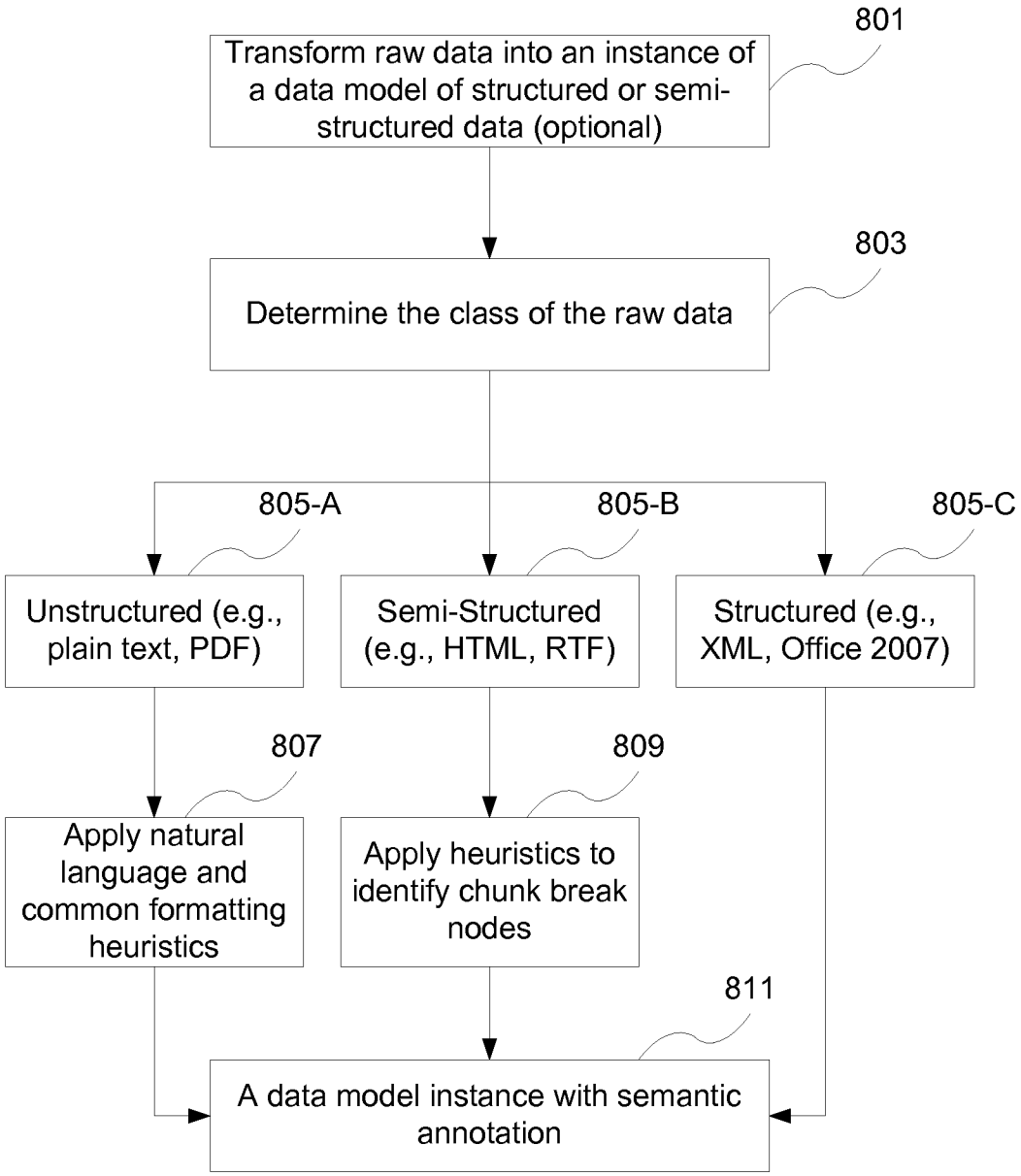

801
Transform raw data into an instance of a data model of structured or semi-structured data (optional)
803
Determine the class of the raw data
805-A
805-B
805-C
Unstructured (e.g., plain text, PDF)
Semi-Structured (e.g., HTML, RTF)
Structured (e.g., XML, Office 2007)
807
809
Apply natural language and common formatting heuristics
Apply heuristics to identify chunk break nodes
811
A data model instance with semantic annotation

Figure 8A

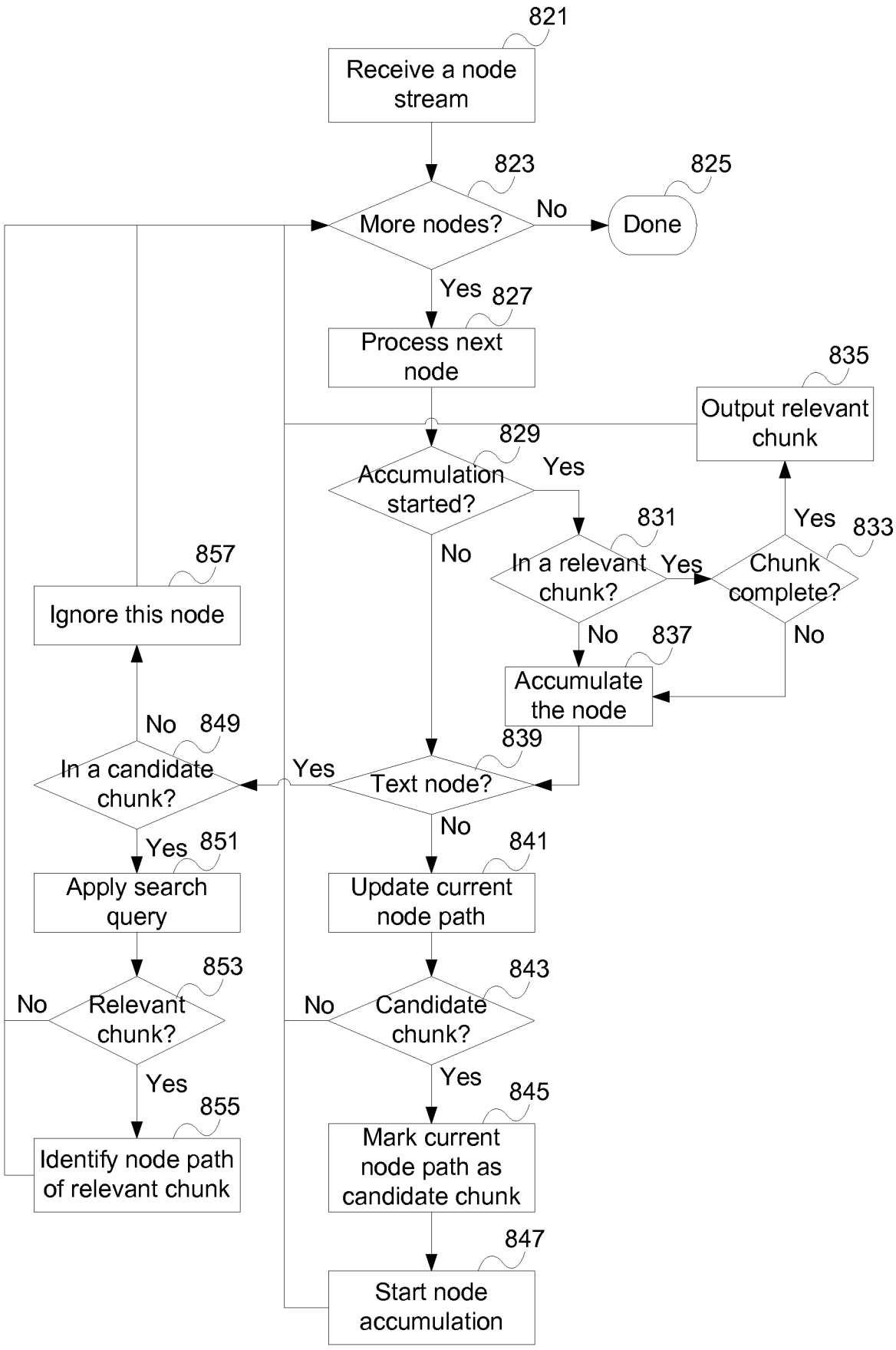

821
Receive a node stream
823
More nodes?
No
825
Done
Yes
827
Process next node
835
Output relevant chunk
829
Accumulation started?
Yes
831
In a relevant chunk?
Yes
Yes
833
Chunk complete?
No
857
Ignore this node
No
837
Accumulate the node
No
849
In a candidate chunk?
Yes
839
Text node?
No
Yes
851
Apply search query
841
Update current node path
853
Relevant chunk?
No
843
Candidate chunk?
No
Yes
Yes
855
Identify node path of relevant chunk
845
Mark current node path as candidate chunk
847
Start node accumulation

Figure 8B

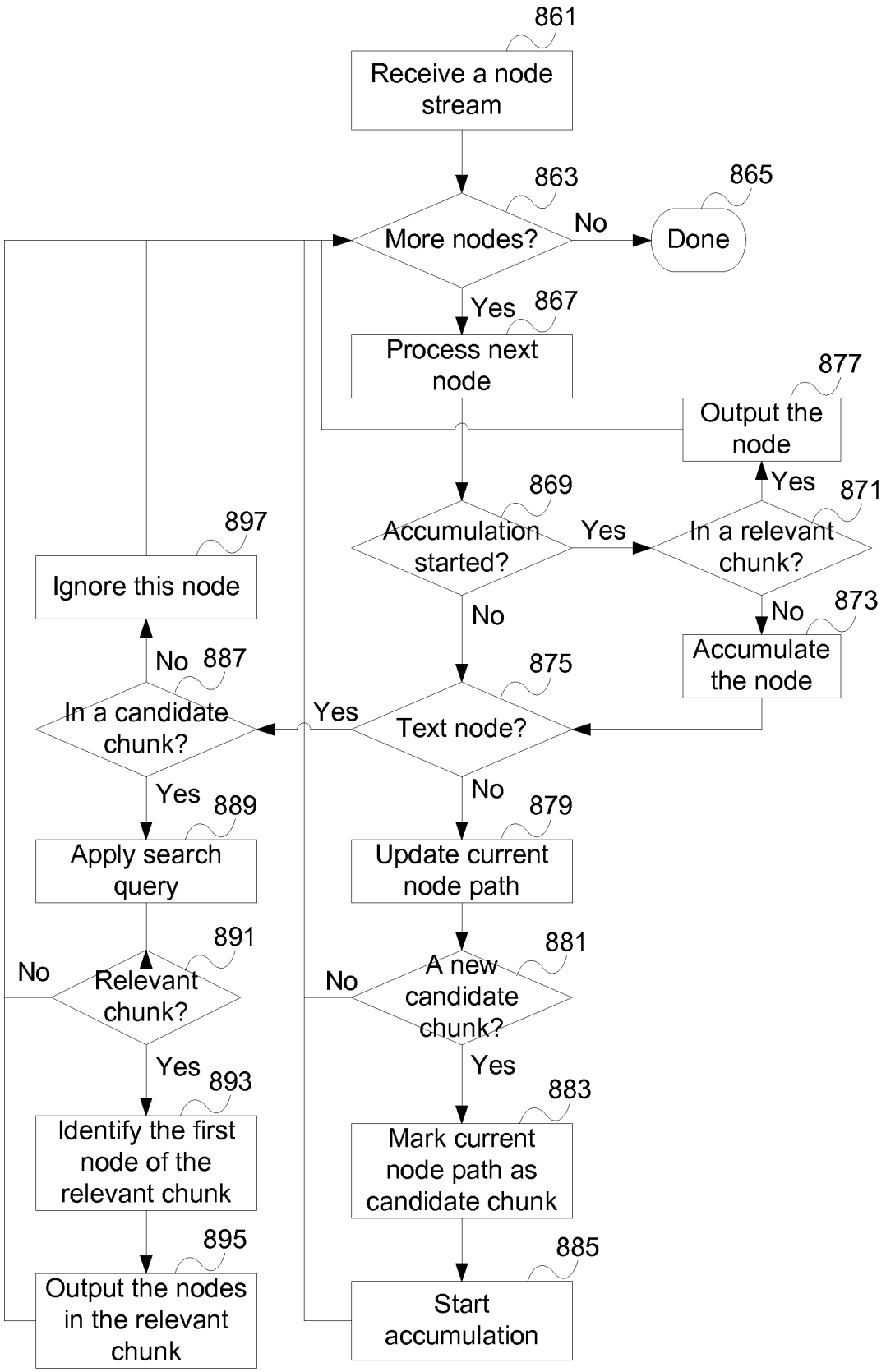

861
Receive a node stream
863
More nodes?
No
865
Done
Yes
867
Process next node
877
Output the node
Yes
871
In a relevant chunk?
No
873
Accumulate the node
869
Accumulation started?
Yes
No
875
Text node?
Yes
No
887
In a candidate chunk?
897
Ignore this node
Yes
889
Apply search query
891
Relevant chunk?
No
Yes
893
Identify the first node of the relevant chunk
895
Output the nodes in the relevant chunk
879
Update current node path
881
A new candidate chunk?
No
Yes
883
Mark current node path as candidate chunk
885
Start accumulation

Figure 8C

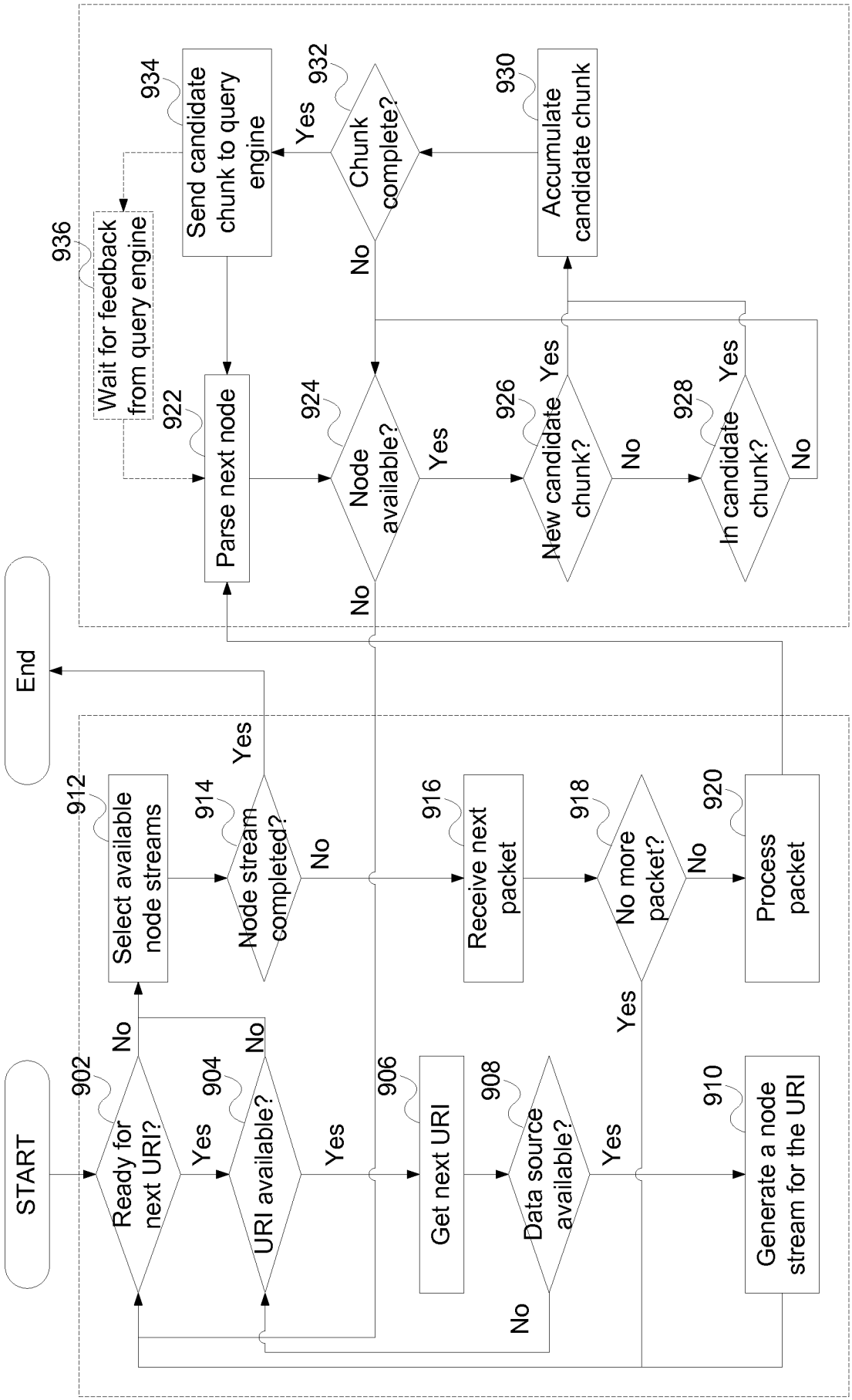
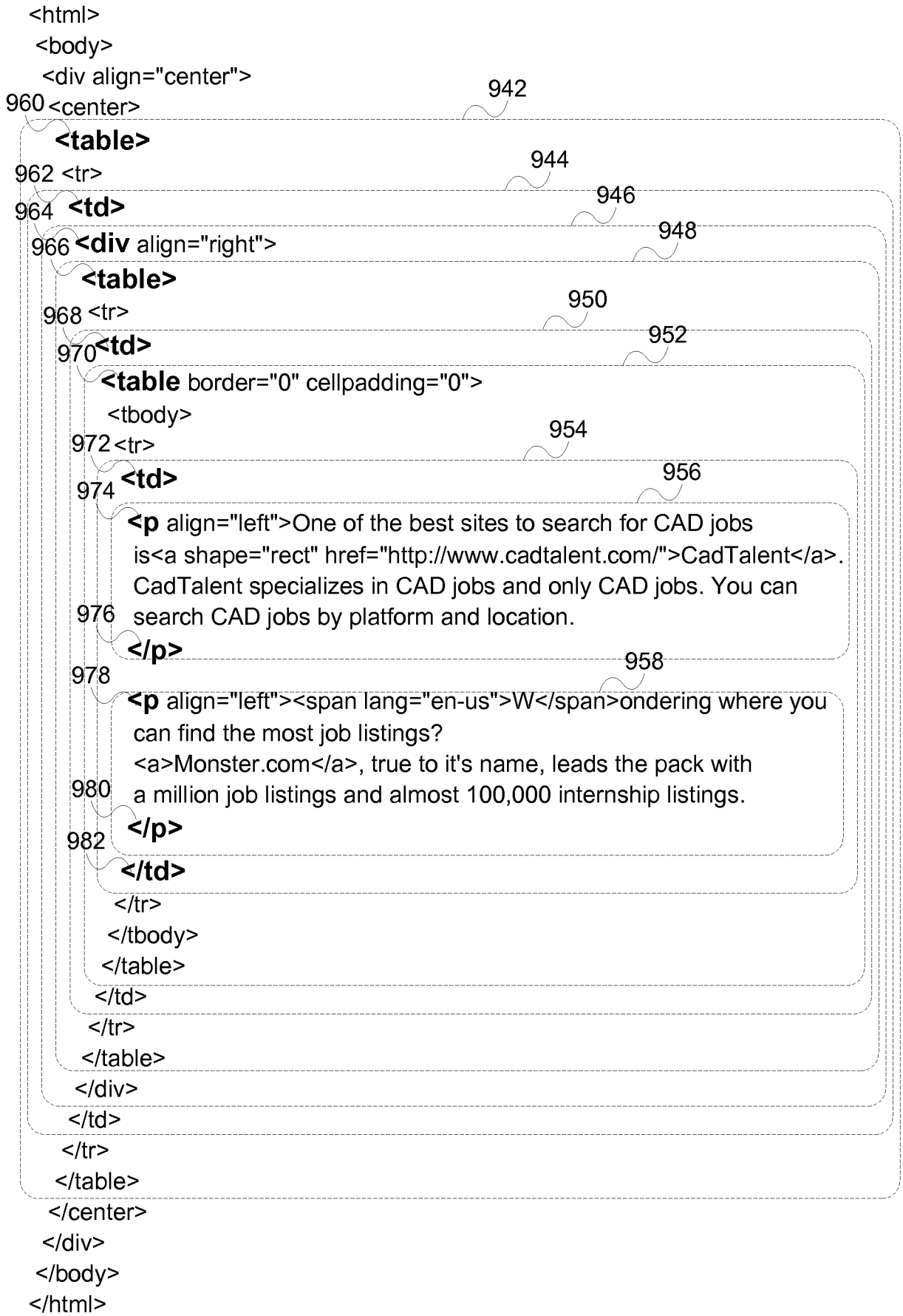

START
902
Ready for next URI?
No
Yes
904
URI available?
No
Yes
906
Get next URI
908
Data source available?
No
Yes
910
Generate a node stream for the URI
912
Select available node streams
914
Node stream completed?
Yes
No
916
Receive next packet
918
No more packet?
Yes
No
920
Process packet
End
DOCUMENT STREAMING
922
Parse next node
924
Node available?
No
Yes
926
New candidate chunk?
Yes
No
928
In candidate chunk?
Yes
No
930
Accumulate candidate chunk
932
Chunk complete?
No
Yes
934
Send candidate chunk to query engine
936
Wait for feedback from query engine
NODE PROCESSING

Figure 9A


<html>
<body>

960 
<table>
962 <tr>
964 <td>
966 
<table>
968 <tr>
970 <td>
<table border="0" cellpadding="0">
<tbody>
972 <tr>
974 <td>
One of the best sites to search for CAD jobs is<a shape="rect" href="http://www.cadtalent.com/">CadTalent</a>. CadTalent specializes in CAD jobs and only CAD jobs. You can
976 search CAD jobs by platform and location.

978
Wondering where you can find the most job listings?
<a>Monster.com</a>, true to it's name, leads the pack with
980 a million job listings and almost 100,000 internship listings.

982
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</table>

</td>
</tr>
</table>

</body>
</html>
942
944
946
948
950
952
954
956
958

Figure 9B

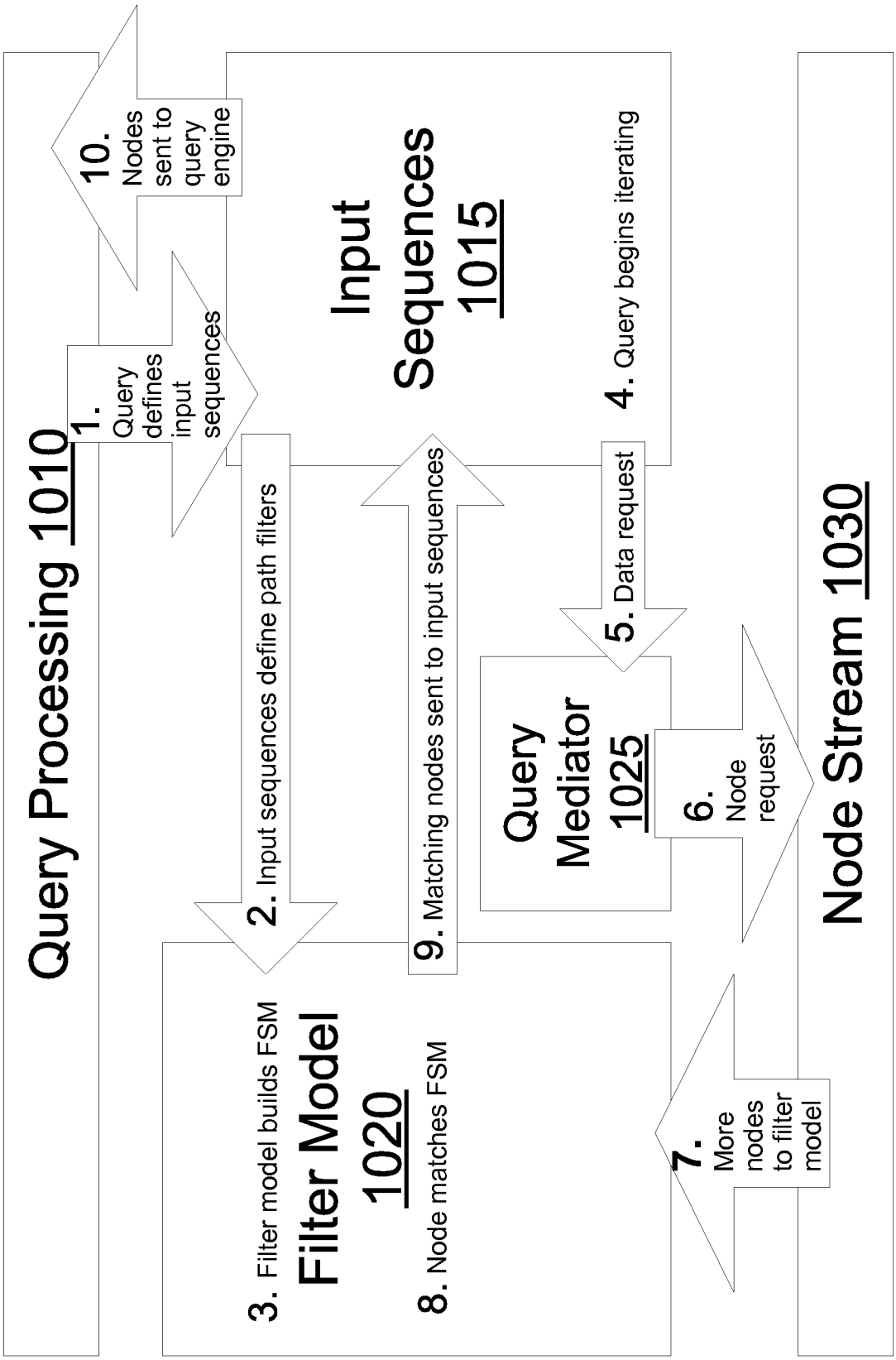

Query Processing 1010
1. Query defines input sequences
10. Nodes sent to query engine
Input Sequences 1015
4. Query begins iterating
5. Data request
Query Mediator 1025
6. Node request
Node Stream 1030
7. More nodes to filter model
2. Input sequences define path filters
3. Filter model builds FSM
Filter Model 1020
8. Node matches FSM
9. Matching nodes sent to input sequences

Figure 10A

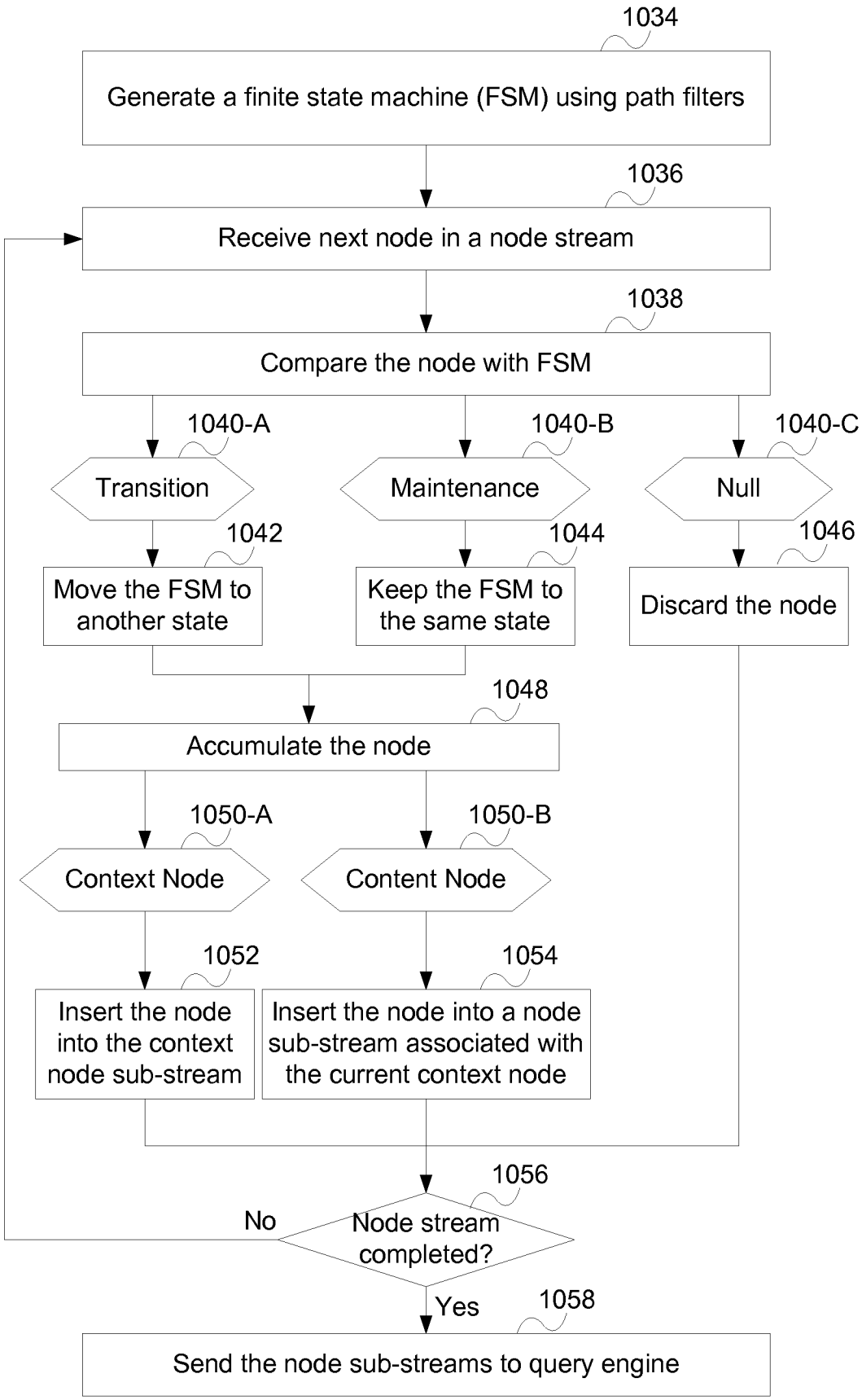

1034
Generate a finite state machine (FSM) using path filters
1036
Receive next node in a node stream
1038
Compare the node with FSM
1040-A
Transition
1040-B
Maintenance
1040-C
Null
1042
Move the FSM to another state
1044
Keep the FSM to the same state
1046
Discard the node
1048
Accumulate the node
1050-A
Context Node
1050-B
Content Node
1052
Insert the node into the context node sub-stream
1054
Insert the node into a node sub-stream associated with the current context node
1056
Node stream completed?
No
Yes
1058
Send the node sub-streams to query engine

Figure 10B


1100
<bib>
1102
<book year="1994">
<title>TCP/IP Illustrated</title>
<author><last>Stevens</last><first>W.</first></author>
<publisher>Addison-Wesley</publisher>
<price> 65.95</price>
</book>
1104
<book year="1992">
<title>Advanced Programming in the Unix environment</title>
<author><last>Stevens</last><first>W.</first></author>
<publisher>Addison-Wesley</publisher>
<price>65.95</price>
</book>
1106
<book year="2000">
<title>Data on the Web</title>
<author><last>Abiteboul</last><first>Serge</first></author>
<author><last>Buneman</last><first>Peter</first></author>
<author><last>Suciu</last><first>Dan</first></author>
<publisher>Morgan Kaufmann Publishers</publisher>
<price>39.95</price>
</book>
1108
<book year="1999">
<title>The Economics of Technology and Content for Digital TV</title>
<editor>
<last>Gerbarg</last><first>Darcy</first>
<affiliation>CITI</affiliation>
</editor>
<publisher>Kluwer Academic Publishers</publisher>
<price>129.95</price>
</book>
</bib>

Figure 11A

1110

```
<bib>
{
        for $b in doc('bib.xml')//book
        where $b/publisher = "Addison-Wesley"
        and     $b/@year > 1991
        return
                <book year="{ $b/@year }">
                { $b/title }
                </book>
}
</bib>
```

Figure 11B

1115

| Input Sequence | Path Filter | Mode | Parent |
|---|---|---|---|
| Node Sub-Stream (0) | bib/book | Context | N/A |
| Node Sub-Stream (1) | bib/book/publisher | Content | Node Sub-Stream (0) |
| Node Sub-Stream (2) | bib/book/@year | Context | Node Sub-Stream (0) |
| Node Sub-Stream (3) | bib/book/@year | Context | Node Sub-Stream (0) |
| Node Sub-Stream (4) | bib/book/title | All | Node Sub-Stream (0) |

Figure 11C

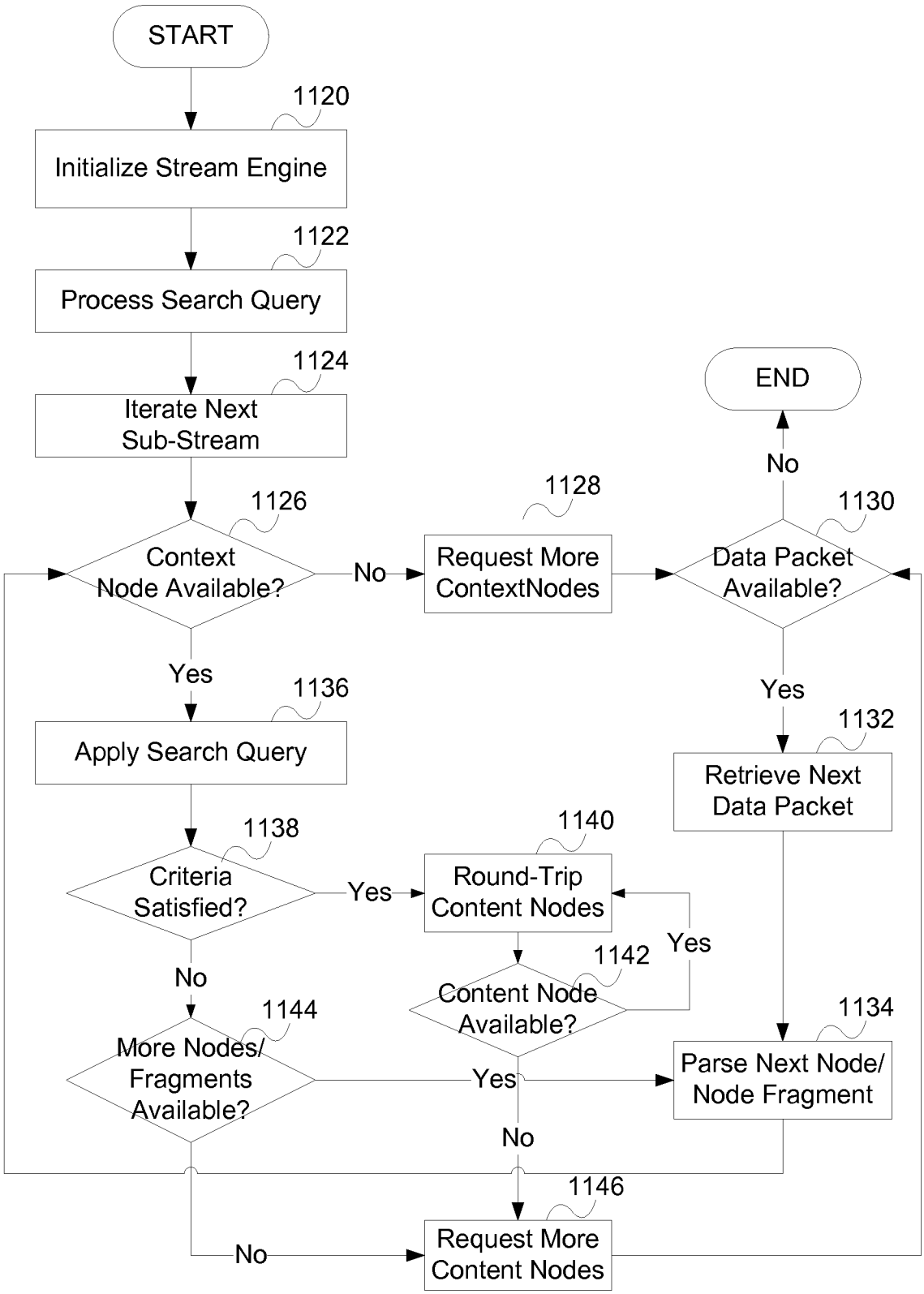

START
1120
Initialize Stream Engine
1122
Process Search Query
1124
Iterate Next Sub-Stream
1126
Context Node Available?
No
1128
Request More ContextNodes
1130
Data Packet Available?
No
END
Yes
1132
Retrieve Next Data Packet
Yes
1136
Apply Search Query
1138
Criteria Satisfied?
Yes
1140
Round-Trip Content Nodes
Yes
1142
Content Node Available?
No
1144
More Nodes/ Fragments Available?
Yes
1134
Parse Next Node/ Node Fragment
No
1146
No
Request More Content Nodes

Figure 11D

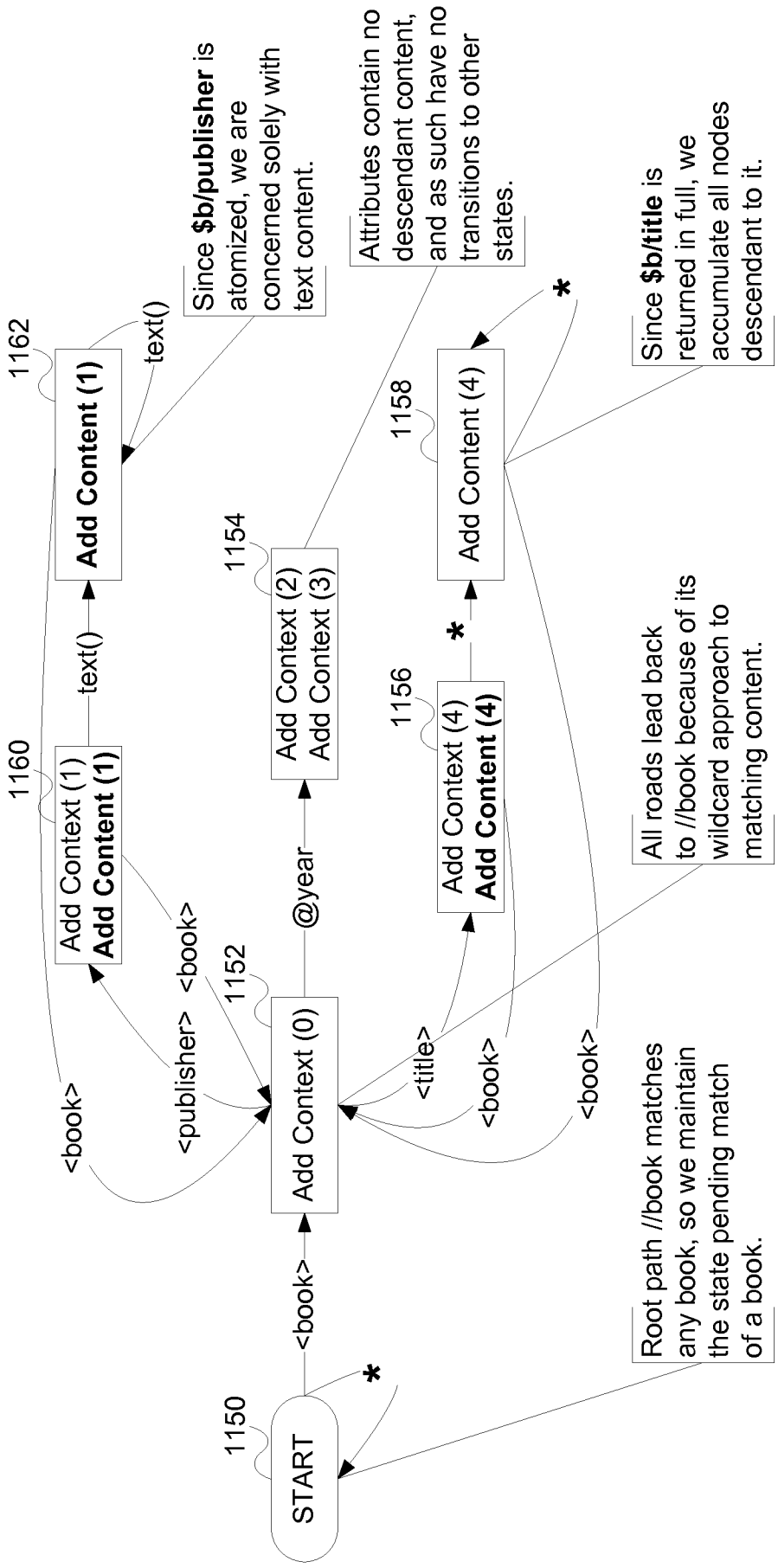

1150
START
*
<book>
1152
Add Context (0)
<book>
<publisher>
<book>
@year
<title>
<book>
<book>
1154
Add Context (2)
Add Context (3)
1156
Add Context (4)
Add Content (4)
*
1158
Add Content (4)
*
1160
Add Context (1)
Add Content (1)
text()
1162
Add Content (1)
text()
Root path //book matches any book, so we maintain the state pending match of a book.
All roads lead back to //book because of its wildcard approach to matching content.
Attributes contain no descendant content, and as such have no transitions to other states.
Since $b/publisher is atomized, we are concerned solely with text content.
Since $b/title is returned in full, we accumulate all nodes descendant to it.

Figure 11E

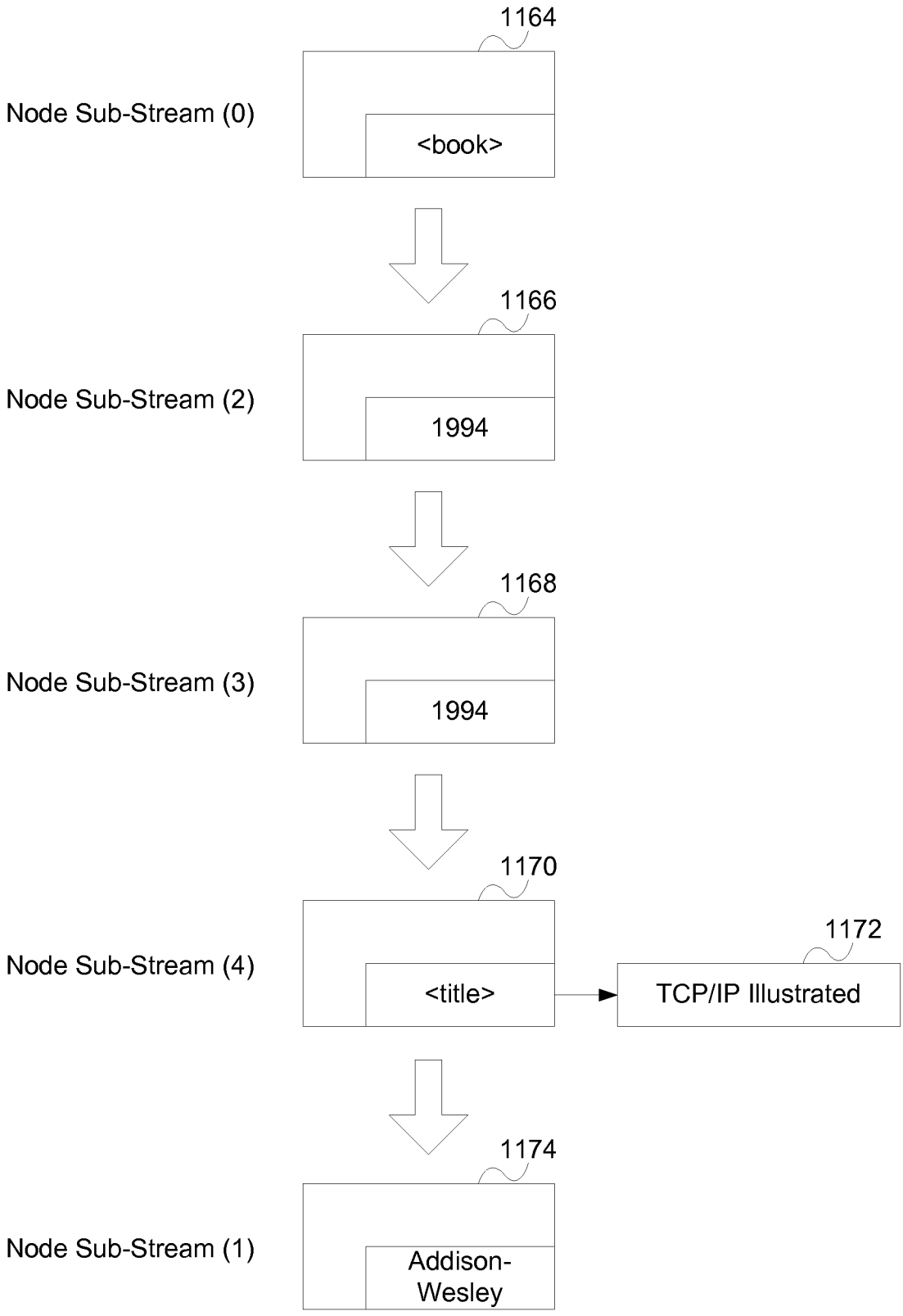

1164
Node Sub-Stream (0)
<book>
1166
Node Sub-Stream (2)
1994
1168
Node Sub-Stream (3)
1994
1170
1172
Node Sub-Stream (4)
<title>
TCP/IP Illustrated
1174
Node Sub-Stream (1)
Addison-
Wesley

Figure 11F

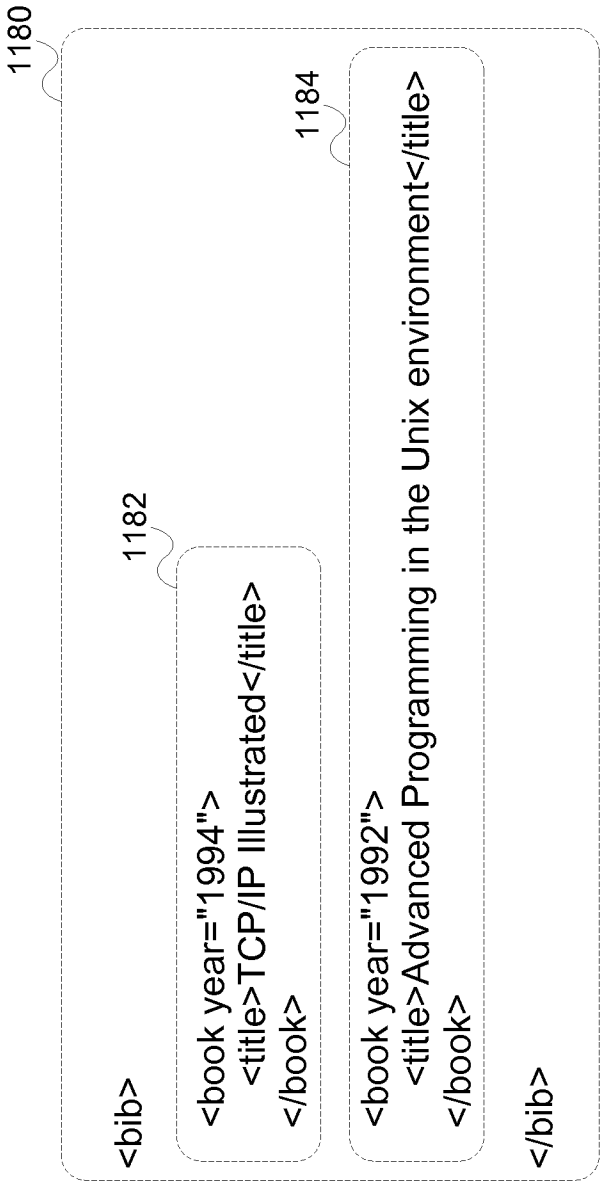

1180
1182
1184
<bib>
<book year="1994">
<title>TCP/IP Illustrated</title>
</book>
<book year="1992">
<title>Advanced Programming in the Unix environment</title>
</book>
</bib>

Figure 11G

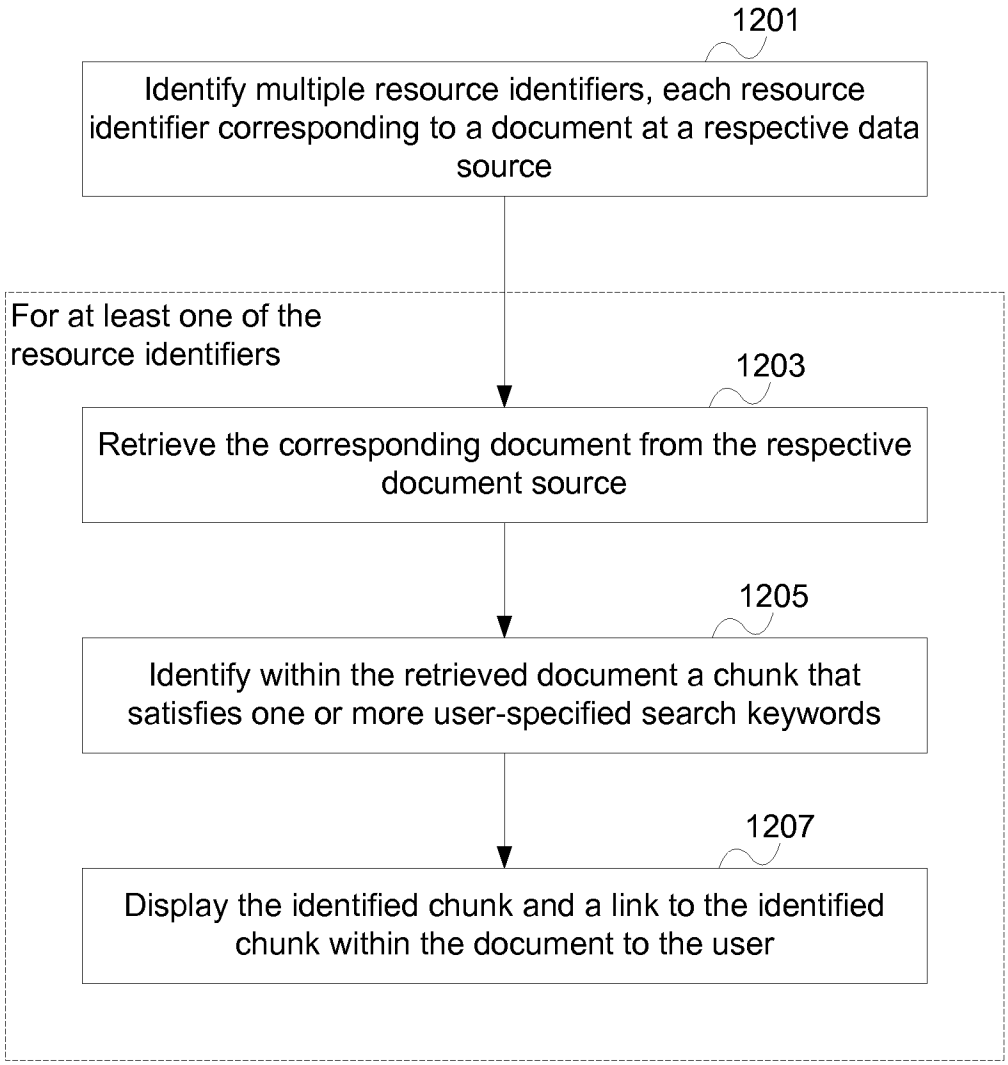

1201
Identify multiple resource identifiers, each resource identifier corresponding to a document at a respective data source
For at least one of the resource identifiers
1203
Retrieve the corresponding document from the respective document source
1205
Identify within the retrieved document a chunk that satisfies one or more user-specified search keywords
1207
Display the identified chunk and a link to the identified chunk within the document to the user

Figure 12A

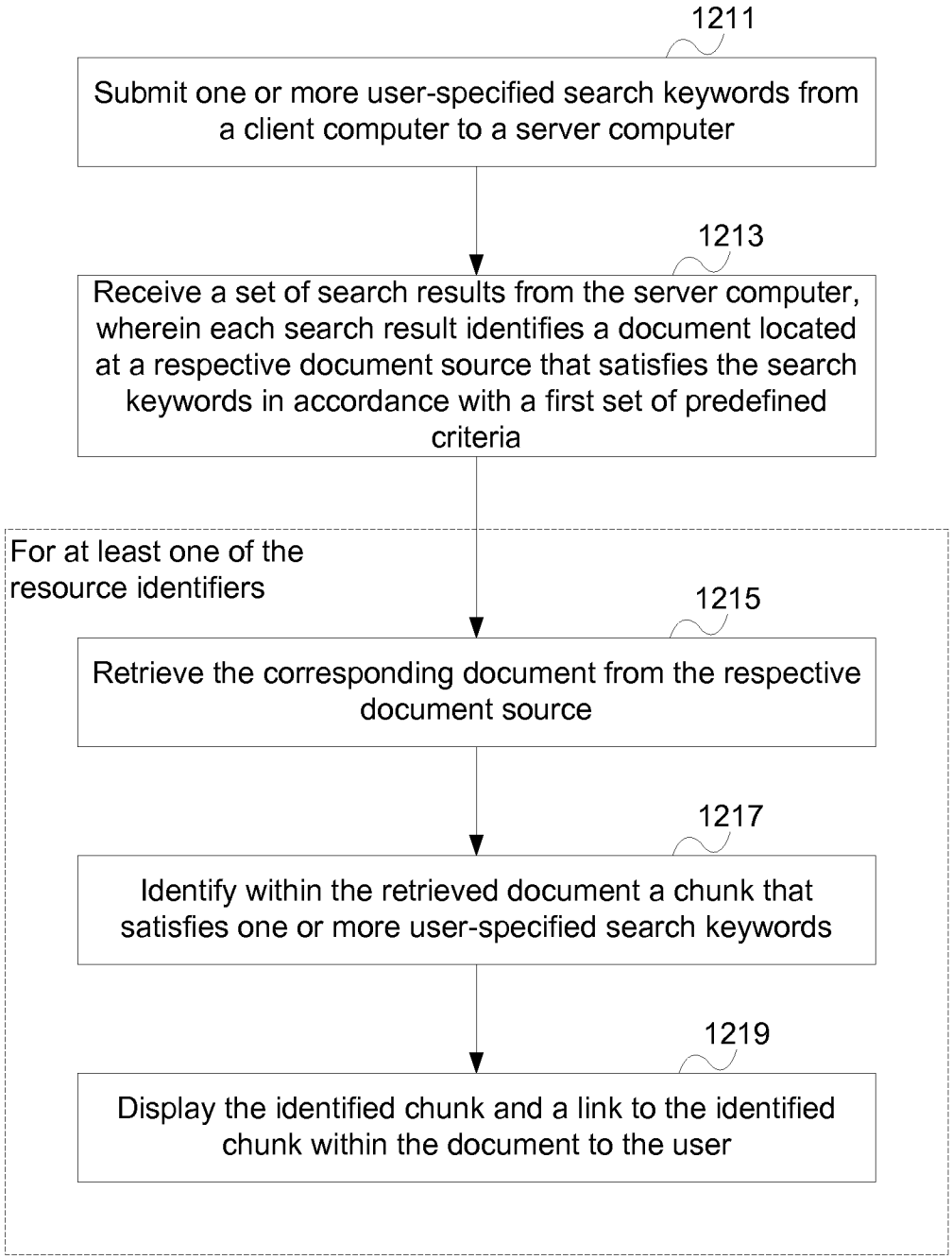

1211
Submit one or more user-specified search keywords from a client computer to a server computer
1213
Receive a set of search results from the server computer, wherein each search result identifies a document located at a respective document source that satisfies the search keywords in accordance with a first set of predefined criteria
For at least one of the resource identifiers
1215
Retrieve the corresponding document from the respective document source
1217
Identify within the retrieved document a chunk that satisfies one or more user-specified search keywords
1219
Display the identified chunk and a link to the identified chunk within the document to the user

Figure 12B

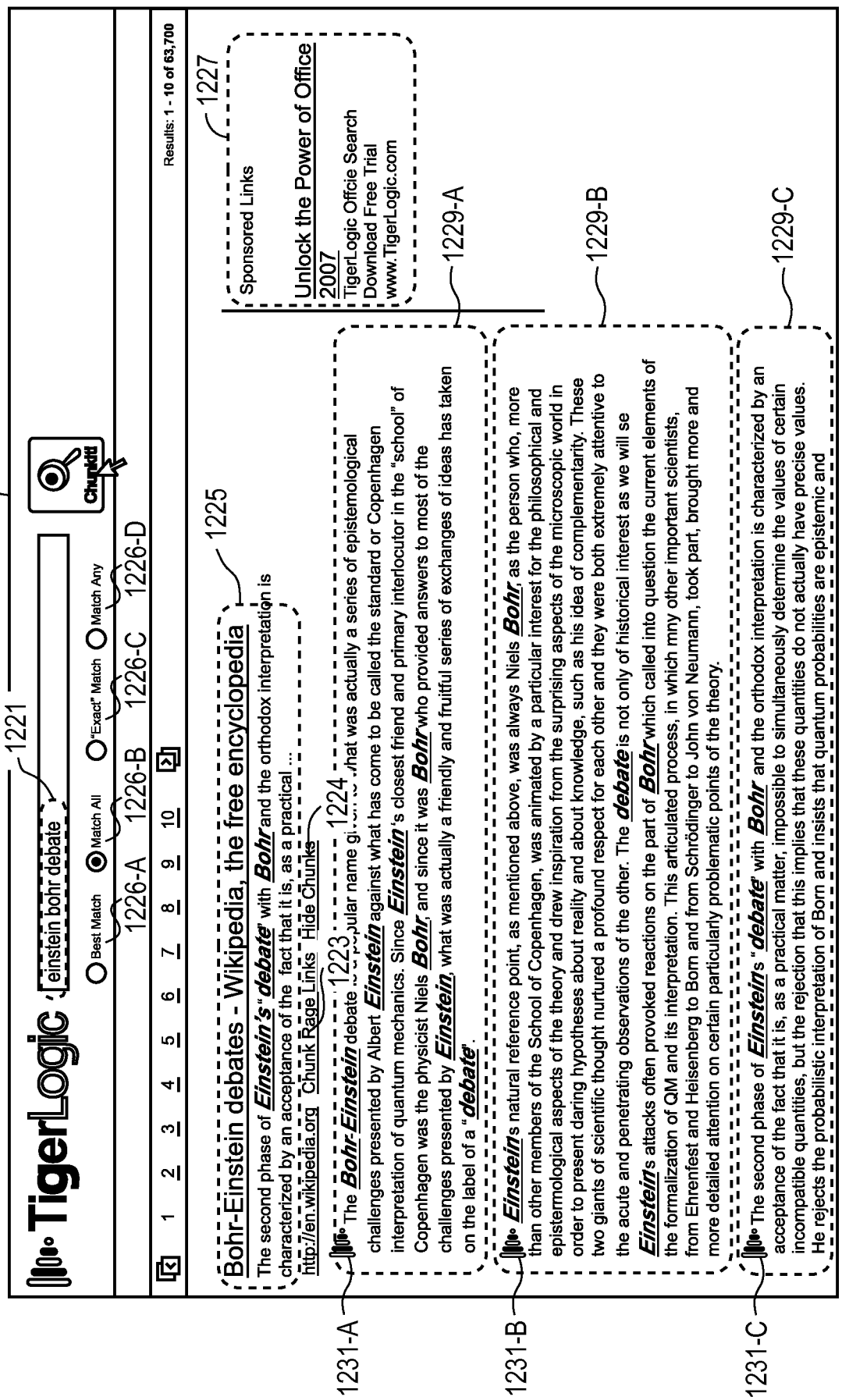

1220
TigerLogic
1221
einstein bohr debate
Best Match
Match All
"Exact" Match
Match Any
ChunkIt!
1226-A
1226-B
1226-C
1226-D
1 2 3 4 5 6 7 8 9 10
Results: 1 - 10 of 63,700
1225
Bohr-Einstein debates - Wikipedia, the free encyclopedia
The second phase of Einstein's "debate" with Bohr and the orthodox interpretation is characterized by an acceptance of the fact that it is, as a practical...
http://en.wikipedia.org Chunk Rage Links Hide Chunks
1223
1224
1227
Sponsored Links
Unlock the Power of Office 2007
TigerLogic Offcie Search
Download Free Trial
www.TigerLogic.com
1229-A
1229-B
1229-C
1231-A
The Bohr-Einstein debate ... popular name gi... that was actually a series of epistemological challenges presented by Albert Einstein against what has come to be called the standard or Copenhagen interpretation of quantum mechanics. Since Einstein's closest friend and primary interlocutor in the "school" of Copenhagen was the physicist Niels Bohr, and since it was Bohr who provided answers to most of the challenges presented by Einstein, what was actually a friendly and fruitful series of exchanges of ideas has taken on the label of a "debate".
1231-B
Einstein's natural reference point, as mentioned above, was always Niels Bohr, as the person who, more than other members of the School of Copenhagen, was animated by a particular interest for the philosophical and epistemological aspects of the theory and drew inspiration from the surprising aspects of the microscopic world in order to present daring hypotheses about reality and about knowledge, such as his idea of complementarity. These two giants of scientific thought nurtured a profound respect for each other and they were both extremely attentive to the acute and penetrating observations of the other. The debate is not only of historical interest as we will se Einstein's attacks often provoked reactions on the part of Bohr which called into question the current elements of the formalization of QM and its interpretation. This articulated process, in which mny other important scientists, from Ehrenfest and Heisenberg to Born and from Schrödinger to John von Neumann, took part, brought more and more detailed attention on certain particularly problematic points of the theory.
1231-C
The second phase of Einstein's "debate" with Bohr and the orthodox interpretation is characterized by an acceptance of the fact that it is, as a practical matter, impossible to simultaneously determine the values of certain incompatible quantities, but the rejection that this implies that these quantities do not actually have precise values. He rejects the probabilistic interpretation of Born and insists that quantum probabilities are epistemic and

Figure 12C

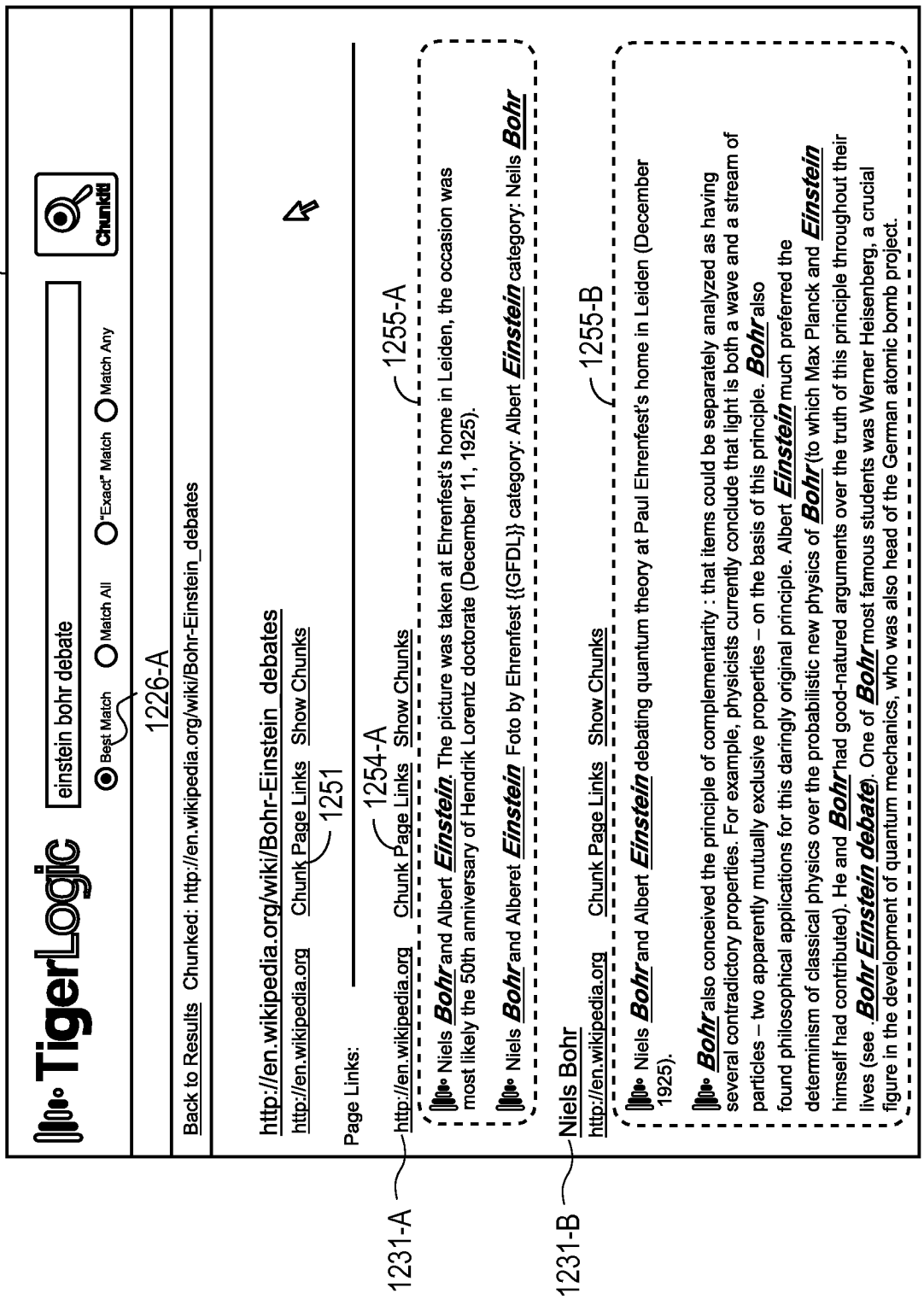

1220
TigerLogic
einstein bohr debate
Best Match
Match All
"Exact" Match
Match Any
Chunkit
1226-A
Back to Results Chunked: http://en.wikipedia.org/wiki/Bohr-Einstein_debates
http://en.wikipedia.org/wiki/Bohr-Einstein_debates
http://en.wikipedia.org Chunk Page Links Show Chunks
1251
Page Links:
1254-A
1231-A
http://en.wikipedia.org Chunk Page Links Show Chunks
1255-A
Niels Bohr and Albert Einstein. The picture was taken at Ehrenfest's home in Leiden, the occasion was most likely the 50th anniversary of Hendrik Lorentz doctorate (December 11, 1925).
Niels Bohr and Alberet Einstein Foto by Ehrenfest {{GFDL}} category: Albert Einstein category: Neils Bohr
1231-B
Niels Bohr
http://en.wikipedia.org Chunk Page Links Show Chunks
1255-B
Niels Bohr and Albert Einstein debating quantum theory at Paul Ehrenfest's home in Leiden (December 1925).
Bohr also conceived the principle of complementarity : that items could be separately analyzed as having several contradictory properties. For example, physicists currently conclude that light is both a wave and a stream of particles – two apparently mutually exclusive properties – on the basis of this principle. Bohr also found philosophical applications for this daringly original principle. Albert Einstein much preferred the determinism of classical physics over the probabilistic new physics of Bohr (to which Max Planck and Einstein himself had contributed). He and Bohr had good-natured arguments over the truth of this principle throughout their lives (see .Bohr Einstein debate). One of Bohr most famous students was Werner Heisenberg, a crucial figure in the development of quantum mechanics, who was also head of the German atomic bomb project.

Figure 12D

TigerLogic
einstein bohr debate
Best Match
Match All
"Exact" Match
Match Any
ChunkIt!
1 2 3 4 5 6 7 8 9 10
Bohr-Einstein debates - Wikipedia, the free encyclopedia
The second phase of Einstein's "debate" with Bohr and the orthodox interpretation is characterized by an acceptance of the fact that it is, as a practical ...
http://en.wikipedia.org Chunk Rage Links Hide Chunks
Bohr v Einstein
1257-A
The debate between Bohr and Einstein over the interpretation of quantum theory .....
The debate between Einstein and Bohr was conducted with the two talking
http://homepaces.paradise.net.nz Chunk Rage Links Hide Chunks
1257-B
Einstein/Bohr Debate and Quantum Computing
[PDF]
http://www.nanohub.org Chunk Rage Links Hide Chunks
Location skipped: content can not be chunked.
nanoHUB - Einstein?Bohr Debate and Quantum Computing
nanoHUB.org: online simulation and more... Our mission: To be the place where experiment, theory and simulation meet and move nanoscience to nanotechnology ..
http://www.nanohub.org Chunk Rage Links Hide Chunks
No relevant content found at this location. Re-Chunk with match any keyword.
Einstein Exhibit -- The Quantum and The Cosmo
At the Solvay Conferences of 1927 and 1930 the debate between Bohr and Einstein went on day and night, neither man conceding defeat. Werner Heisenberg ...
http://www.aip.org Chunk Rage Links Hide Chunks

Figure 12E

TigerLogic
bohr-einstein debates
Best Match
Match All
"Exact" Match
Match Any
ChunkIt
1226-C
1 2 3 4 5 6 7 8 9 10
Bohr-Einstein debates - EPR paradox
A selection of articles related to Bohr-Einstein debates - EPR paradox
http://www.experiencefestival.com Chunk Rage Links Hide Chunks
1233-A
A Wisdom Archive on Bohr-Einstein debates - EPR paradox
A selection of articles related to Bohr-Einstein debates - EPR paradox
More material related to Bohr-einstein Debates can be found here:
Main Page. for. Bohr-einstein Debates
Index of Articles. related to. Bohr-einstein Debates
Index of Articles. related to. Bohr-Einstein debates - E...
Bohr-Einstein debates, Bohr-Einstein debates - Diffraction the single slit experiment, Bohr-Einstein debates - EPR paradox, Bohr-Einstein debates - Interference the double slit experiment, Bohr-Einstein debates - Photon in a box, Afshar's experiment, Complementarity, Copenhagen interpretation, Quantum eraser, Schrödinger's cat, Uncertainty principle, Wheeler's delayed choice experiment
ARTICLES RELATED TO Bohr-Einstein debates - EPR paracox
1233-B
The Bohr-Einstein debates on foundational aspects on quantum mechanics happened during the Solvay conferences. They consisted of analyses of thought experiments. Put simply, they were an attempt by Einstein to explain away the aspects of Bohr's interpretation of Quantum Mechanics that he disliked. Bohr attempted (and, most scholars agree, largely succeeded) to rebuild these challenges. The Bohr-Einstein debates remain among the most important in the history of the philosophy of physics, and are certainly ...
Bohr-Einstein debates - Diffraction the single slit experiment

Figure 12F

TigerLogic
bohr-einstein debates
Best Match
Match All
"Exact" Match
Match Any
Chunks
1 2 3 4 5 6 7 8 9 10
Bohr-Einstein debates - EPR paradox
A selection of articles related to Bohr-Einstein debates - EPR paradox
http://www.experiencefestival.com Chunk Rage Links Hide Chunks
1233-B
The Bohr-Einstein debates on foundational aspects on quantum mechanics happened during the Solvay conferences. They consisted of analyses of thought experiments. Put simply, they were an attempt by Einstein to explain away the aspects of Bohr's interpretation of Quantum Mechanics that he disliked. Bohr attempted (and, most scholars agree, largely succeeded) to rebuild these challenges. The Bohr-Einstein debates remain among the most important in the history of the philosophy of physics, and are certainly ...
A Wisdom Archive on Bohr-Einstein debates - EPR paradox
1233-A
A selection of articles related to Bohr-Einstein debates - EPR paradox
More material related to Bohr-einstein Debates can be found here:
Main Page. for. Bohr-einstein Debates
Index of Articles. related to. Bohr-einstein Debates
Index of Articles. related to. Bohr-Einstein debates - E...
Bohr-Einstein debates, Bohr-Einstein debates - Diffraction the single slit experiment, Bohr-Einstein debates - EPR paradox, Bohr-Einstein debates - Interference the double slit experiment, Bohr-Einstein debates - Photon in a box, Afshar's experiment, Complementarity, Copenhagen interpretation, Quantum eraser, Schrödinger's cat, Uncertainty principle, Wheeler's delayed choice experiment
ARTICLES RELATED TO Bohr-Einstein debates - EPR paracox
Bohr-Einstein debates - Diffraction the single slit experiment

Figure 12G

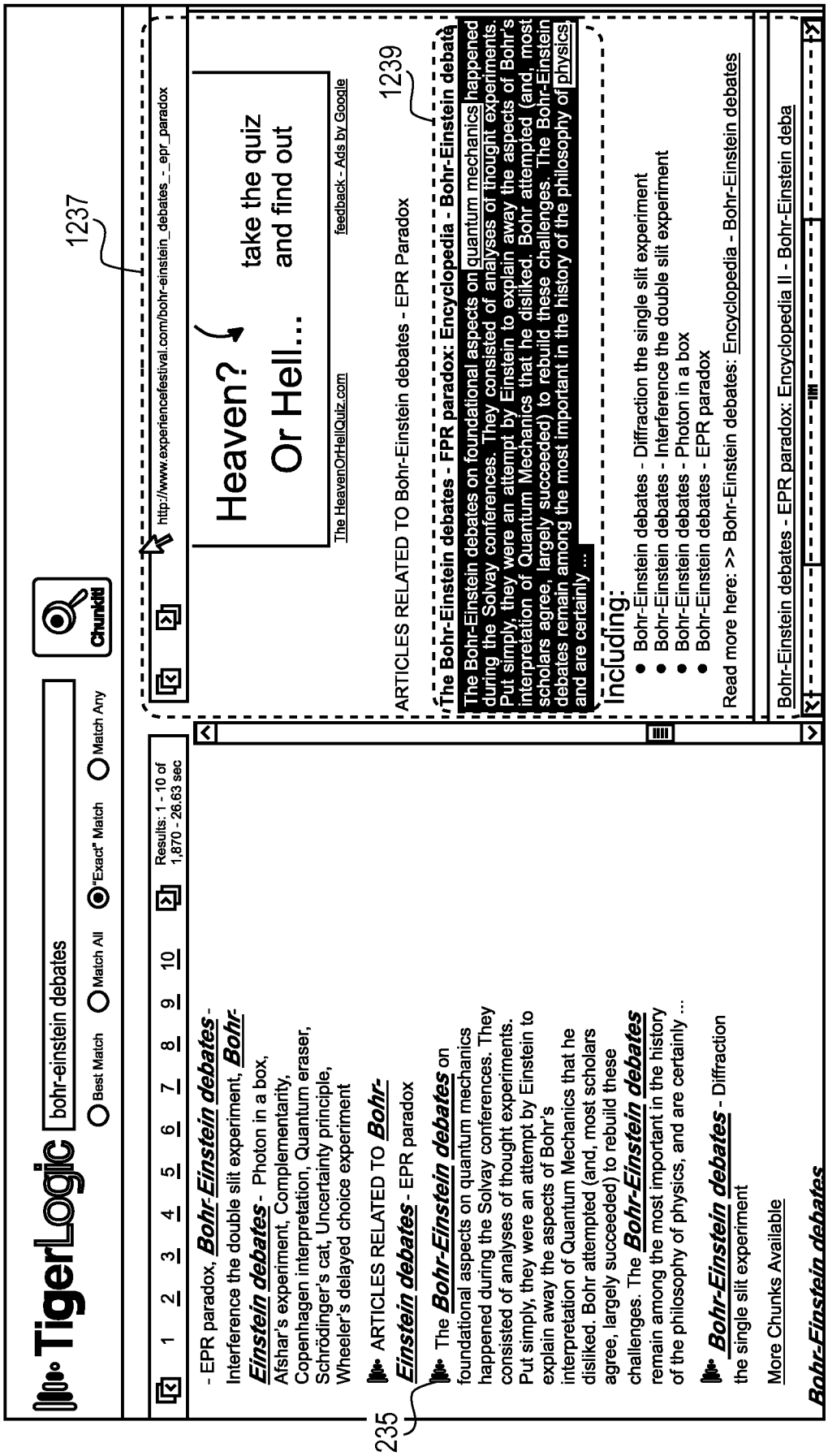

TigerLogic
bohr-einstein debates
Best Match
Match All
"Exact" Match
Match Any
Chunkit!
1237
http://www.experiencefestival.com/bohr-einstein_debates_-_epr_paradox
1 2 3 4 5 6 7 8 9 10
Results: 1 - 10 of 1,870 - 26.63 sec
Heaven?
Or Hell...
take the quiz and find out
The HeavenOrHellQuiz.com
feedback - Ads by Google
- EPR paradox, Bohr-Einstein debates - Interference the double slit experiment, Bohr-Einstein debates - Photon in a box, Afshar's experiment, Complementarity, Copenhagen interpretation, Quantum eraser, Schrödinger's cat, Uncertainty principle, Wheeler's delayed choice experiment
ARTICLES RELATED TO Bohr-Einstein debates - EPR Paradox
1239
The Bohr-Einstein debates - FPR paradox: Encyclopedia - Bohr-Einstein debate
The Bohr-Einstein debates on foundational aspects on quantum mechanics happened during the Solvay conferences. They consisted of analyses of thought experiments. Put simply, they were an attempt by Einstein to explain away the aspects of Bohr's interpretation of Quantum Mechanics that he disliked. Bohr attempted (and, most scholars agree, largely succeeded) to rebuild these challenges. The Bohr-Einstein debates remain among the most important in the history of the philosophy of physics and are certainly ...
Including:
Bohr-Einstein debates - Diffraction the single slit experiment
Bohr-Einstein debates - Interference the double slit experiment
Bohr-Einstein debates - Photon in a box
Bohr-Einstein debates - EPR paradox
Read more here: >> Bohr-Einstein debates: Encyclopedia - Bohr-Einstein debates
1235
ARTICLES RELATED TO Bohr-Einstein debates - EPR paradox
The Bohr-Einstein debates on quantum mechanics happened during the Solvay conferences. They consisted of analyses of thought experiments. Put simply, they were an attempt by Einstein to explain away the aspects of Bohr's interpretation of Quantum Mechanics that he disliked. Bohr attempted (and, most scholars agree, largely succeeded) to rebuild these challenges. The Bohr-Einstein debates remain among the most important in the history of the philosophy of physics, and are certainly ...
Bohr-Einstein debates - Diffraction the single slit experiment
More Chunks Available
Bohr-Einstein debates - EPR paradox: Encyclopedia II - Bohr-Einstein deba
Bohr-Einstein debates

Figure 12H

TigerLogic
einstein big bang
Best Match
Match All
"Exact" Match
Match Any
ChunkIt!
1 2 3 4 5 6 7 8 9 10
Cosmology: Big Bang Theory
The Big bang Model is a broadly accepted theory for the origin and evolution of ... Alber .Einstein at the chalkboard. The first key idea dates to 1916 when ...
http://map.gsfc.nasa.gov Chunk Page Links Hide Chunks
After the introduction of General Relativity a number of scientists, including Einstein, tried to apply the new gravitational dynamics to the universe as a whole. At the time this required as assumption about how the matter in the universe was distributed. The simplest assumption to make is that if you viewed the contents of the universe with sufficiently poor vision, it would appear roughly the same everywhere and in every direction. That is, the matter in the universe is homogenous and isotrophic when averaged over very large scales. This is called the Cosmological Principle. This assumption is being tested continuously as we actually observe the distribution of galaxies on ever larger scales. The accompanying picture shows how uniform the distribution of measured galaxies is over a 30° swath of sky. In addition the cosmic microwave background radiation, the remnant heat from the Big Bang, has a temperature which is highly uniform over the entire sky. This fact strongly supports the notion that the gas which emitted this radiation long ago was very uniformly distributed.
1241
Stephen King, The Big Bang, and God
Einstein considered this to be the greatest blunder of his scientific career ... The .big bang theory is significant evidence against Hinduism ...
http://www.eaderu.com Chunk Page Links Hide Chunks
1243
No relevant content found at this location. Re-Chunk with match any keyword.
The Big Bang Theory
So how does it all stack up? How much evidence do we actually have to support the Big Bang model of the universe? Einstein's Theory of Relativity. ...
http://www.theke.board.org.uk Chunk Page Links Hide Chunks

Figure 12I

TigerLogic
einstein big bang
Best Match
Match All
"Exact" Match
Match Any
Chunklt!
1226-D
1 2 3 4 5 6 7 8 9 10
Stephen King, The Big Bang, and God
Einstein considered this to be the greatest blunder of his scientific career ... The .big bang theory is significant evidence against Hinduism ...
http://www.leaderu.com Chunk Page Links Hide Chunks
1245
The idea that the universe had a specific time of origin has been philosophically resisted by some very distinguished scientists. We could begin with Arthur Eddington, who experimentally confirmed Einstein's general theory of relativity in 1919. He stated a dozen years later: "Philosophically, the notion of a beginning to the present order is repugnant to me and I should like to find a genuine loophole." He later said, "We must allow evolution an infinite amount of time to get started."
Albert Einstein's reaction to the consequences of his own general theory of relativity appear to acknowledge the threat of an encounter with God. Through the equations of general relativity, we can trace the origin of the universe backward in time to some sort of a beginning. However, before publishing his cosmological inferences. Einstein introduced a cosmological constant, a "fudge factor," to yield a static model for the universe. Einstein later considered this to be the greatest blunder of his scientific career.
Einstein ultimately gave grudging acceptance to what he called "the necessity for a beginning" and eventually to "the presence of a superior reasoning power." But he never did accept the reality of a personal God.
1247
In 1946, George Gamow, a Russian-born scientist, proposed that the primeval fireball, the big bang" was an intense concentration of pure energy. It was the source of all the matter that now exists in the universe. The theory predicts that all the galaxies in the universe should be rushing away from each other at high speeds as a result of that initial big bang. A dictionary definition of the hot big bang theory is "the entire physical universe, all the matter and energy and even the four dimensions of time and space, burst forth from a state of infinite or near infinite density, temperature, and pressure."
The 1965 observation of the microwave background radiation by Arno Penzias and Robert Wilson from the Bell Telephone Laboratories convinced most scientists of the validity of the Big Bang theory. Further observations reported

Figure 12J

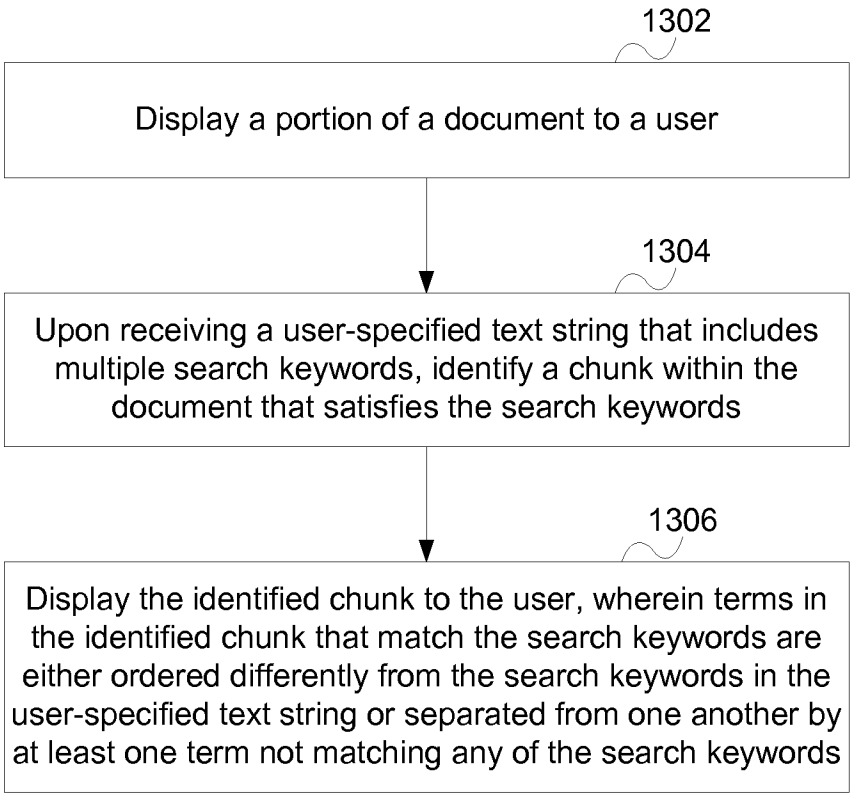

1302
Display a portion of a document to a user
1304
Upon receiving a user-specified text string that includes multiple search keywords, identify a chunk within the document that satisfies the search keywords
1306
Display the identified chunk to the user, wherein terms in the identified chunk that match the search keywords are either ordered differently from the search keywords in the user-specified text string or separated from one another by at least one term not matching any of the search keywords

Figure 13A

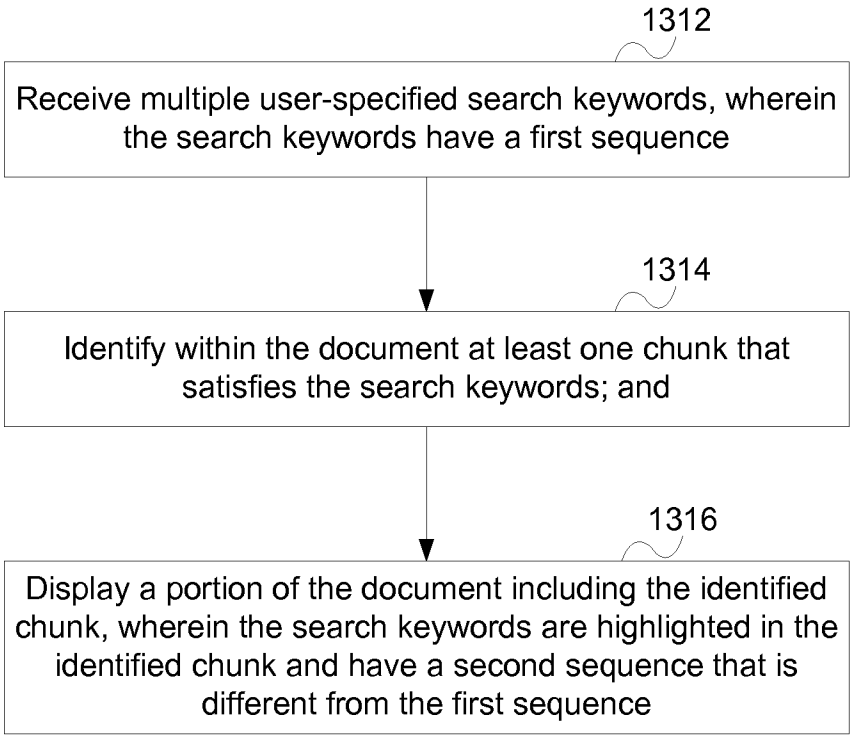

1312
Receive multiple user-specified search keywords, wherein the search keywords have a first sequence
1314
Identify within the document at least one chunk that satisfies the search keywords; and
1316
Display a portion of the document including the identified chunk, wherein the search keywords are highlighted in the identified chunk and have a second sequence that is different from the first sequence

Figure 13B

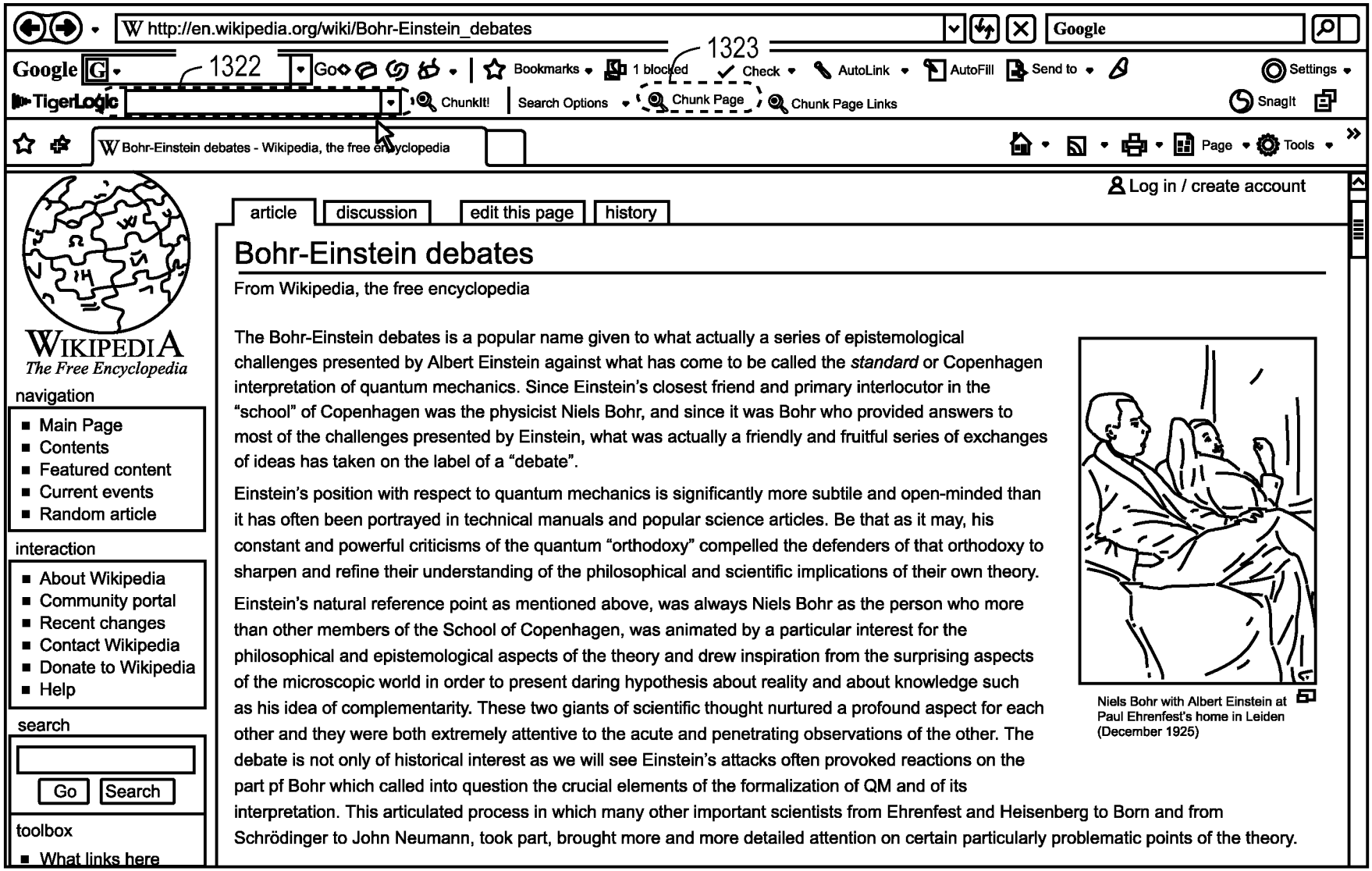
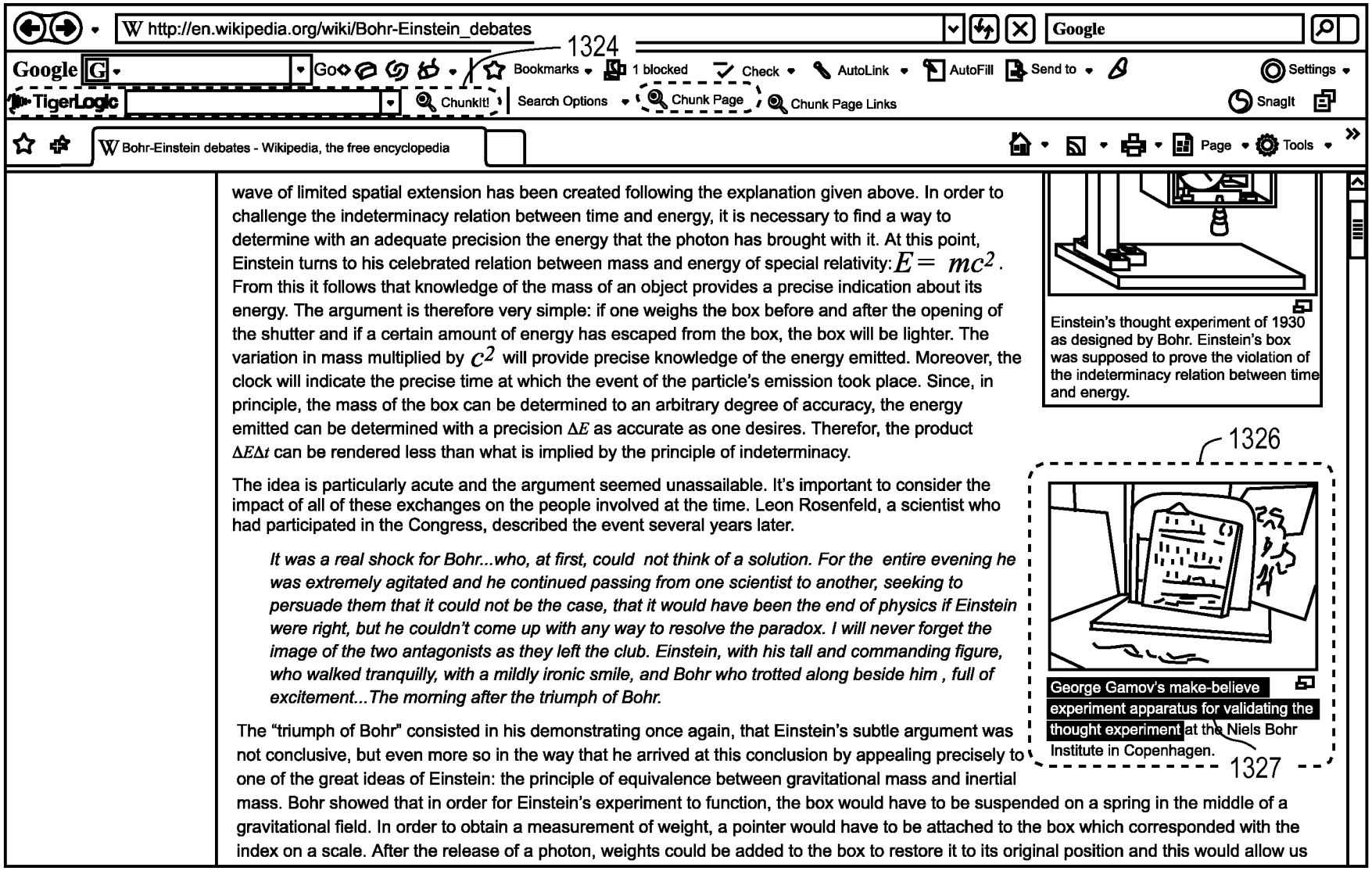

http://en.wikipedia.org/wiki/Bohr-Einstein_debates
Google
1322
1323
Go
Bookmarks
1 blocked
Check
AutoLink
AutoFill
Send to
Settings
TigerLogic
ChunkIt!
Search Options
Chunk Page
Chunk Page Links
SnagIt
Bohr-Einstein debates - Wikipedia, the free encyclopedia
Page
Tools
Log in / create account
article
discussion
edit this page
history
WIKIPEDIA
The Free Encyclopedia
navigation
Main Page
Contents
Featured content
Current events
Random article
interaction
About Wikipedia
Community portal
Recent changes
Contact Wikipedia
Donate to Wikipedia
Help
search
Go
Search
toolbox
What links here
Bohr-Einstein debates
From Wikipedia, the free encyclopedia
The Bohr-Einstein debates is a popular name given to what actually a series of epistemological challenges presented by Albert Einstein against what has come to be called the *standard* or Copenhagen interpretation of quantum mechanics. Since Einstein's closest friend and primary interlocutor in the "school" of Copenhagen was the physicist Niels Bohr, and since it was Bohr who provided answers to most of the challenges presented by Einstein, what was actually a friendly and fruitful series of exchanges of ideas has taken on the label of a "debate".
Einstein's position with respect to quantum mechanics is significantly more subtile and open-minded than it has often been portrayed in technical manuals and popular science articles. Be that as it may, his constant and powerful criticisms of the quantum "orthodoxy" compelled the defenders of that orthodoxy to sharpen and refine their understanding of the philosophical and scientific implications of their own theory.
Einstein's natural reference point as mentioned above, was always Niels Bohr as the person who more than other members of the School of Copenhagen, was animated by a particular interest for the philosophical and epistemological aspects of the theory and drew inspiration from the surprising aspects of the microscopic world in order to present daring hypothesis about reality and about knowledge such as his idea of complementarity. These two giants of scientific thought nurtured a profound aspect for each other and they were both extremely attentive to the acute and penetrating observations of the other. The debate is not only of historical interest as we will see Einstein's attacks often provoked reactions on the part pf Bohr which called into question the crucial elements of the formalization of QM and of its interpretation. This articulated process in which many other important scientists from Ehrenfest and Heisenberg to Born and from Schrödinger to John Neumann, took part, brought more and more detailed attention on certain particularly problematic points of the theory.
Niels Bohr with Albert Einstein at Paul Ehrenfest's home in Leiden (December 1925)

Figure 13C




wave of limited spatial extension has been created following the explanation given above. In order to challenge the indeterminacy relation between time and energy, it is necessary to find a way to determine with an adequate precision the energy that the photon has brought with it. At this point, Einstein turns to his celebrated relation between mass and energy of special relativity: $E = mc^2$. From this it follows that knowledge of the mass of an object provides a precise indication about its energy. The argument is therefore very simple: if one weighs the box before and after the opening of the shutter and if a certain amount of energy has escaped from the box, the box will be lighter. The variation in mass multiplied by $c^2$ will provide precise knowledge of the energy emitted. Moreover, the clock will indicate the precise time at which the event of the particle's emission took place. Since, in principle, the mass of the box can be determined to an arbitrary degree of accuracy, the energy emitted can be determined with a precision $\Delta E$ as accurate as one desires. Therefor, the product $\Delta E \Delta t$ can be rendered less than what is implied by the principle of indeterminacy.

Einstein's thought experiment of 1930 as designed by Bohr. Einstein's box was supposed to prove the violation of the indeterminacy relation between time and energy.

The idea is particularly acute and the argument seemed unassailable. It's important to consider the impact of all of these exchanges on the people involved at the time. Leon Rosenfeld, a scientist who had participated in the Congress, described the event several years later.

> *It was a real shock for Bohr...who, at first, could not think of a solution. For the entire evening he was extremely agitated and he continued passing from one scientist to another, seeking to persuade them that it could not be the case, that it would have been the end of physics if Einstein were right, but he couldn't come up with any way to resolve the paradox. I will never forget the image of the two antagonists as they left the club. Einstein, with his tall and commanding figure, who walked tranquilly, with a mildly ironic smile, and Bohr who trotted along beside him , full of excitement...The morning after the triumph of Bohr.*

George Gamov's make-believe experiment apparatus for validating the thought experiment at the Niels Bohr Institute in Copenhagen.

The "triumph of Bohr" consisted in his demonstrating once again, that Einstein's subtle argument was not conclusive, but even more so in the way that he arrived at this conclusion by appealing precisely to one of the great ideas of Einstein: the principle of equivalence between gravitational mass and inertial mass. Bohr showed that in order for Einstein's experiment to function, the box would have to be suspended on a spring in the middle of a gravitational field. In order to obtain a measurement of weight, a pointer would have to be attached to the box which corresponded with the index on a scale. After the release of a photon, weights could be added to the box to restore it to its original position and this would allow us

Figure 13D

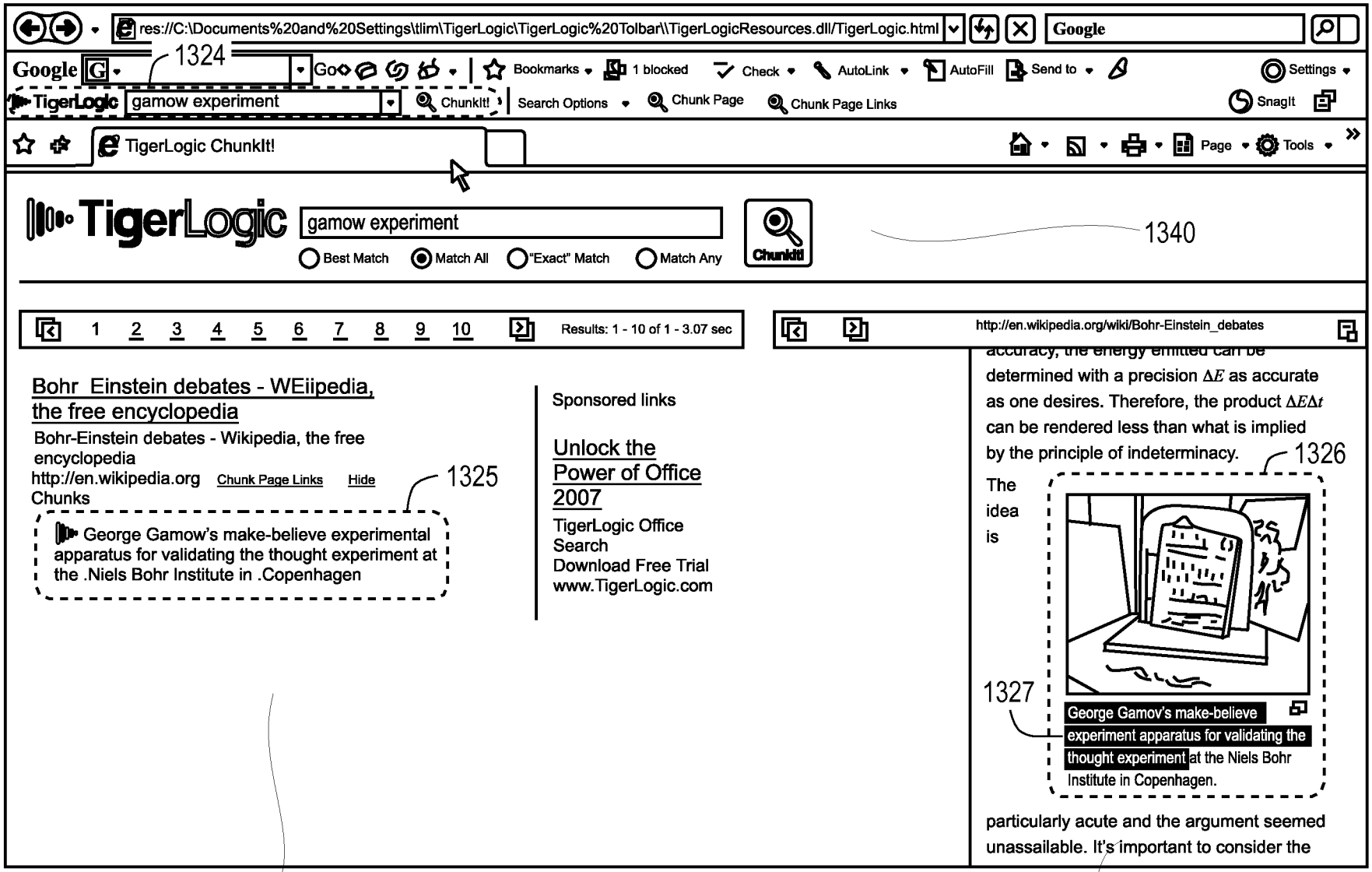

res://C:\Documents%20and%20Settings\tlim\TigerLogic\TigerLogic%20Tolbar\\TigerLogicResources.dll/TigerLogic.html
Google
1324
Google
Go
Bookmarks
1 blocked
Check
AutoLink
AutoFill
Send to
Settings
TigerLogic
gamow experiment
ChunkIt!
Search Options
Chunk Page
Chunk Page Links
SnagIt
TigerLogic ChunkIt!
Page
Tools
TigerLogic
gamow experiment
Best Match
Match All
"Exact" Match
Match Any
ChunkIt!
1340
1 2 3 4 5 6 7 8 9 10
Results: 1 - 10 of 1 - 3.07 sec
http://en.wikipedia.org/wiki/Bohr-Einstein_debates
Bohr Einstein debates - WEiipedia, the free encyclopedia
Bohr-Einstein debates - Wikipedia, the free encyclopedia
http://en.wikipedia.org Chunk Page Links Hide
Chunks
1325
George Gamow's make-believe experimental apparatus for validating the thought experiment at the .Niels Bohr Institute in .Copenhagen
Sponsored links
Unlock the Power of Office 2007
TigerLogic Office
Search
Download Free Trial
www.TigerLogic.com
determined with a precision ΔE as accurate as one desires. Therefore, the product ΔEΔt can be rendered less than what is implied by the principle of indeterminacy.
1326
The idea is
1327
George Gamov's make-believe experiment apparatus for validating the thought experiment at the Niels Bohr Institute in Copenhagen.
particularly acute and the argument seemed unassailable. It's important to consider the
1342
1344

Figure 13E

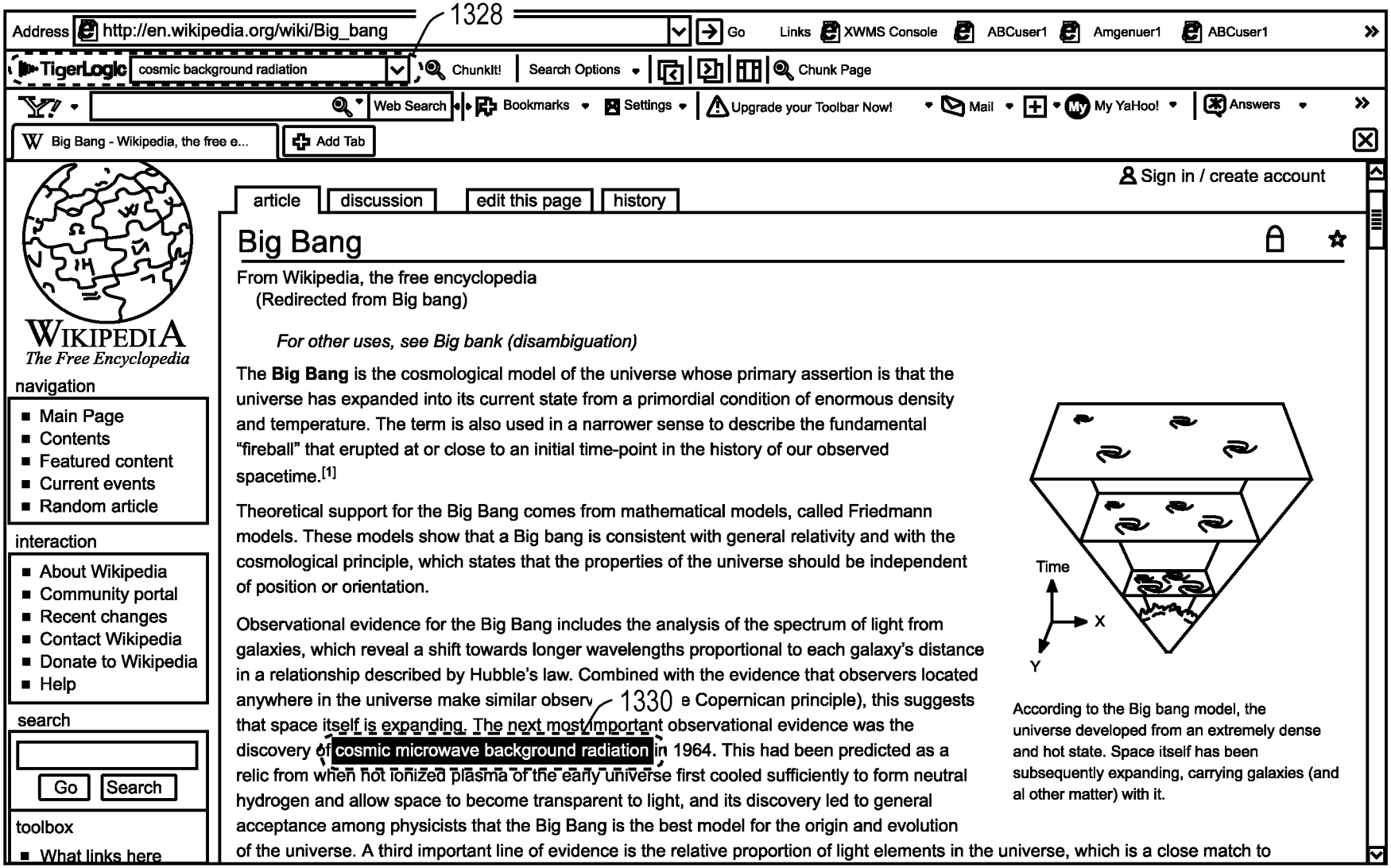

1328
Address http://en.wikipedia.org/wiki/Big_bang
Go
Links
XWMS Console
ABCuser1
Amgenuer1
ABCuser1
TigerLogic
cosmic background radiation
ChunkIt!
Search Options
Chunk Page
Web Search
Bookmarks
Settings
Upgrade your Toolbar Now!
Mail
My YaHoo!
Answers
Big Bang - Wikipedia, the free e...
Add Tab
Sign in / create account
article
discussion
edit this page
history
Big Bang
From Wikipedia, the free encyclopedia
(Redirected from Big bang)
For other uses, see Big bank (disambiguation)
The Big Bang is the cosmological model of the universe whose primary assertion is that the universe has expanded into its current state from a primordial condition of enormous density and temperature. The term is also used in a narrower sense to describe the fundamental "fireball" that erupted at or close to an initial time-point in the history of our observed spacetime.[1]
Theoretical support for the Big Bang comes from mathematical models, called Friedmann models. These models show that a Big bang is consistent with general relativity and with the cosmological principle, which states that the properties of the universe should be independent of position or orientation.
Observational evidence for the Big Bang includes the analysis of the spectrum of light from galaxies, which reveal a shift towards longer wavelengths proportional to each galaxy's distance in a relationship described by Hubble's law. Combined with the evidence that observers located anywhere in the universe make similar observ
1330
e Copernican principle), this suggests that space itself is expanding. The next most important observational evidence was the discovery of cosmic microwave background radiation in 1964. This had been predicted as a relic from when hot ionized plasma of the early universe first cooled sufficiently to form neutral hydrogen and allow space to become transparent to light, and its discovery led to general acceptance among physicists that the Big Bang is the best model for the origin and evolution of the universe. A third important line of evidence is the relative proportion of light elements in the universe, which is a close match to
Time
X
Y
According to the Big bang model, the universe developed from an extremely dense and hot state. Space itself has been subsequently expanding, carrying galaxies (and al other matter) with it.
WIKIPEDIA
The Free Encyclopedia
navigation
Main Page
Contents
Featured content
Current events
Random article
interaction
About Wikipedia
Community portal
Recent changes
Contact Wikipedia
Donate to Wikipedia
Help
search
Go
Search
toolbox
What links here

Figure 13F

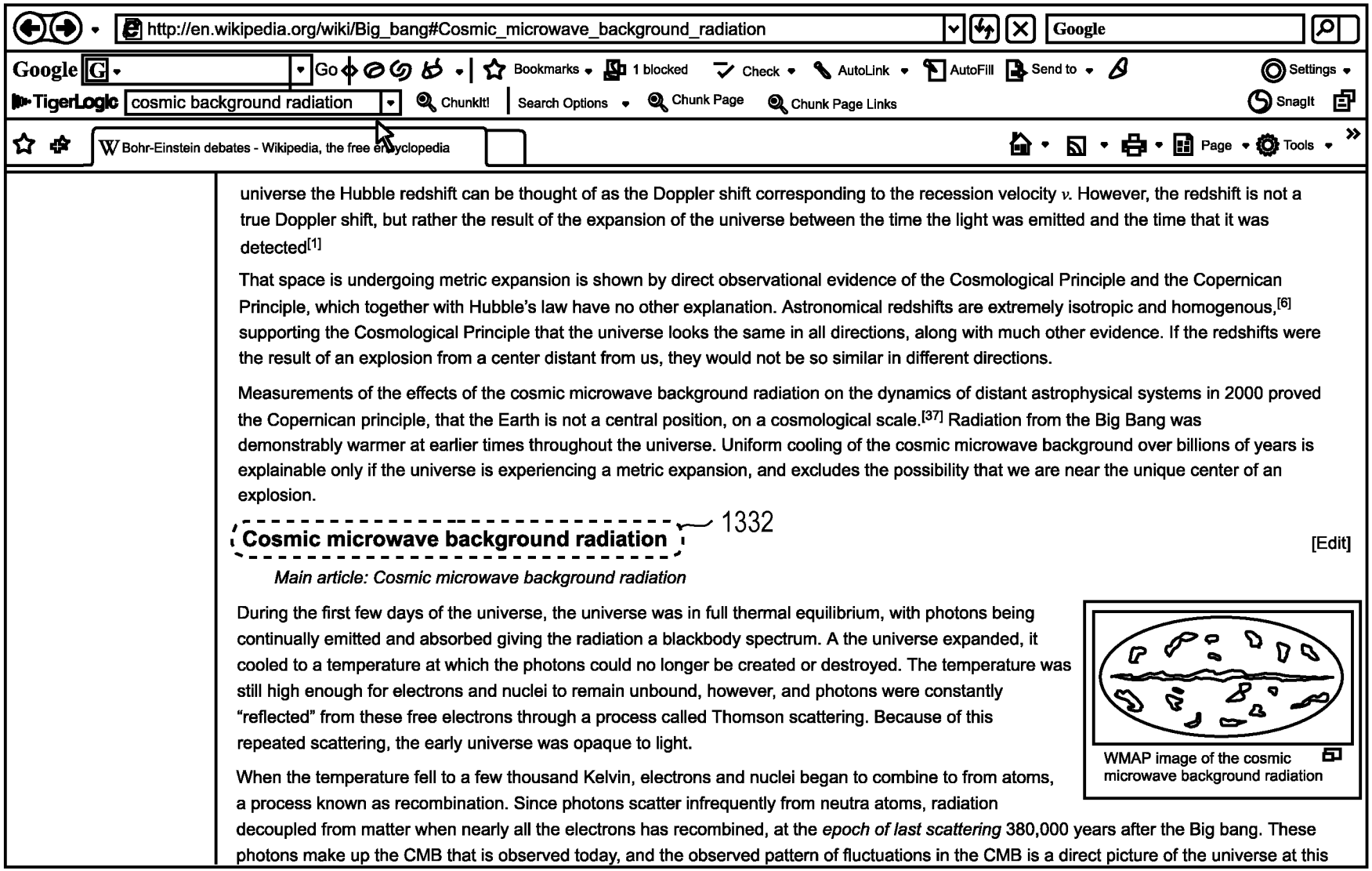

http://en.wikipedia.org/wiki/Big_bang#Cosmic_microwave_background_radiation
Google
Google
Go
Bookmarks
1 blocked
Check
AutoLink
AutoFill
Send to
Settings
TigerLogic
cosmic background radiation
ChunkIt!
Search Options
Chunk Page
Chunk Page Links
SnagIt
Bohr-Einstein debates - Wikipedia, the free encyclopedia
Page
Tools
universe the Hubble redshift can be thought of as the Doppler shift corresponding to the recession velocity v. However, the redshift is not a true Doppler shift, but rather the result of the expansion of the universe between the time the light was emitted and the time that it was detected[1]
That space is undergoing metric expansion is shown by direct observational evidence of the Cosmological Principle and the Copernican Principle, which together with Hubble's law have no other explanation. Astronomical redshifts are extremely isotropic and homogenous,[6] supporting the Cosmological Principle that the universe looks the same in all directions, along with much other evidence. If the redshifts were the result of an explosion from a center distant from us, they would not be so similar in different directions.
Measurements of the effects of the cosmic microwave background radiation on the dynamics of distant astrophysical systems in 2000 proved the Copernican principle, that the Earth is not a central position, on a cosmological scale.[37] Radiation from the Big Bang was demonstrably warmer at earlier times throughout the universe. Uniform cooling of the cosmic microwave background over billions of years is explainable only if the universe is experiencing a metric expansion, and excludes the possibility that we are near the unique center of an explosion.
1332
Cosmic microwave background radiation
[Edit]
Main article: Cosmic microwave background radiation
During the first few days of the universe, the universe was in full thermal equilibrium, with photons being continually emitted and absorbed giving the radiation a blackbody spectrum. A the universe expanded, it cooled to a temperature at which the photons could no longer be created or destroyed. The temperature was still high enough for electrons and nuclei to remain unbound, however, and photons were constantly "reflected" from these free electrons through a process called Thomson scattering. Because of this repeated scattering, the early universe was opaque to light.
WMAP image of the cosmic microwave background radiation
When the temperature fell to a few thousand Kelvin, electrons and nuclei began to combine to from atoms, a process known as recombination. Since photons scatter infrequently from neutra atoms, radiation decoupled from matter when nearly all the electrons has recombined, at the epoch of last scattering 380,000 years after the Big bang. These photons make up the CMB that is observed today, and the observed pattern of fluctuations in the CMB is a direct picture of the universe at this

Figure 13G

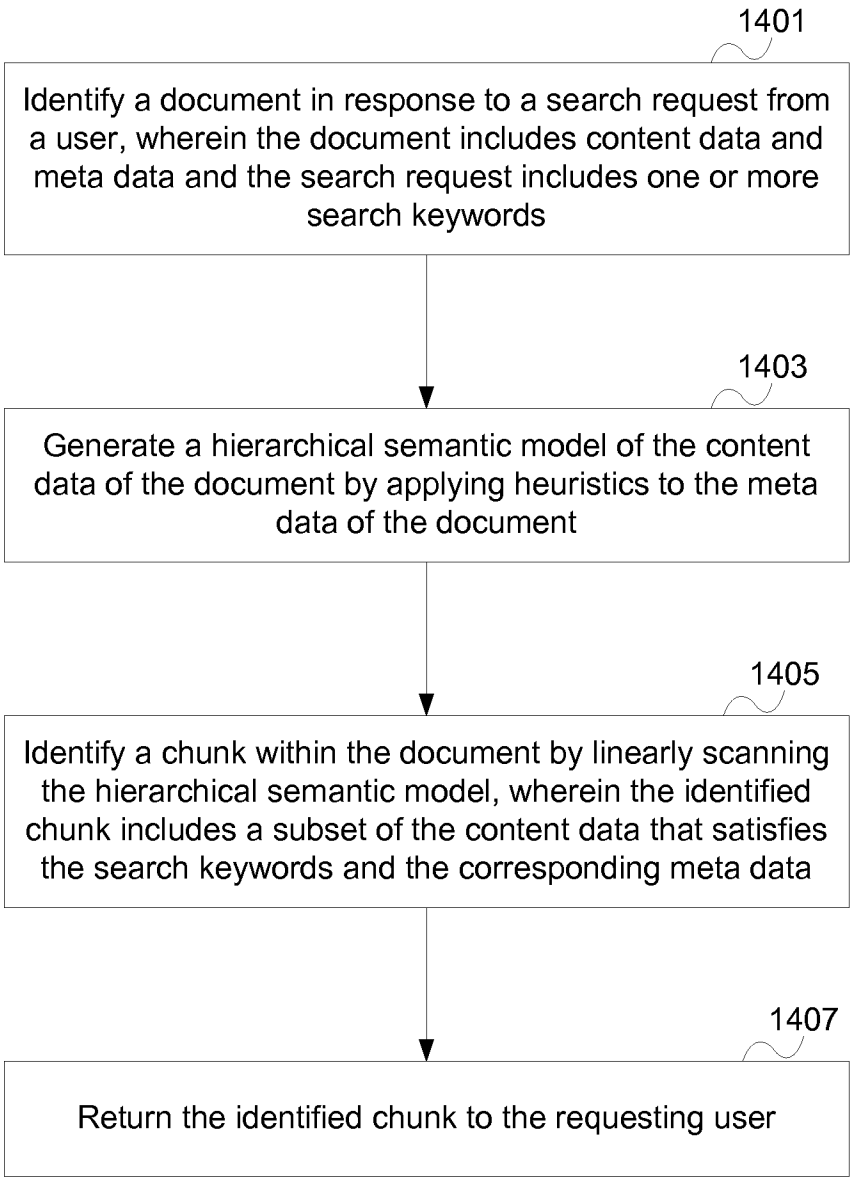

1401
Identify a document in response to a search request from a user, wherein the document includes content data and meta data and the search request includes one or more search keywords
1403
Generate a hierarchical semantic model of the content data of the document by applying heuristics to the meta data of the document
1405
Identify a chunk within the document by linearly scanning the hierarchical semantic model, wherein the identified chunk includes a subset of the content data that satisfies the search keywords and the corresponding meta data
1407
Return the identified chunk to the requesting user

Figure 14

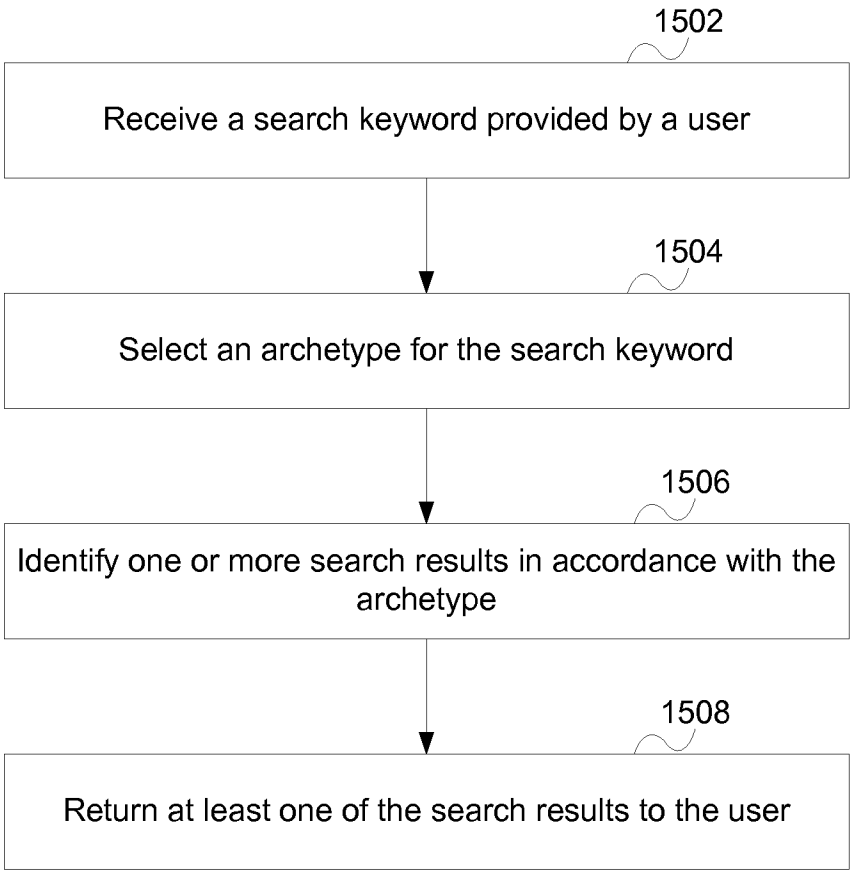

1502
Receive a search keyword provided by a user
1504
Select an archetype for the search keyword
1506
Identify one or more search results in accordance with the archetype
1508
Return at least one of the search results to the user

Figure 15

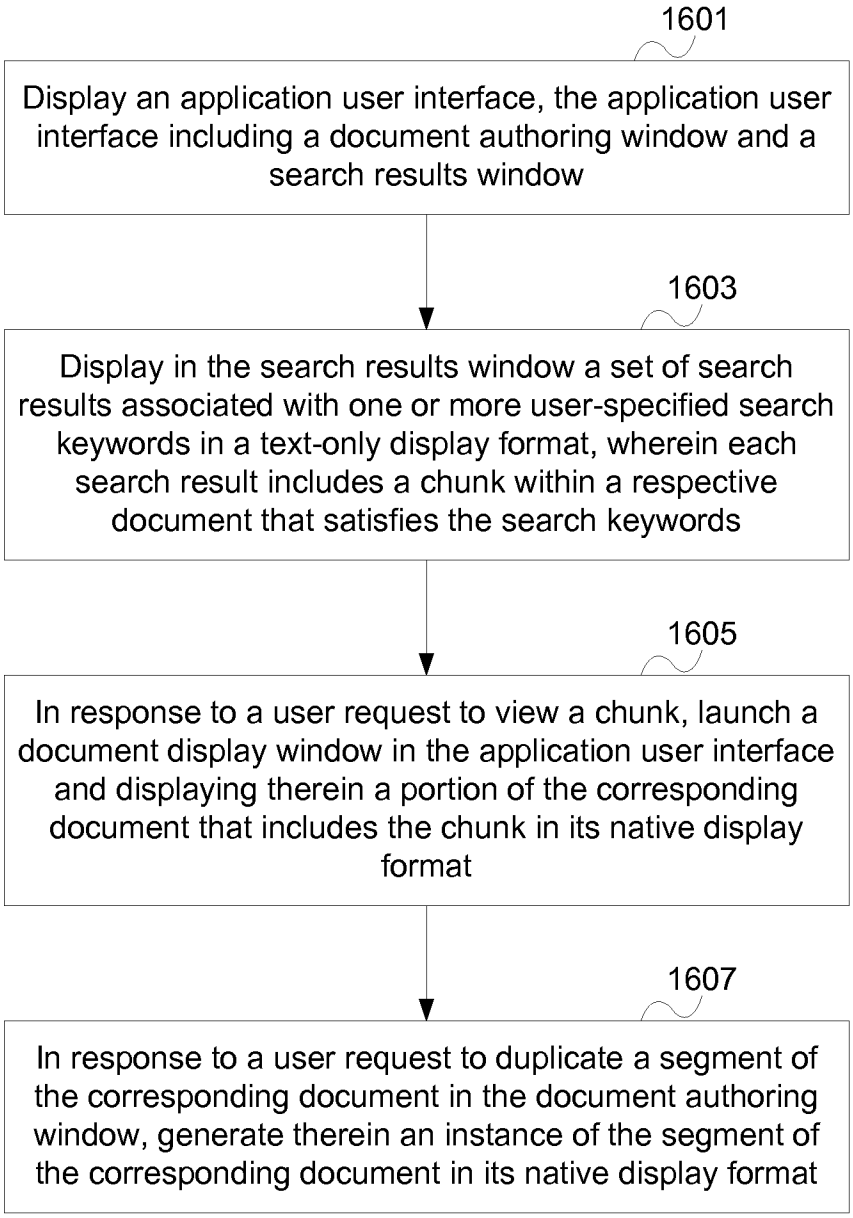

1601
Display an application user interface, the application user interface including a document authoring window and a search results window
1603
Display in the search results window a set of search results associated with one or more user-specified search keywords in a text-only display format, wherein each search result includes a chunk within a respective document that satisfies the search keywords
1605
In response to a user request to view a chunk, launch a document display window in the application user interface and displaying therein a portion of the corresponding document that includes the chunk in its native display format
1607
In response to a user request to duplicate a segment of the corresponding document in the document authoring window, generate therein an instance of the segment of the corresponding document in its native display format

Figure 16A

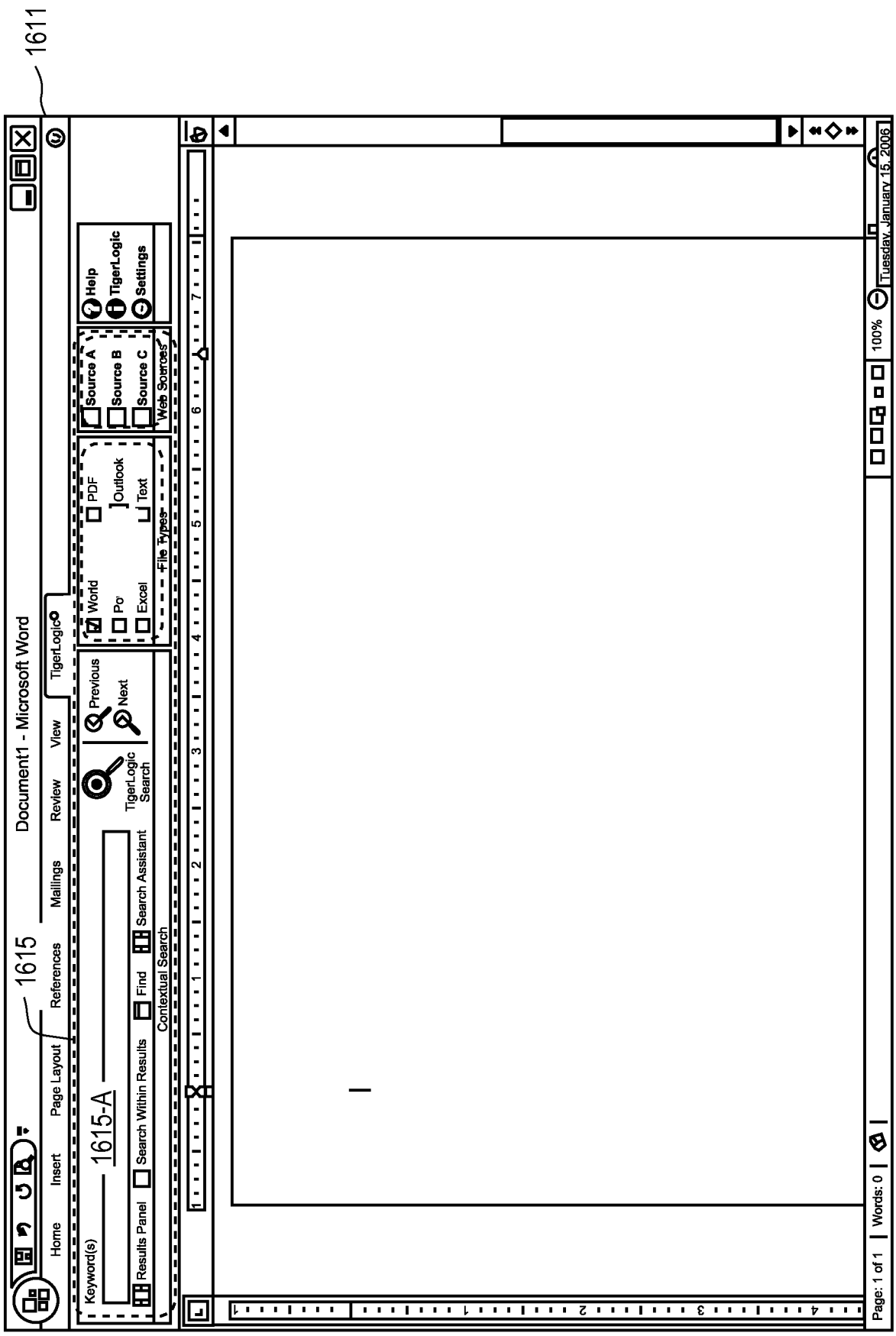

1611
1615
1615-A
Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
TigerLogic
Keyword(s)
Results Panel
Search Within Results
Find
Search Assistant
Contextual Search
TigerLogic Search
Previous
Next
World
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
Page: 1 of 1
Words: 0
100%
Tuesday, January 15, 2006

Figure 16B

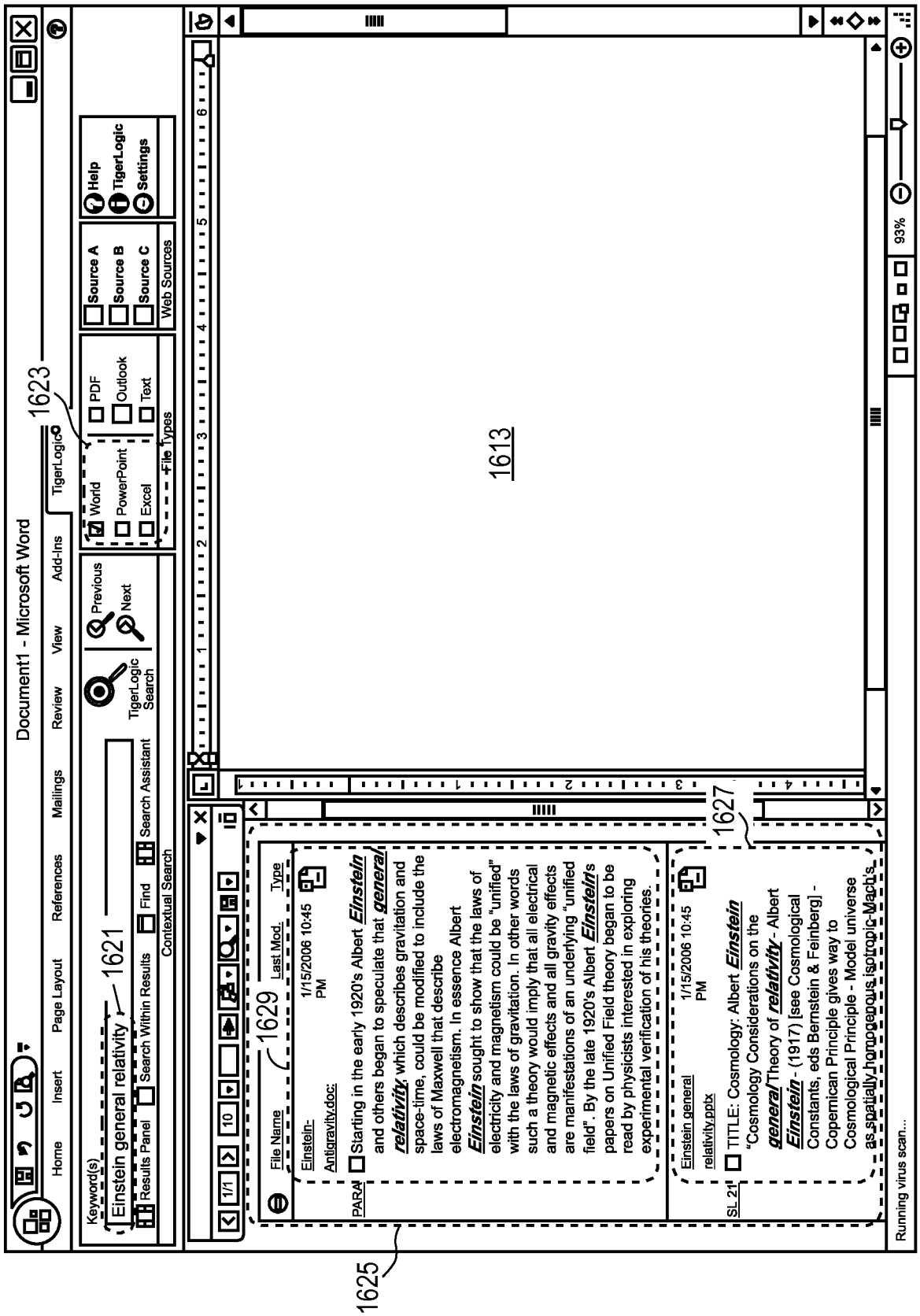

Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Add-Ins
TigerLogic
Keyword(s)
Einstein general relativity
Results Panel
Search Within Results
Find
Search Assistant
TigerLogic Search
Previous
Next
Contextual Search
World
PowerPoint
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
1621
1623
1613
1625
1627
1629
File Name
Last Mod.
Type
Einstein-Antigravity.doc:
1/15/2006 10:45 PM
PARA
Starting in the early 1920's Albert Einstein and others began to speculate that general relativity which describes gravitation and space-time, could be modified to include the laws of Maxwell that describe electromagnetism. In essence Albert Einstein sought to show that the laws of electricity and magnetism could be "unified" with the laws of gravitation. In other words such a theory would imply that all electrical and magnetic effects and all gravity effects are manifestations of an underlying "unified field". By the late 1920's Albert Einstein's papers on Unified Field theory began to be read by physicists interested in exploring experimental verification of his theories.
Einstein general relativity.pptx
1/15/2006 10:45 PM
SL 21
TITLE: Cosmology: Albert Einstein "Cosmology Considerations on the general Theory of relativity - Albert Einstein - (1917) [see Cosmological Constants, eds Bernstein & Feinberg] - Copernican Principle gives way to Cosmological Principle - Model universe as spatially homogenous isotropic Mach's
Running virus scan...
93%

Figure 16C

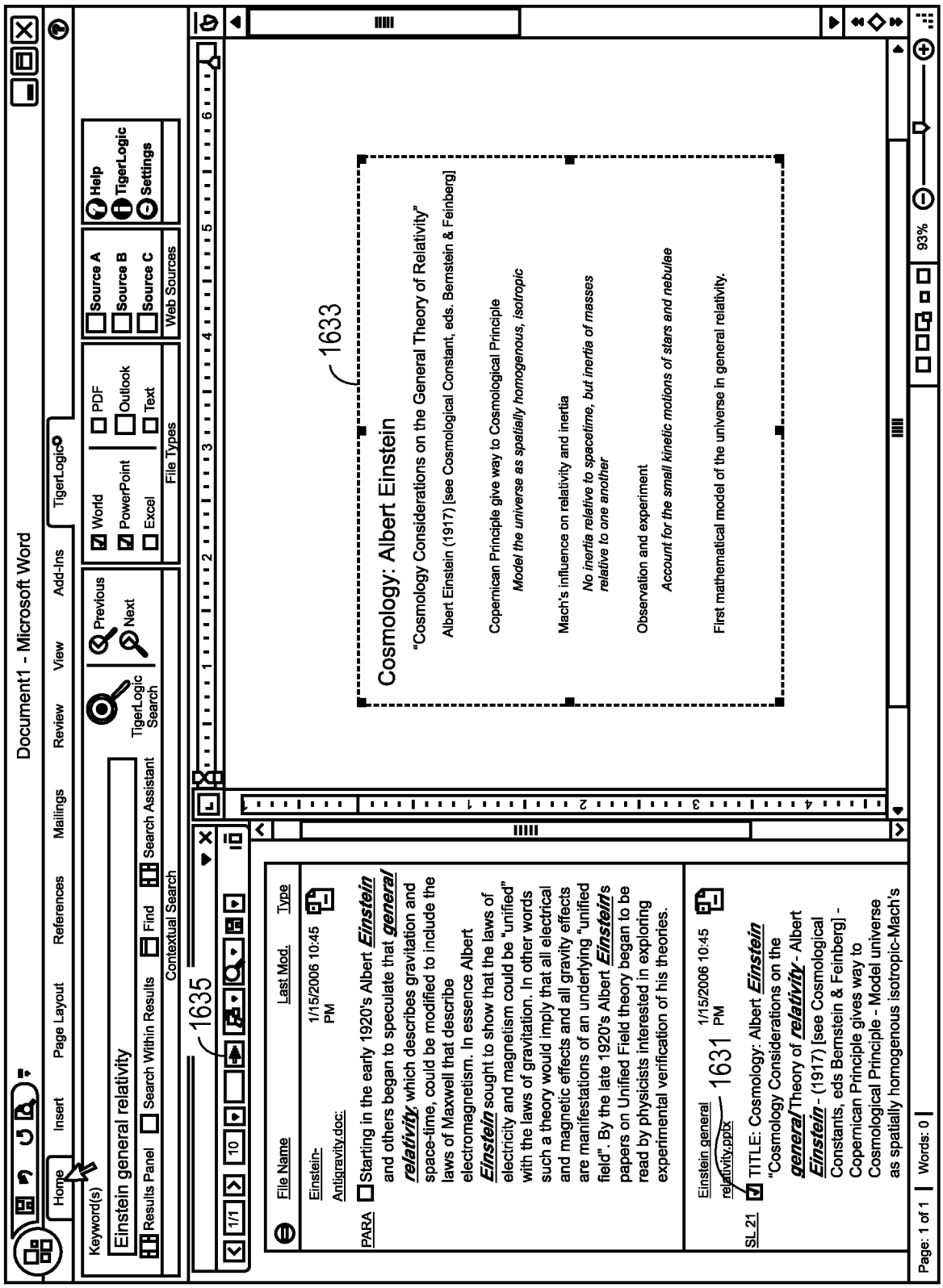

Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Add-Ins
TigerLogic
Keyword(s)
Einstein general relativity
Results Panel
Search Within Results
Find
Search Assistant
TigerLogic Search
Previous
Next
Contextual Search
World
PowerPoint
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
1635
1633
File Name
Last Mod.
Type
Einstein-Antigravity.doc:
1/15/2006 10:45 PM
PARA
Starting in the early 1920's Albert Einstein and others began to speculate that general relativity, which describes gravitation and space-time, could be modified to include the laws of Maxwell that describe electromagnetism. In essence Albert Einstein sought to show that the laws of electricity and magnetism could be "unified" with the laws of gravitation. In other words such a theory would imply that all electrical and magnetic effects and all gravity effects are manifestations of an underlying "unified field". By the late 1920's Albert Einstein's papers on Unified Field theory began to be read by physicists interested in exploring experimental verification of his theories.
Einstein general relativity.pptx
1/15/2006 10:45 PM
1631
SL 21
TITLE: Cosmology: Albert Einstein "Cosmology Considerations on the general Theory of relativity - Albert Einstein - (1917) [see Cosmological Constants, eds Bernstein & Feinberg] - Copernican Principle gives way to Cosmological Principle - Model universe as spatially homogenous isotropic-Mach's
Cosmology: Albert Einstein
"Cosmology Considerations on the General Theory of Relativity"
Albert Einstein (1917) [see Cosmological Constant, eds. Bernstein & Feinberg]
Copernican Principle give way to Cosmological Principle
Model the universe as spatially homogenous, isotropic
Mach's influence on relativity and inertia
No inertia relative to spacetime, but inertia of masses relative to one another
Observation and experiment
Account for the small kinetic motions of stars and nebulae
First mathematical model of the universe in general relativity.
Page: 1 of 1
Words: 0
93%

Figure 16D

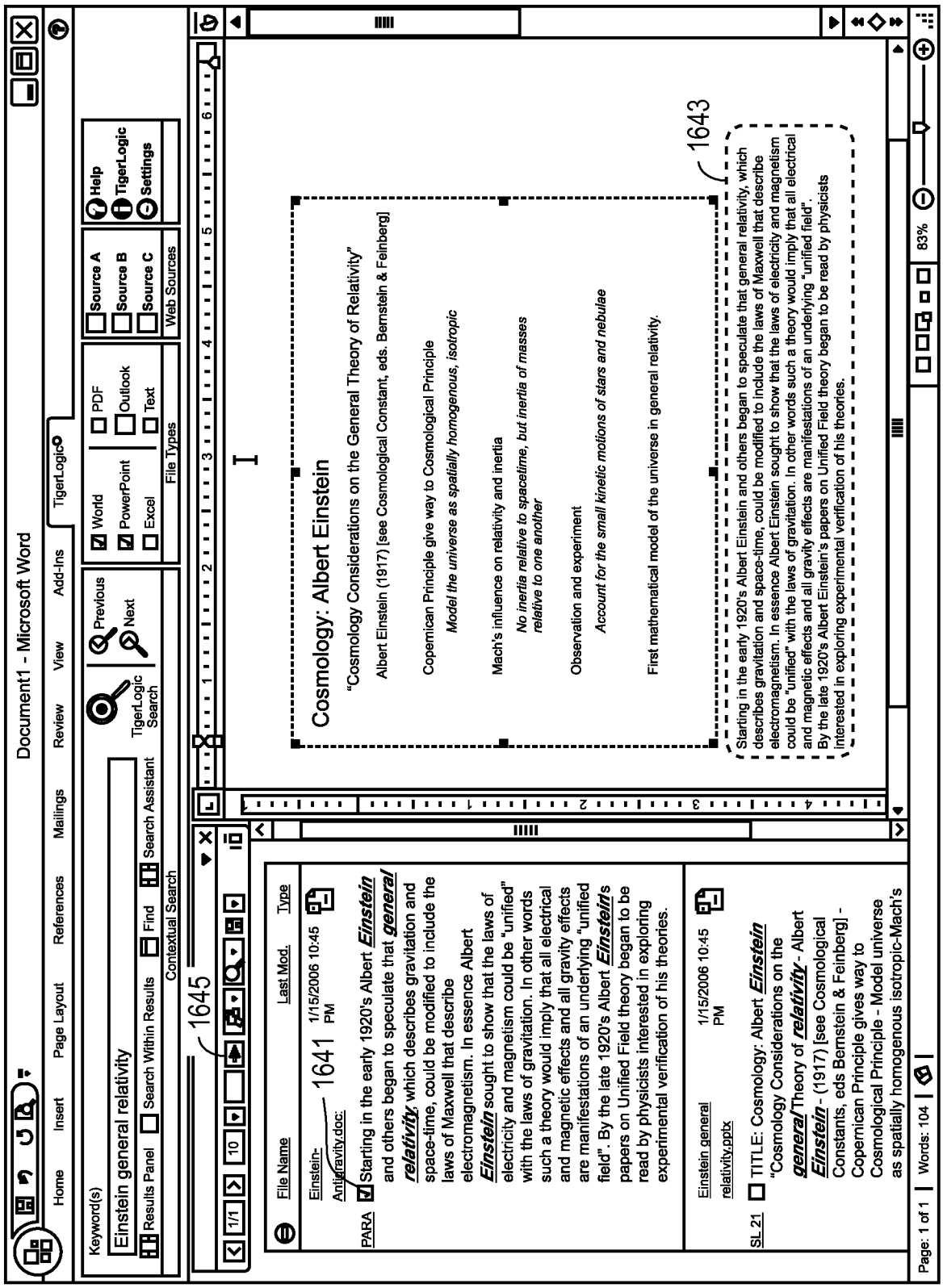

Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Add-Ins
TigerLogic
Keyword(s)
Einstein general relativity
Results Panel
Search Within Results
Find
Search Assistant
Contextual Search
TigerLogic Search
Previous
Next
World
PowerPoint
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
1645
1/1
10
File Name
Last Mod.
Type
Einstein-Antigravity.doc:
1/15/2006 10:45 PM
1641
PARA
Starting in the early 1920's Albert Einstein and others began to speculate that general relativity which describes gravitation and space-time, could be modified to include the laws of Maxwell that describe electromagnetism. In essence Albert Einstein sought to show that the laws of electricity and magnetism could be "unified" with the laws of gravitation. In other words such a theory would imply that all electrical and magnetic effects and all gravity effects are manifestations of an underlying "unified field". By the late 1920's Albert Einstein's papers on Unified Field theory began to be read by physicists interested in exploring experimental verification of his theories.
Einstein general relativity.pptx
1/15/2006 10:45 PM
SL 21
TITLE: Cosmology: Albert Einstein "Cosmology Considerations on the general/Theory of relativity - Albert Einstein - (1917) [see Cosmological Constants, eds Bernstein & Feinberg] - Copernican Principle gives way to Cosmological Principle - Model universe as spatially homogenous isotropic-Mach's
Page: 1 of 1
Words: 104
83%
Cosmology: Albert Einstein
"Cosmology Considerations on the General Theory of Relativity"
Albert Einstein (1917) [see Cosmological Constant, eds. Bernstein & Feinberg]
Copernican Principle give way to Cosmological Principle
Model the universe as spatially homogenous, isotropic
Mach's influence on relativity and inertia
No inertia relative to spacetime, but inertia of masses relative to one another
Observation and experiment
Account for the small kinetic motions of stars and nebulae
First mathematical model of the universe in general relativity.
1643
Starting in the early 1920's Albert Einstein and others began to speculate that general relativity, which describes gravitation and space-time, could be modified to include the laws of Maxwell that describe electromagnetism. In essence Albert Einstein sought to show that the laws of electricity and magnetism could be "unified" with the laws of gravitation. In other words such a theory would imply that all electrical and magnetic effects and all gravity effects are manifestations of an underlying "unified field". By the late 1920's Albert Einstein's papers on Unified Field theory began to be read by physicists interested in exploring experimental verification of his theories.

Figure 16E

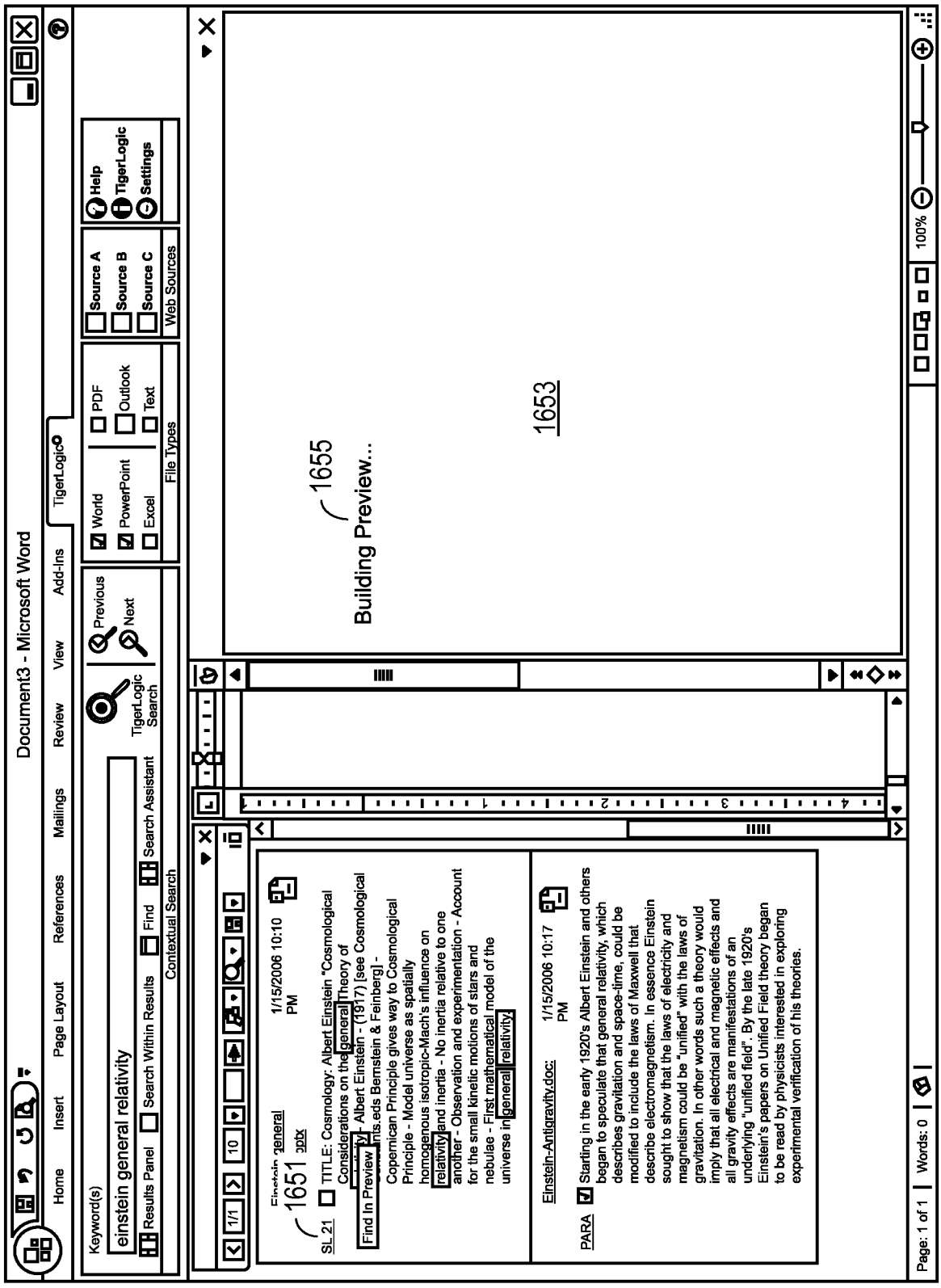

Document3 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Add-Ins
TigerLogic
Keyword(s)
einstein general relativity
Results Panel
Search Within Results
Find
Search Assistant
Contextual Search
TigerLogic Search
Previous
Next
World
PowerPoint
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
1655
Building Preview...
1653
1651
SL 21
TITLE: Cosmology: Albert Einstein "Cosmological Considerations on the General Theory of Relativity" - Albert Einstein - (1917) [see Cosmological Principle - Model universe as spatially homogenous isotropic-Mach's influence on relativity and inertia - No inertia relative to one another - Observation and experimentation - Account for the small kinetic motions of stars and nebulae - First mathematical model of the universe in general relativity
Find In Preview
PARA
Starting in the early 1920's Albert Einstein and others began to speculate that general relativity, which describes gravitation and space-time, could be modified to include the laws of Maxwell that describe electromagnetism. In essence Einstein sought to show that the laws of electricity and magnetism could be "unified" with the laws of gravitation. In other words such a theory would imply that all electrical and magnetic effects and all gravity effects are manifestations of an underlying "unified field". By the late 1920's Einstein's papers on Unified Field theory began to be read by physicists interested in exploring experimental verification of his theories.
Page: 1 of 1
Words: 0
100%

Figure 16F

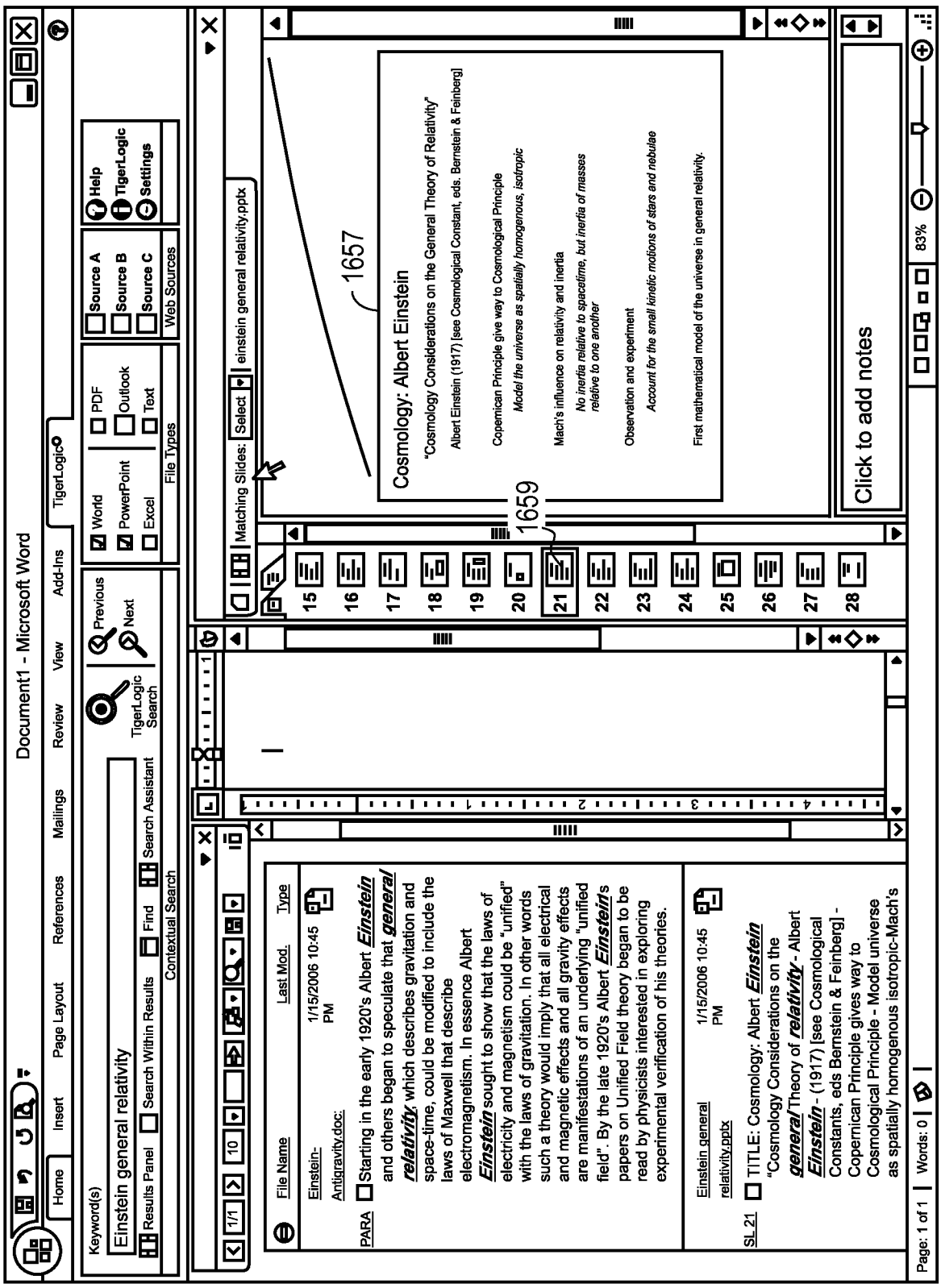

Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Add-Ins
TigerLogic
Keyword(s)
Einstein general relativity
Results Panel
Search Within Results
Find
Search Assistant
TigerLogic Search
Previous
Next
Contextual Search
World
PowerPoint
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
Matching Slides:
Select
einstein general relativity.pptx
1657
Cosmology: Albert Einstein
"Cosmology Considerations on the General Theory of Relativity"
Albert Einstein (1917) [see Cosmological Constant, eds. Bernstein & Feinberg]
Copernican Principle give way to Cosmological Principle
Model the universe as spatially homogenous, isotropic
Mach's influence on relativity and inertia
No inertia relative to spacetime, but inertia of masses relative to one another
Observation and experiment
Account for the small kinetic motions of stars and nebulae
First mathematical model of the universe in general relativity.
1659
Click to add notes
File Name
Last Mod.
Type
Einstein-Antigravity.doc:
1/15/2006 10:45 PM
PARA
Starting in the early 1920's Albert Einstein and others began to speculate that general relativity which describes gravitation and space-time, could be modified to include the laws of Maxwell that describe electromagnetism. In essence Albert Einstein sought to show that the laws of electricity and magnetism could be "unified" with the laws of gravitation. In other words such a theory would imply that all electrical and magnetic effects and all gravity effects are manifestations of an underlying "unified field". By the late 1920's Albert Einstein's papers on Unified Field theory began to be read by physicists interested in exploring experimental verification of his theories.
Einstein general relativity.pptx
1/15/2006 10:45 PM
SL 21
TITLE: Cosmology: Albert Einstein "Cosmology Considerations on the general Theory of relativity - Albert Einstein - (1917) [see Cosmological Constants, eds Bernstein & Feinberg] - Copernican Principle gives way to Cosmological Principle - Model universe as spatially homogenous isotropic-Mach's
Page: 1 of 1
Words: 0
83%

Figure 16G

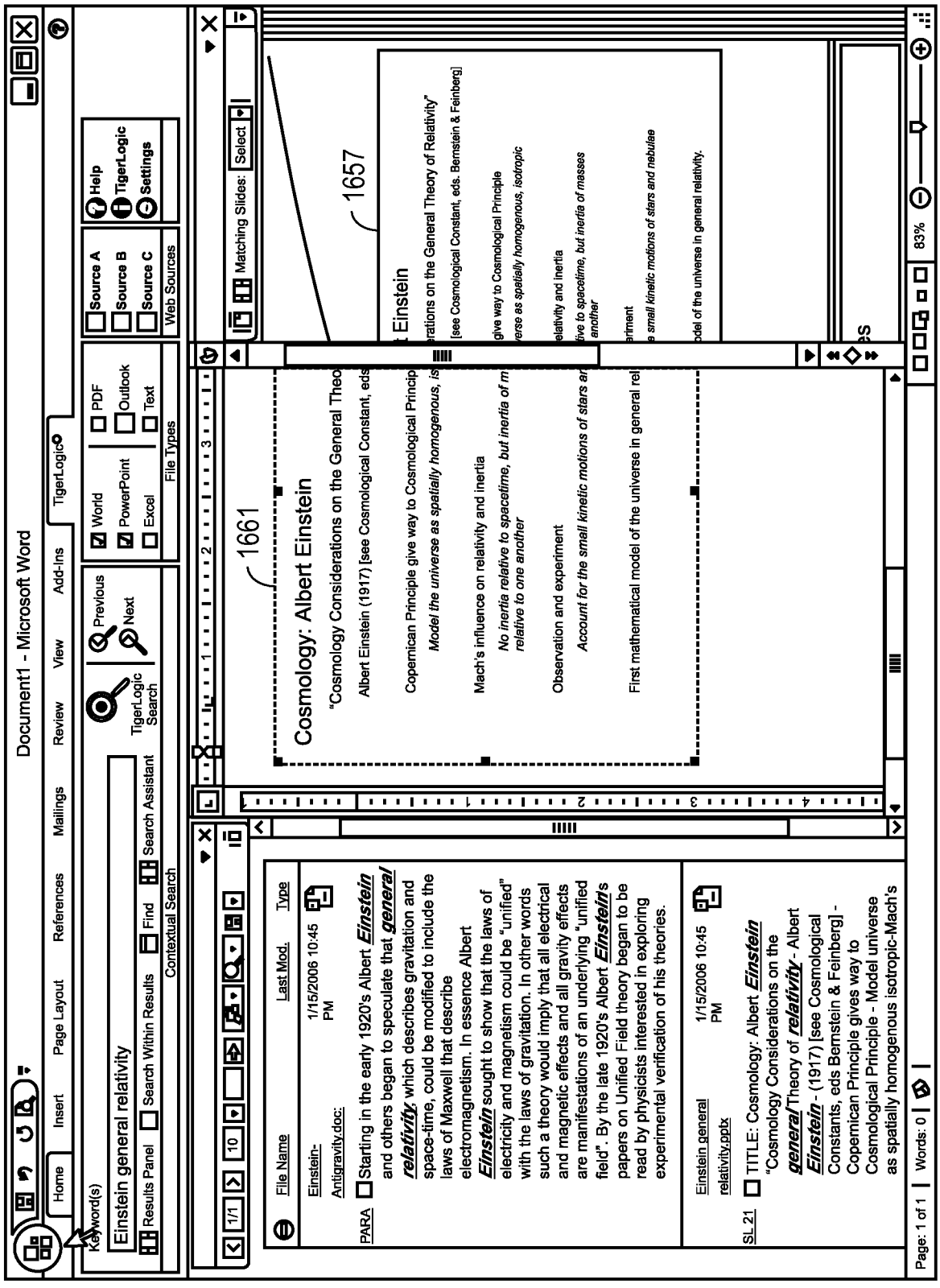

Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Add-Ins
TigerLogic
Keyword(s)
Einstein general relativity
Results Panel
Search Within Results
Find
Search Assistant
Contextual Search
TigerLogic Search
Previous
Next
World
PowerPoint
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
Matching Slides:
Select
1661
Cosmology: Albert Einstein
"Cosmology Considerations on the General Theo
Albert Einstein (1917) [see Cosmological Constant, eds
Copernican Principle give way to Cosmological Princip
Model the universe as spatially homogenous, is
Mach's influence on relativity and inertia
No inertia relative to spacetime, but inertia of m
relative to one another
Observation and experiment
Account for the small kinetic motions of stars ar
First mathematical model of the universe in general rel
1657
Einstein
rations on the General Theory of Relativity"
[see Cosmological Constant, eds. Bernstein & Feinberg]
give way to Cosmological Principle
erse as spatially homogenous, isotropic
elativity and inertia
tive to spacetime, but inertia of masses
another
arirment
e small kinetic motions of stars and nebulae
odel of the universe in general relativity.
File Name
Last Mod.
Type
Einstein-Antigravity.doc:
1/15/2006 10:45 PM
PARA
Starting in the early 1920's Albert Einstein and others began to speculate that general relativity which describes gravitation and space-time, could be modified to include the laws of Maxwell that describe electromagnetism. In essence Albert Einstein sought to show that the laws of electricity and magnetism could be "unified" with the laws of gravitation. In other words such a theory would imply that all electrical and magnetic effects and all gravity effects are manifestations of an underlying "unified field". By the late 1920's Albert Einstein's papers on Unified Field theory began to be read by physicists interested in exploring experimental verification of his theories.
Einstein general relativity.pptx
1/15/2006 10:45 PM
SL 21
TITLE: Cosmology: Albert Einstein "Cosmology Considerations on the general Theory of relativity - Albert Einstein - (1917) [see Cosmological Constants, eds Bernstein & Feinberg] - Copernican Principle gives way to Cosmological Principle - Model universe as spatially homogenous isotropic-Mach's
Page: 1 of 1
Words: 0
83%

Figure 16H

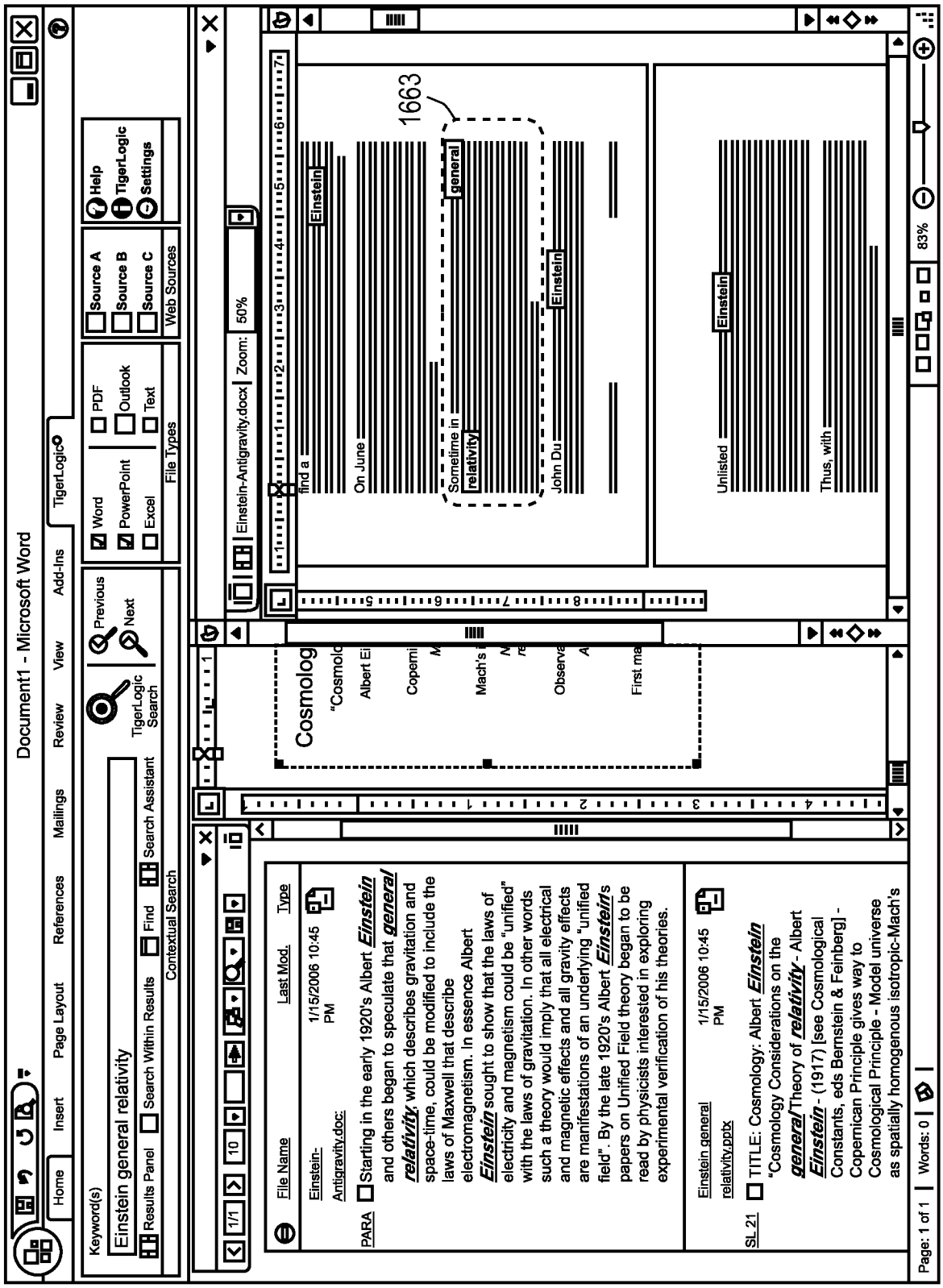

Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Add-Ins
TigerLogic
Keyword(s)
Einstein general relativity
Results Panel
Search Within Results
Find
Search Assistant
Contextual Search
TigerLogic Search
Previous
Next
Word
PowerPoint
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
Einstein-Antigravity.docx
Zoom: 50%
1663
Cosmolog
find a
Einstein
On June
Sometime in
general
relativity
John Du
Einstein
Unlisted
Einstein
Thus, with
File Name
Last Mod.
Type
Einstein-Antigravity.doc:
1/15/2006 10:45 PM
PARA
Starting in the early 1920's Albert Einstein and others began to speculate that general relativity which describes gravitation and space-time, could be modified to include the laws of Maxwell that describe electromagnetism. In essence Albert Einstein sought to show that the laws of electricity and magnetism could be "unified" with the laws of gravitation. In other words such a theory would imply that all electrical and magnetic effects and all gravity effects are manifestations of an underlying "unified field". By the late 1920's Albert Einstein's papers on Unified Field theory began to be read by physicists interested in exploring experimental verification of his theories.
Einstein general relativity.pptx
1/15/2006 10:45 PM
SL 21
TITLE: Cosmology: Albert Einstein "Cosmology Considerations on the general Theory of relativity - Albert Einstein - (1917) [see Cosmological Constants, eds Bernstein & Feinberg] - Copernican Principle gives way to Cosmological Principle - Model universe as spatially homogenous isotropic-Mach's
Page: 1 of 1
Words: 0
83%

Figure 16I

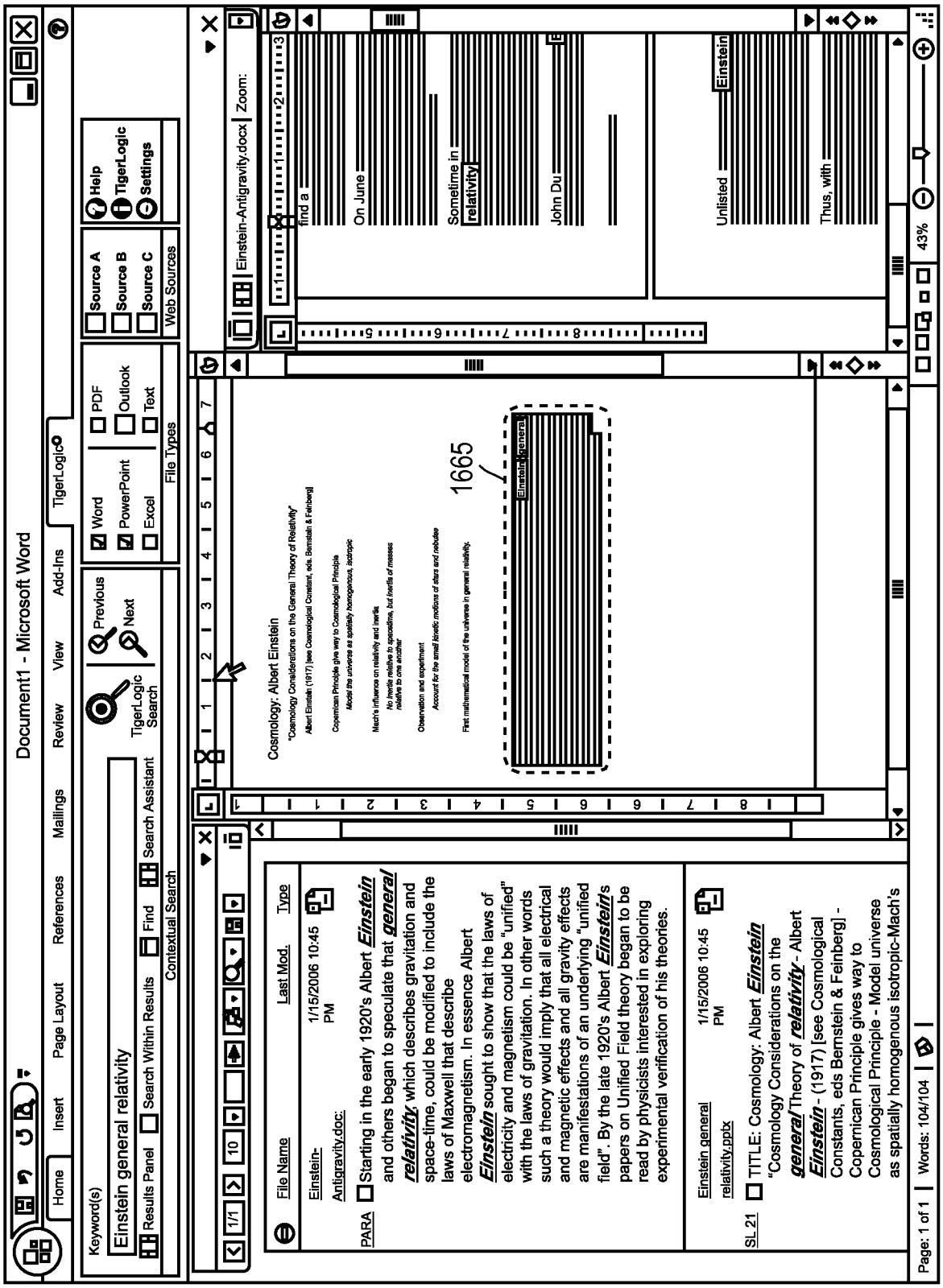

Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Add-Ins
TigerLogic
Keyword(s)
Einstein general relativity
Results Panel
Search Within Results
Find
Search Assistant
Contextual Search
TigerLogic Search
Previous
Next
Word
PowerPoint
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
Einstein-Antigravity.docx
Zoom:
Cosmology: Albert Einstein
"Cosmology Considerations on the General Theory of Relativity"
Albert Einstein (1917) [see Cosmological Constant, eds. Bernstein & Feinberg]
Copernican Principle gives way to Cosmological Principle
Model the universe as spatially homogenous, isotropic
Mach's influence on relativity and inertia
No inertia relative to spacetime, but inertia of masses relative to one another
Observation and experiment
Account for the small kinetic motions of stars and nebulae
First mathematical model of the universe in general relativity.
1665
Einstein general
File Name
Last Mod.
Type
Einstein-Antigravity.doc:
1/15/2006 10:45 PM
PARA
Starting in the early 1920's Albert Einstein and others began to speculate that general relativity which describes gravitation and space-time, could be modified to include the laws of Maxwell that describe electromagnetism. In essence Albert Einstein sought to show that the laws of electricity and magnetism could be "unified" with the laws of gravitation. In other words such a theory would imply that all electrical and magnetic effects and all gravity effects are manifestations of an underlying "unified field". By the late 1920's Albert Einstein's papers on Unified Field theory began to be read by physicists interested in exploring experimental verification of his theories.
Einstein general relativity.pptx
1/15/2006 10:45 PM
SL 21
TITLE: Cosmology: Albert Einstein "Cosmology Considerations on the general Theory of relativity - Albert Einstein - (1917) [see Cosmological Constants, eds Bernstein & Feinberg] - Copernican Principle gives way to Cosmological Principle - Model universe as spatially homogenous isotropic-Mach's
find a
On June
Sometime in
relativity
John Du
Unlisted
Einstein
Thus, with
Page: 1 of 1
Words: 104/104
43%

Figure 16J

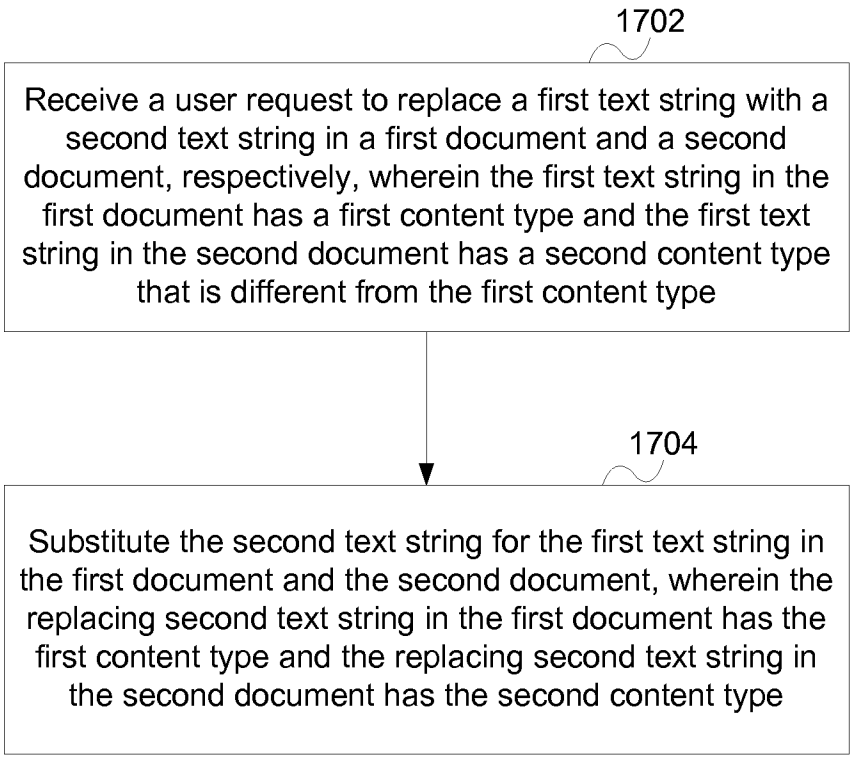

1702
Receive a user request to replace a first text string with a second text string in a first document and a second document, respectively, wherein the first text string in the first document has a first content type and the first text string in the second document has a second content type that is different from the first content type
1704
Substitute the second text string for the first text string in the first document and the second document, wherein the replacing second text string in the first document has the first content type and the replacing second text string in the second document has the second content type

Figure 17A

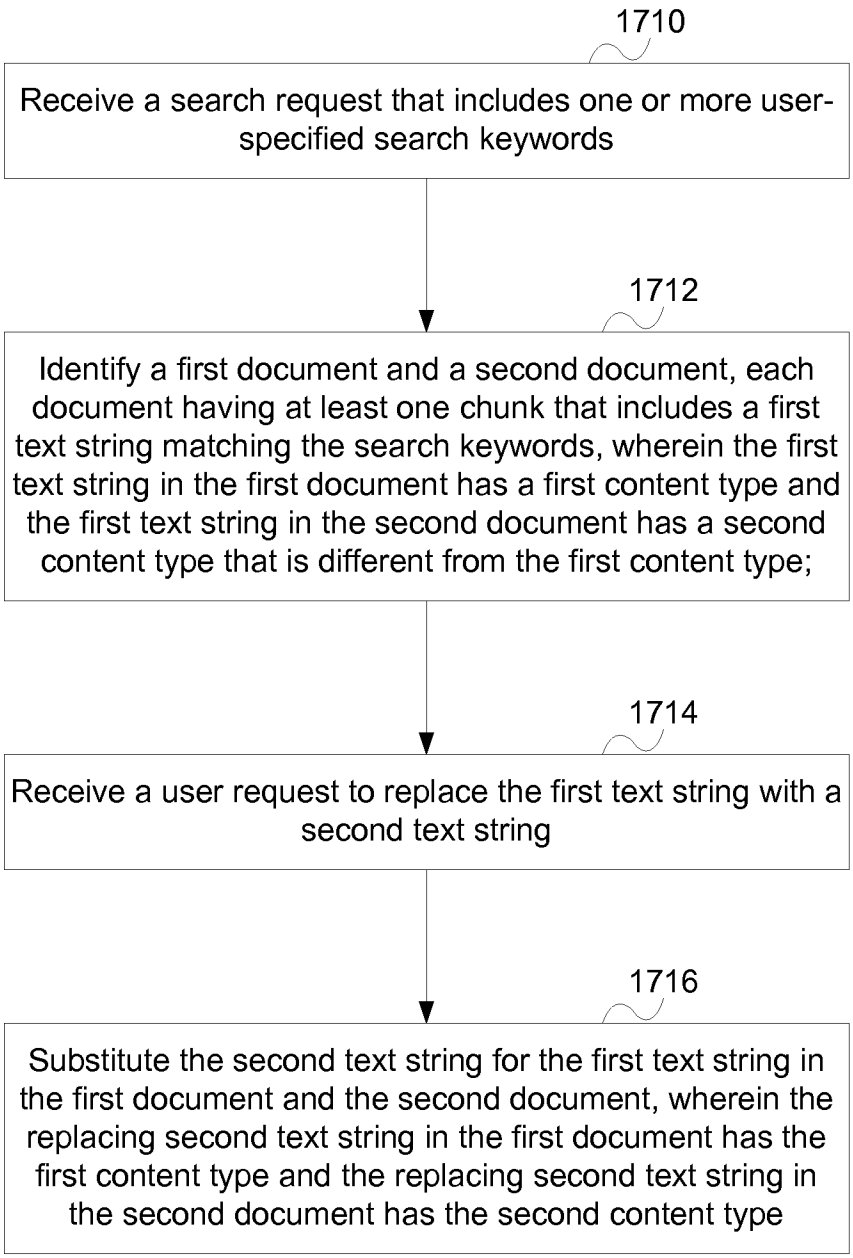

1710
Receive a search request that includes one or more user-specified search keywords
1712
Identify a first document and a second document, each document having at least one chunk that includes a first text string matching the search keywords, wherein the first text string in the first document has a first content type and the first text string in the second document has a second content type that is different from the first content type;
1714
Receive a user request to replace the first text string with a second text string
1716
Substitute the second text string for the first text string in the first document and the second document, wherein the replacing second text string in the first document has the first content type and the replacing second text string in the second document has the second content type

Figure 17B

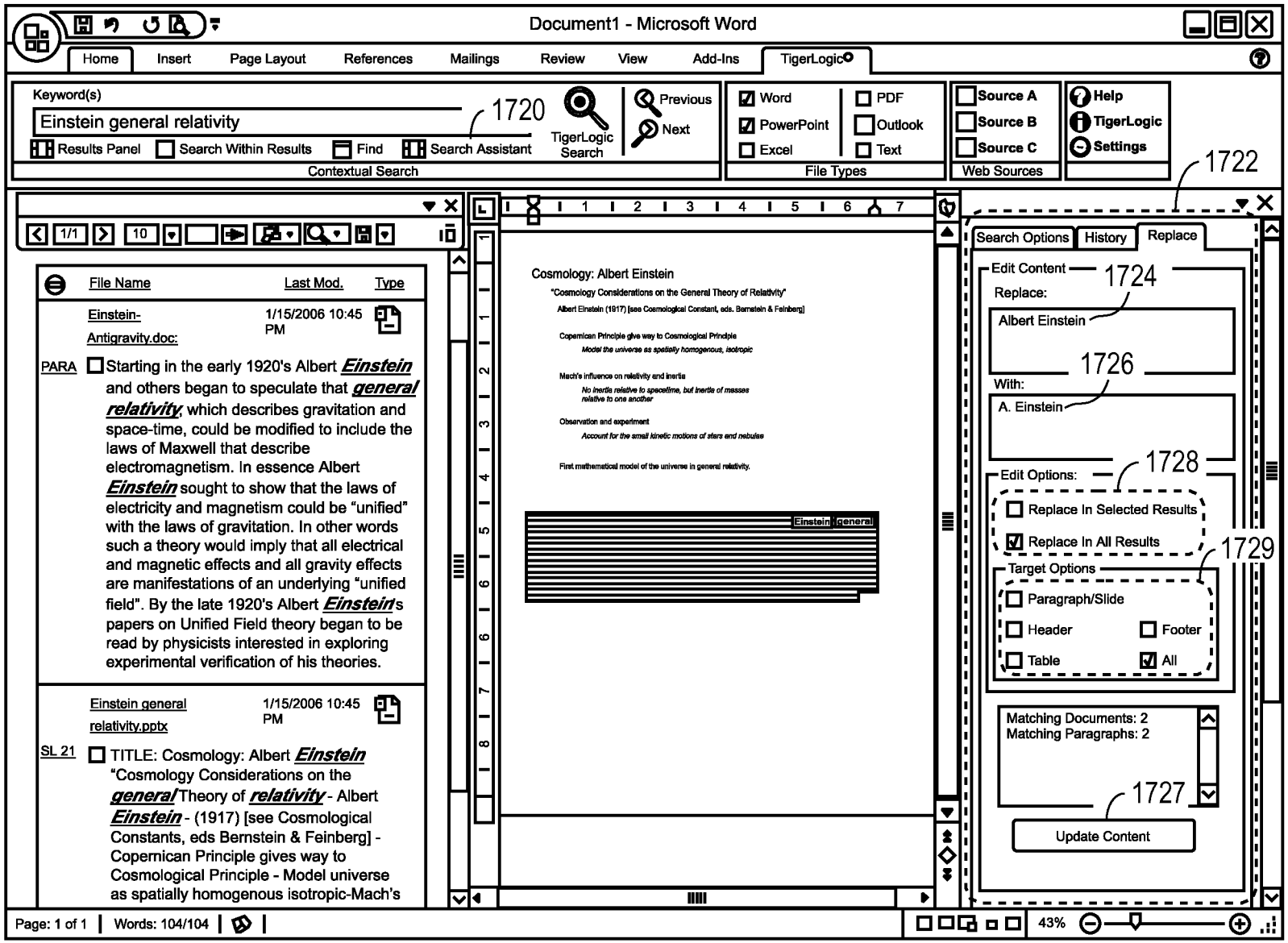

Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Add-Ins
TigerLogic
Keyword(s)
Einstein general relativity
1720
Results Panel
Search Within Results
Find
Search Assistant
TigerLogic Search
Previous
Next
Contextual Search
Word
PowerPoint
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
1722
File Name
Last Mod.
Type
Einstein-Antigravity.doc:
1/15/2006 10:45 PM
PARA
Starting in the early 1920's Albert Einstein and others began to speculate that general relativity, which describes gravitation and space-time, could be modified to include the laws of Maxwell that describe electromagnetism. In essence Albert Einstein sought to show that the laws of electricity and magnetism could be "unified" with the laws of gravitation. In other words such a theory would imply that all electrical and magnetic effects and all gravity effects are manifestations of an underlying "unified field". By the late 1920's Albert Einstein's papers on Unified Field theory began to be read by physicists interested in exploring experimental verification of his theories.
Einstein general relativity.pptx
1/15/2006 10:45 PM
SL 21
TITLE: Cosmology: Albert Einstein "Cosmology Considerations on the general Theory of relativity - Albert Einstein - (1917) [see Cosmological Constants, eds Bernstein & Feinberg] - Copernican Principle gives way to Cosmological Principle - Model universe as spatially homogenous isotropic-Mach's
Cosmology: Albert Einstein
"Cosmology Considerations on the General Theory of Relativity"
Albert Einstein (1917) [see Cosmological Constant, eds. Bernstein & Feinberg]
Copernican Principle give way to Cosmological Principle
Model the universe as spatially homogenous, isotropic
Mach's influence on relativity and inertia
No inertia relative to spacetime, but inertia of masses relative to one another
Observation and experiment
Account for the small kinetic motions of stars and nebulae
First mathematical model of the universe in general relativity.
Einstein general
Search Options
History
Replace
Edit Content
1724
Replace:
Albert Einstein
1726
With:
A. Einstein
1728
Edit Options:
Replace In Selected Results
Replace In All Results
1729
Target Options
Paragraph/Slide
Header
Footer
Table
All
Matching Documents: 2
Matching Paragraphs: 2
1727
Update Content
Page: 1 of 1
Words: 104/104
43%

Figure 17C

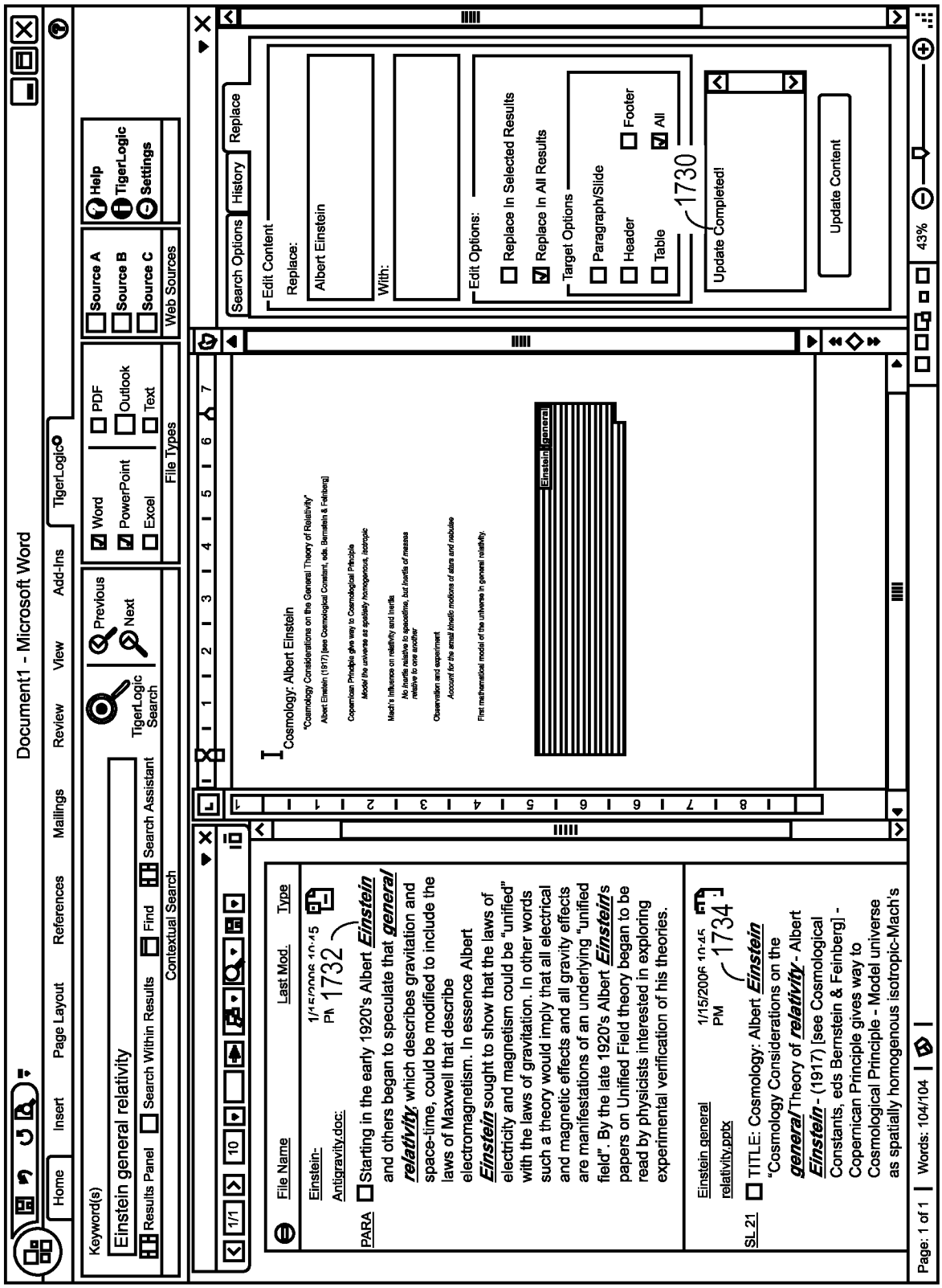

Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Add-Ins
TigerLogic
Keyword(s)
Einstein general relativity
Results Panel
Search Within Results
Find
Search Assistant
Contextual Search
TigerLogic Search
Previous
Next
Word
PowerPoint
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
Search Options
History
Replace
Edit Content
Replace:
Albert Einstein
With:
Edit Options:
Replace In Selected Results
Replace In All Results
Target Options
Paragraph/Slide
Header
Table
Footer
All
1730
Update Completed!
Update Content
Cosmology: Albert Einstein
"Cosmology Considerations on the General Theory of Relativity"
Albert Einstein (1917) [see Cosmological Constant, eds. Bernstein & Feinberg]
Copernican Principle gives way to Cosmological Principle
Model the universe as spatially homogenous, isotropic
Mach's influence on relativity and inertia
No inertia relative to spacetime, but inertia of masses relative to one another
Observation and experiment
Account for the small kinetic motions of stars and nebulae
First mathematical model of the universe in general relativity
Einstein general
File Name
Last Mod.
Type
Einstein-Antigravity.doc
1/15/2006 10:45 PM
1732
PARA
Starting in the early 1920's Albert Einstein and others began to speculate that general relativity which describes gravitation and space-time, could be modified to include the laws of Maxwell that describe electromagnetism. In essence Albert Einstein sought to show that the laws of electricity and magnetism could be "unified" with the laws of gravitation. In other words such a theory would imply that all electrical and magnetic effects and all gravity effects are manifestations of an underlying "unified field". By the late 1920's Albert Einstein's papers on Unified Field theory began to be read by physicists interested in exploring experimental verification of his theories.
Einstein general relativity.pptx
1/15/2006 10:45 PM
1734
SL 21
TITLE: Cosmology: Albert Einstein "Cosmology Considerations on the general Theory of relativity - Albert Einstein - (1917) [see Cosmological Constants, eds Bernstein & Feinberg] - Copernican Principle gives way to Cosmological Principle - Model universe as spatially homogenous isotropic-Mach's
Page: 1 of 1
Words: 104/104
43%

Figure 17D

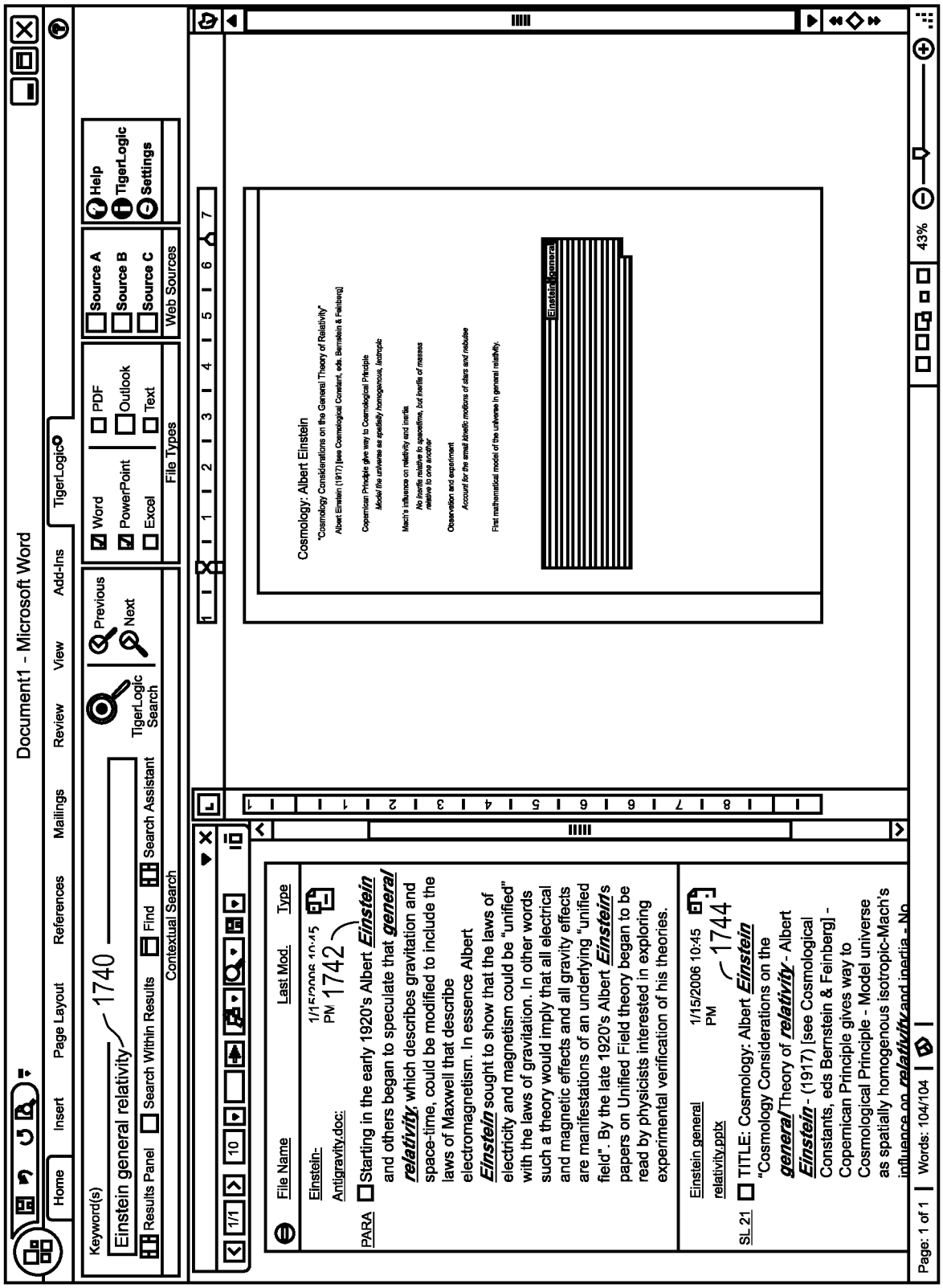

Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Add-Ins
TigerLogic
Keyword(s)
Einstein general relativity
1740
Results Panel
Search Within Results
Find
Search Assistant
TigerLogic Search
Previous
Next
Contextual Search
Word
PowerPoint
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
Cosmology: Albert Einstein
"Cosmology Considerations on the General Theory of Relativity"
Albert Einstein (1917) [see Cosmological Constant, eds. Bernstein & Feinberg]
Copernican Principle gives way to Cosmological Principle
Model the universe as spatially homogenous, isotropic
Mach's influence on relativity and inertia
No inertia relative to spaceltime, but inertia of masses relative to one another
Observation and experiment
Account for the small kinetic motions of stars and nebulae
First mathematical model of the universe in general relativity.
Einstein, General
File Name
Last Mod.
Type
Einstein-Antigravity.doc:
1/15/2006 10:45 PM
1742
PARA
Starting in the early 1920's Albert Einstein and others began to speculate that general relativity, which describes gravitation and space-time, could be modified to include the laws of Maxwell that describe electromagnetism. In essence Albert Einstein sought to show that the laws of electricity and magnetism could be "unified" with the laws of gravitation. In other words such a theory would imply that all electrical and magnetic effects and all gravity effects are manifestations of an underlying "unified field". By the late 1920's Albert Einstein's papers on Unified Field theory began to be read by physicists interested in exploring experimental verification of his theories.
Einstein general relativity.pptx
1/15/2006 10:45 PM
1744
SL 21
TITLE: Cosmology: Albert Einstein "Cosmology Considerations on the general Theory of relativity - Albert Einstein - (1917) [see Cosmological Constants, eds Bernstein & Feinberg] - Copernican Principle gives way to Cosmological Principle - Model universe as spatially homogenous isotropic-Mach's influence on relativity and inertia - No
Page: 1 of 1
Words: 104/104
43%

Figure 17E

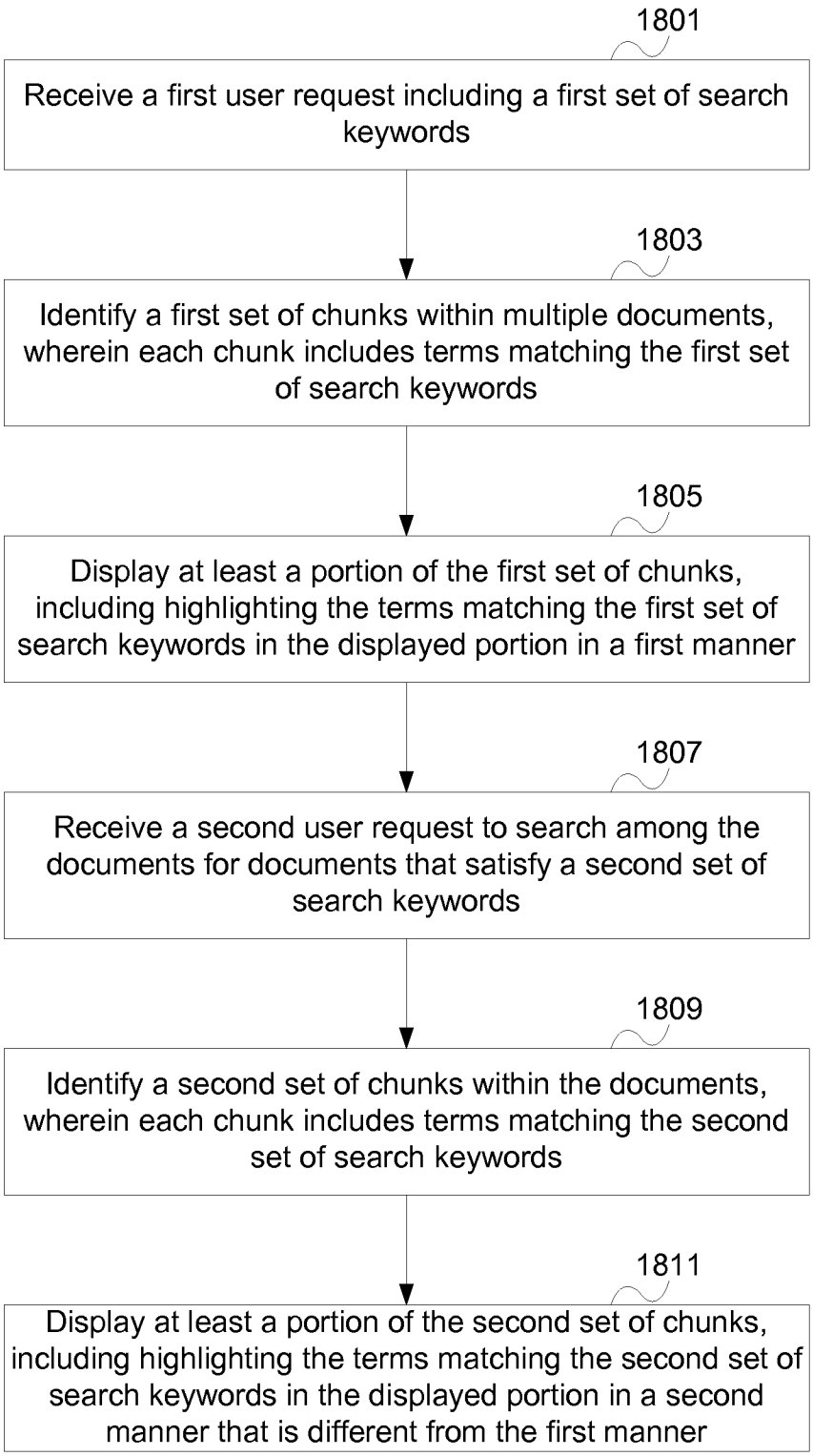

1801
Receive a first user request including a first set of search keywords
1803
Identify a first set of chunks within multiple documents, wherein each chunk includes terms matching the first set of search keywords
1805
Display at least a portion of the first set of chunks, including highlighting the terms matching the first set of search keywords in the displayed portion in a first manner
1807
Receive a second user request to search among the documents for documents that satisfy a second set of search keywords
1809
Identify a second set of chunks within the documents, wherein each chunk includes terms matching the second set of search keywords
1811
Display at least a portion of the second set of chunks, including highlighting the terms matching the second set of search keywords in the displayed portion in a second manner that is different from the first manner

Figure 18A

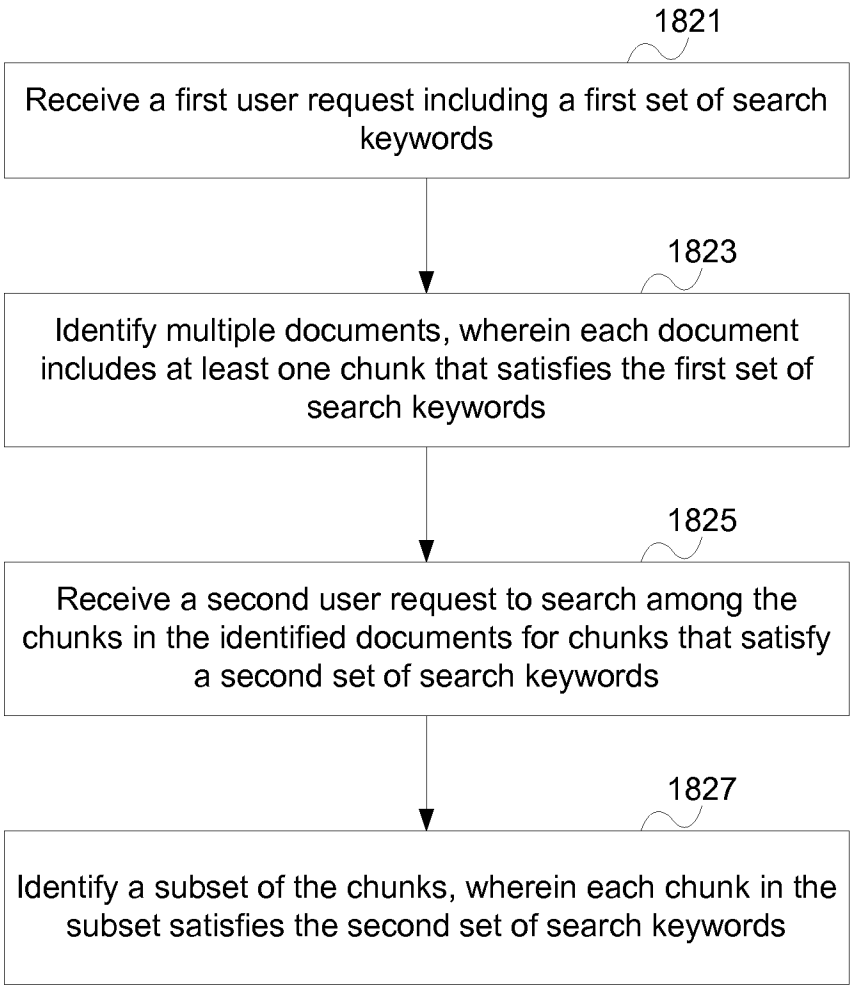

1821
Receive a first user request including a first set of search keywords
1823
Identify multiple documents, wherein each document includes at least one chunk that satisfies the first set of search keywords
1825
Receive a second user request to search among the chunks in the identified documents for chunks that satisfy a second set of search keywords
1827
Identify a subset of the chunks, wherein each chunk in the subset satisfies the second set of search keywords

Figure 18B

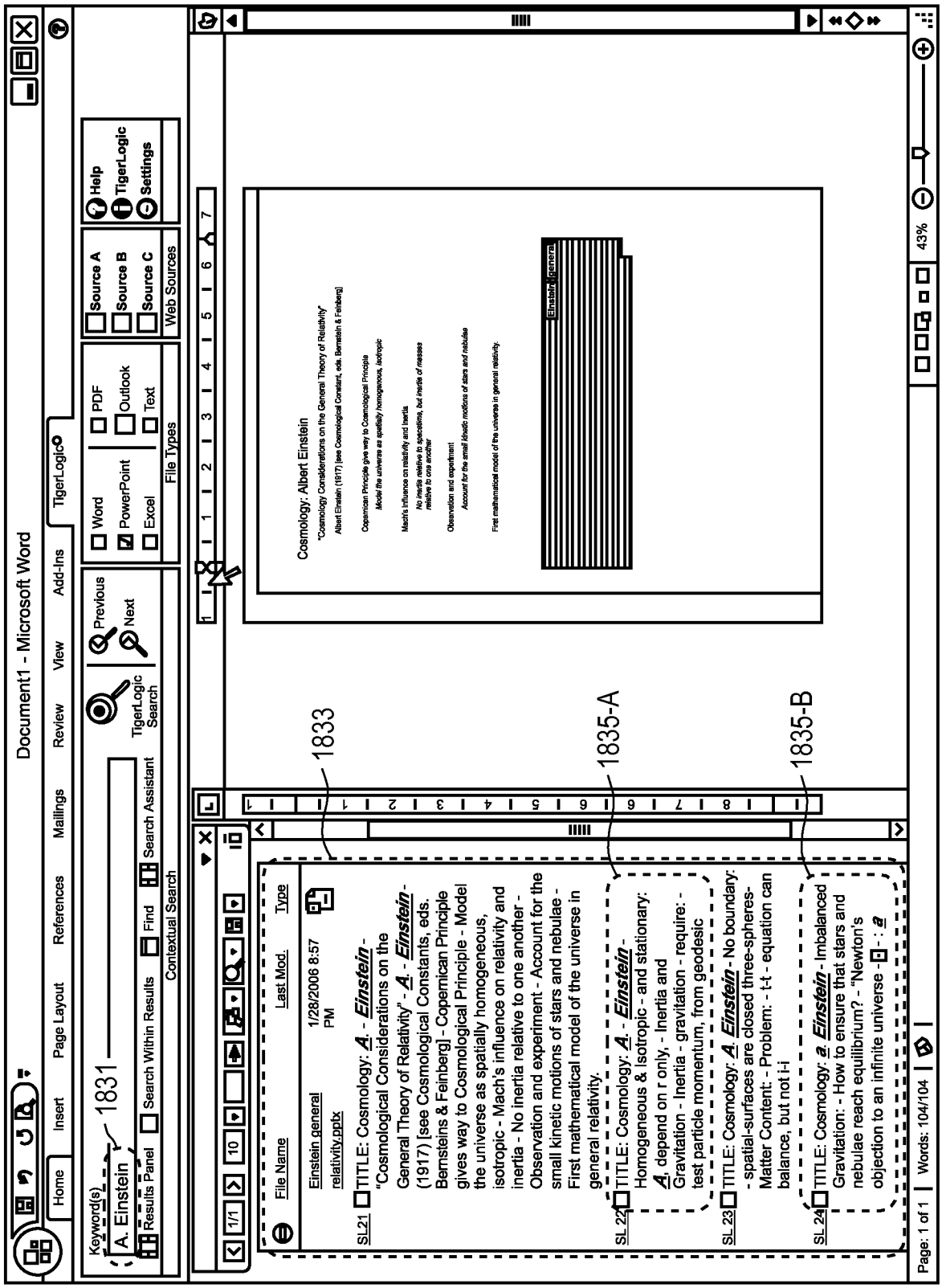

Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Add-Ins
TigerLogic
Keyword(s)
A. Einstein
1831
Results Panel
Search Within Results
Find
Search Assistant
Contextual Search
TigerLogic Search
Previous
Next
Word
PowerPoint
Excel
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
Cosmology: Albert Einstein
"Cosmology Considerations on the General Theory of Relativity"
Albert Einstein (1917) [see Cosmological Constant, eds. Bernstein & Feinberg]
Copernican Principle give way to Cosmological Principle
Model the universe as spatially homogeneous, isotropic
Mach's influence on relativity and inertia
No inertia relative to spacetime, but inertia of masses relative to one another
Observation and experiment
Account for the small kinetic motions of stars and nebulae
First mathematical model of the universe in general relativity.
Einstein general relativity.pptx
1/28/2006 8:57 PM
SL21 TITLE: Cosmology: A. Einstein - "Cosmological Considerations on the General Theory of Relativity" - A. Einstein - (1917) [see Cosmological Constants, eds. Bernsteins & Feinberg] - Copernican Principle gives way to Cosmological Principle - Model the universe as spatially homogeneous, isotropic - Mach's influence on relativity and inertia - No inertia relative to one another - Observation and experiment - Account for the small kinetic motions of stars and nebulae - First mathematical model of the universe in general relativity.
SL 22 TITLE: Cosmology: A. Einstein - Homogeneous & Isotropic - and stationary: A depend on r only, - Inertia and Gravitation - Inertia - gravitation - require: - test particle momentum, from geodesic
SL 23 TITLE: Cosmology: A. Einstein - No boundary: - spatial-surfaces are closed three-spheres- Matter Content: - Problem: - t-t - equation can balance, but not i-i
SL 24 TITLE: Cosmology: a. Einstein - Imbalanced Gravitation: - How to ensure that stars and nebulae reach equilibrium? - "Newton's objection to an infinite universe - : a
1833
1835-A
1835-B
Page: 1 of 1
Words: 104/104
43%

Figure 18C

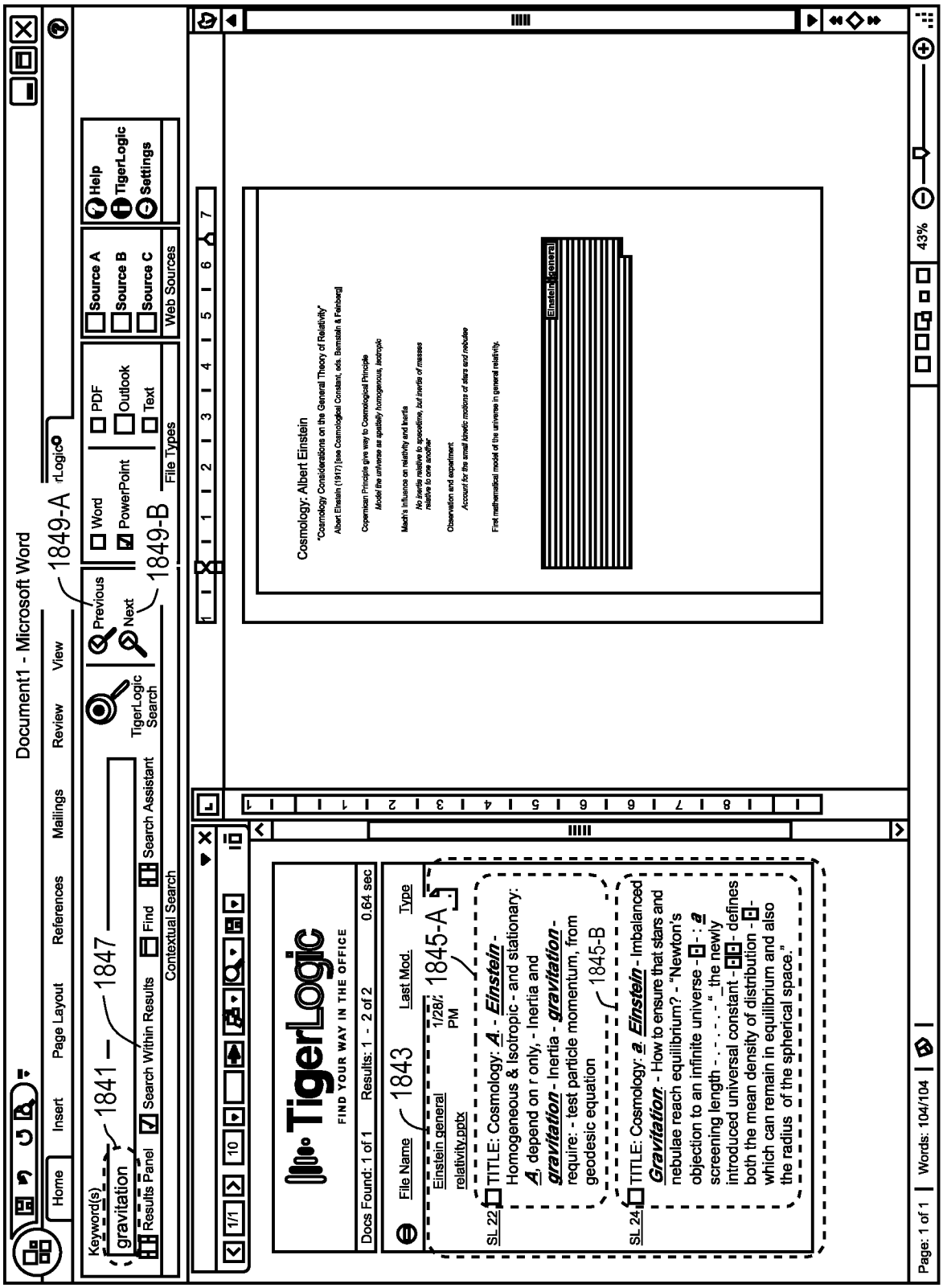

Document1 - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
rLogic
1849-A
Keyword(s)
gravitation
1841
1847
Results Panel
Search Within Results
Find
Search Assistant
TigerLogic Search
Previous
Next
1849-B
Word
PowerPoint
PDF
Outlook
Text
File Types
Source A
Source B
Source C
Web Sources
Help
TigerLogic
Settings
Contextual Search
TigerLogic
FIND YOUR WAY IN THE OFFICE
Docs Found: 1 of 1
Results: 1 - 2 of 2
0.64 sec
File Name
1843
Last Mod.
Type
Einstein general relativity.pptx
1/28/ PM
1845-A
SL 22
TITLE: Cosmology: A - Einstein - Homogeneous & Isotropic - and stationary: A, depend on r only, - Inertia and gravitation - Inertia - gravitation - require: - test particle momentum, from geodesic equation
1845-B
SL 24
TITLE: Cosmology: a Einstein - Imbalanced Gravitation - How to ensure that stars and nebulae reach equilibrium? - "Newton's objection to an infinite universe - : a screening length - . - . - . - " the newly introduced universal constant - defines both the mean density of distribution - which can remain in equilibrium and also the radius of the spherical space."
Cosmology: Albert Einstein
"Cosmology Considerations on the General Theory of Relativity"
Albert Einstein (1917) [see Cosmological Constant, eds. Barnstein & Feinberg]
Copernican Principle give way to Cosmological Principle
Model the universe as spatially homogeneous, isotropic
Mach's influence on relativity and Inertia
No inertia relative to spacetime, but inertia of masses relative to one another
Observation and experiment
Account for the small kinetic motions of stars and nebulae
Find mathematical model of the universe in general relativity.
Einstein (General)
Page: 1 of 1
Words: 104/104
43%

Figure 18D

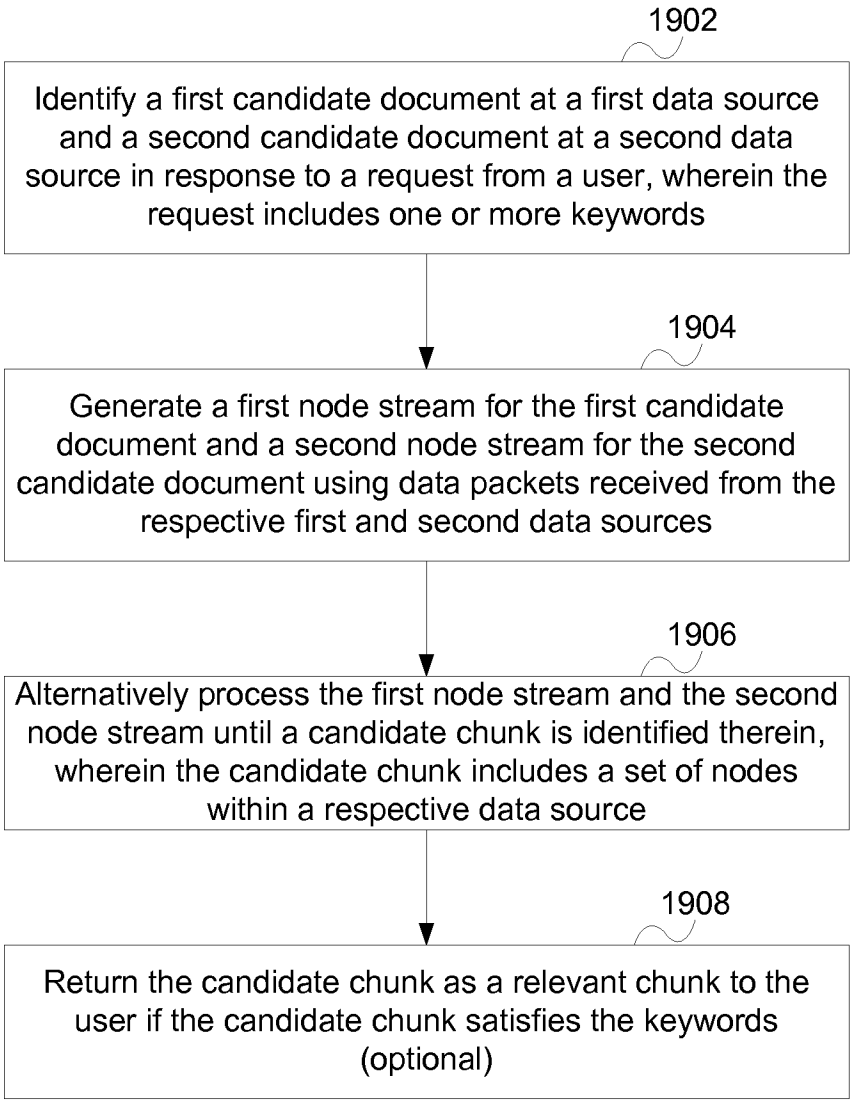

1902
Identify a first candidate document at a first data source and a second candidate document at a second data source in response to a request from a user, wherein the request includes one or more keywords
1904
Generate a first node stream for the first candidate document and a second node stream for the second candidate document using data packets received from the respective first and second data sources
1906
Alternatively process the first node stream and the second node stream until a candidate chunk is identified therein, wherein the candidate chunk includes a set of nodes within a respective data source
1908
Return the candidate chunk as a relevant chunk to the user if the candidate chunk satisfies the keywords (optional)

Figure 19

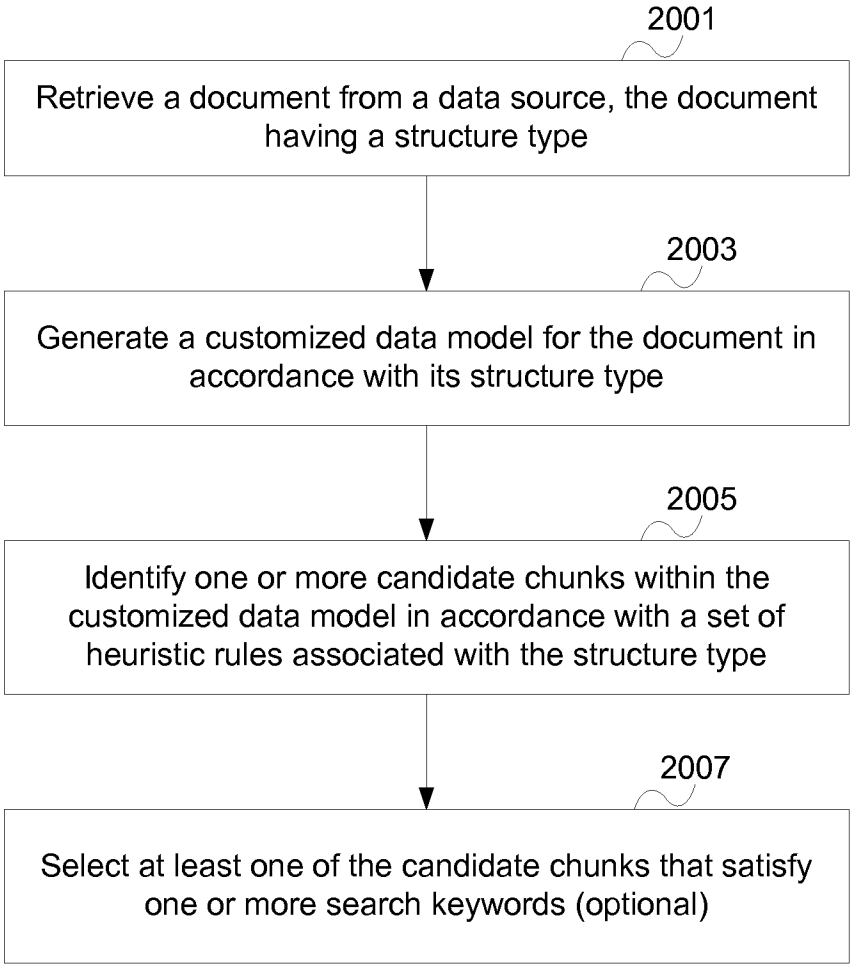

2001
Retrieve a document from a data source, the document having a structure type
2003
Generate a customized data model for the document in accordance with its structure type
2005
Identify one or more candidate chunks within the customized data model in accordance with a set of heuristic rules associated with the structure type
2007
Select at least one of the candidate chunks that satisfy one or more search keywords (optional)

Figure 20

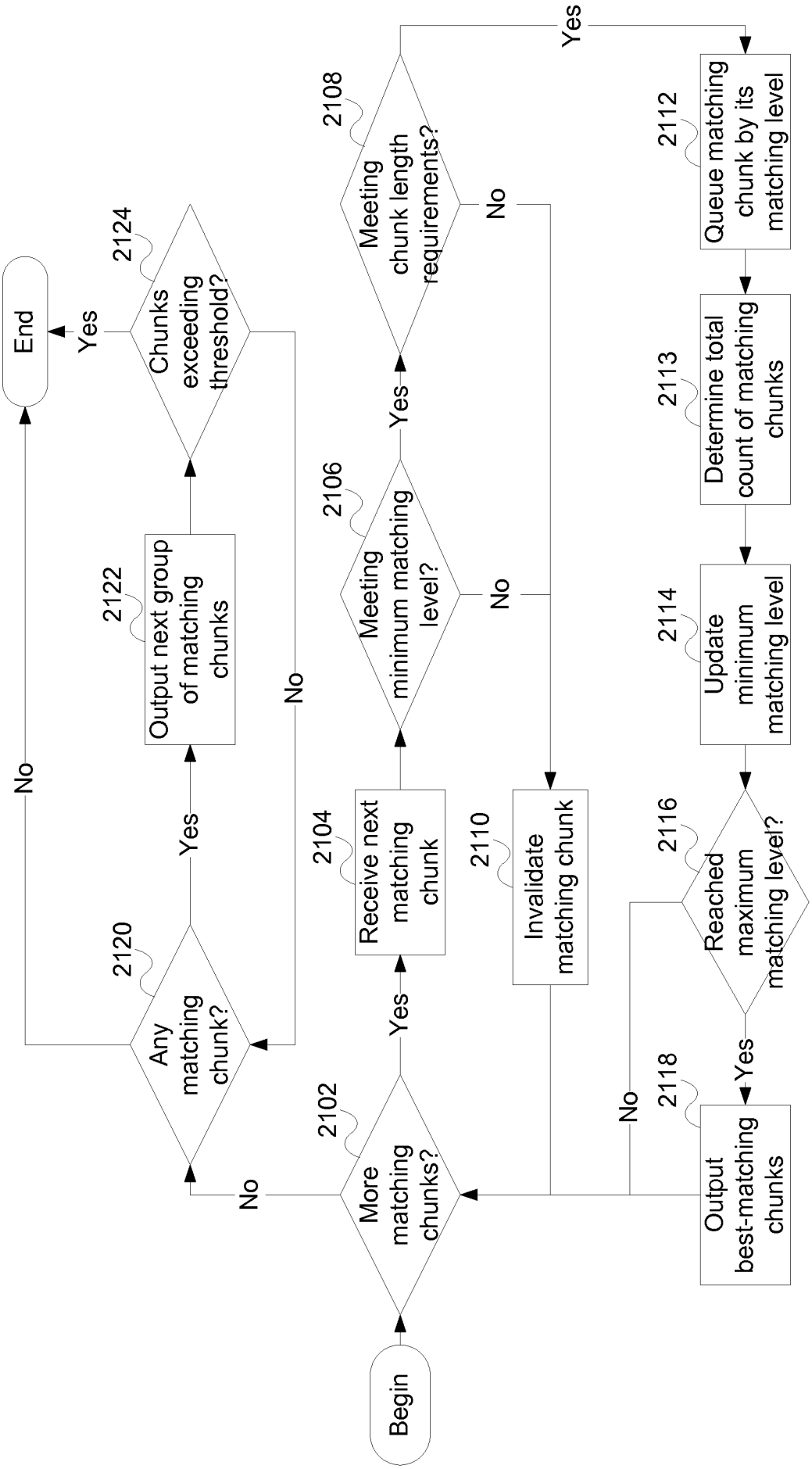

Begin
More matching chunks?
2102
Yes
No
Receive next matching chunk
2104
Meeting minimum matching level?
2106
Yes
No
Meeting chunk length requirements?
2108
Yes
No
Invalidate matching chunk
2110
Queue matching chunk by its matching level
2112
Determine total count of matching chunks
2113
Update minimum matching level
2114
Reached maximum matching level?
2116
Yes
No
Output best-matching chunks
2118
Any matching chunk?
2120
Yes
No
Output next group of matching chunks
2122
Chunks exceeding threshold?
2124
Yes
No
End

Figure 21A


2130
2131
2133
2135
2137
2139
<html>
<body>

<h1 cid="2">An introduction to the Best Match algorithm.</h1>
Herein you'll find all of the scintillating bits of the underlying algorithm.

And some
examples of the algorithms

</body>
</html>

Figure 21B

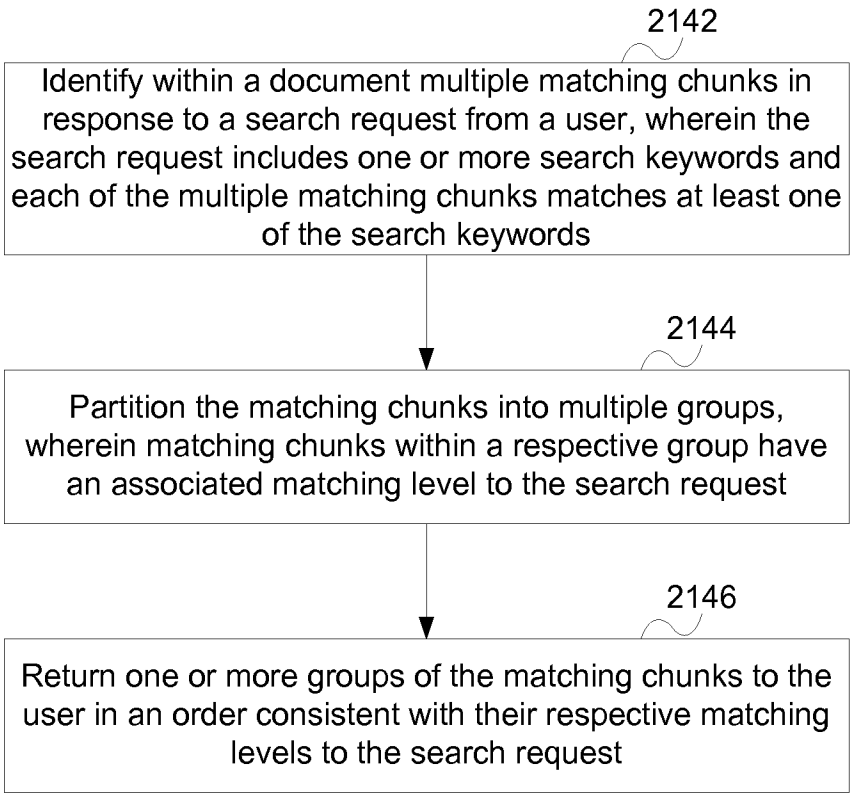

2142
Identify within a document multiple matching chunks in response to a search request from a user, wherein the search request includes one or more search keywords and each of the multiple matching chunks matches at least one of the search keywords
2144
Partition the matching chunks into multiple groups, wherein matching chunks within a respective group have an associated matching level to the search request
2146
Return one or more groups of the matching chunks to the user in an order consistent with their respective matching levels to the search request

Figure 21C

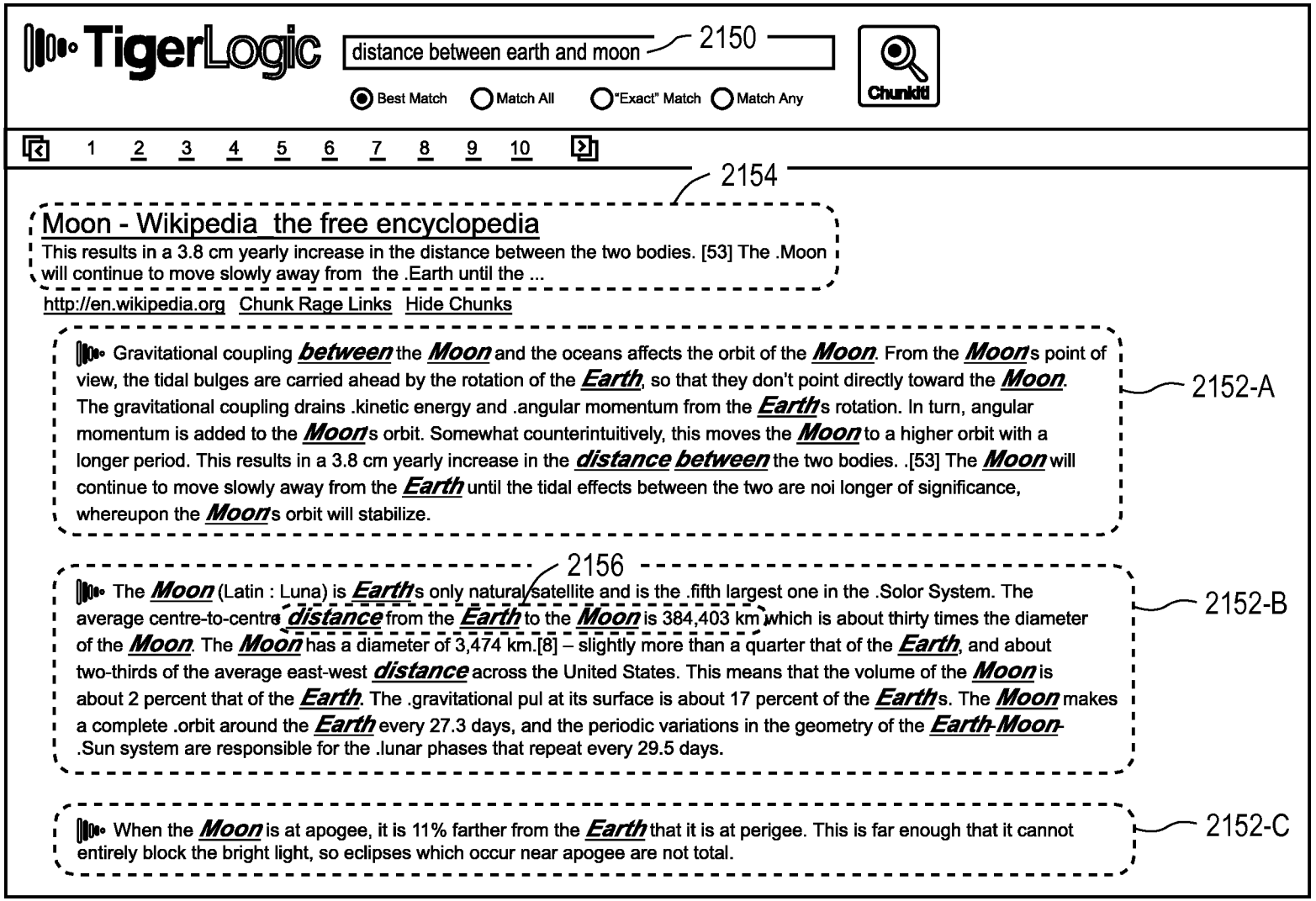

TigerLogic
distance between earth and moon
2150
Best Match
Match All
"Exact" Match
Match Any
Chunkit
1 2 3 4 5 6 7 8 9 10
2154
Moon - Wikipedia the free encyclopedia
This results in a 3.8 cm yearly increase in the distance between the two bodies. [53] The .Moon will continue to move slowly away from the .Earth until the ...
http://en.wikipedia.org Chunk Rage Links Hide Chunks
Gravitational coupling between the Moon and the oceans affects the orbit of the Moon. From the Moon's point of view, the tidal bulges are carried ahead by the rotation of the Earth, so that they don't point directly toward the Moon. The gravitational coupling drains .kinetic energy and .angular momentum from the Earth's rotation. In turn, angular momentum is added to the Moon's orbit. Somewhat counterintuitively, this moves the Moon to a higher orbit with a longer period. This results in a 3.8 cm yearly increase in the distance between the two bodies. .[53] The Moon will continue to move slowly away from the Earth until the tidal effects between the two are noi longer of significance, whereupon the Moon's orbit will stabilize.
2152-A
2156
The Moon (Latin : Luna) is Earth's only natural satellite and is the .fifth largest one in the .Solor System. The average centre-to-centre distance from the Earth to the Moon is 384,403 km, which is about thirty times the diameter of the Moon. The Moon has a diameter of 3,474 km.[8] – slightly more than a quarter that of the Earth, and about two-thirds of the average east-west distance across the United States. This means that the volume of the Moon is about 2 percent that of the Earth. The .gravitational pul at its surface is about 17 percent of the Earth's. The Moon makes a complete .orbit around the Earth every 27.3 days, and the periodic variations in the geometry of the Earth-Moon-.Sun system are responsible for the .lunar phases that repeat every 29.5 days.
2152-B
When the Moon is at apogee, it is 11% farther from the Earth that it is at perigee. This is far enough that it cannot entirely block the bright light, so eclipses which occur near apogee are not total.
2152-C

Figure 21D

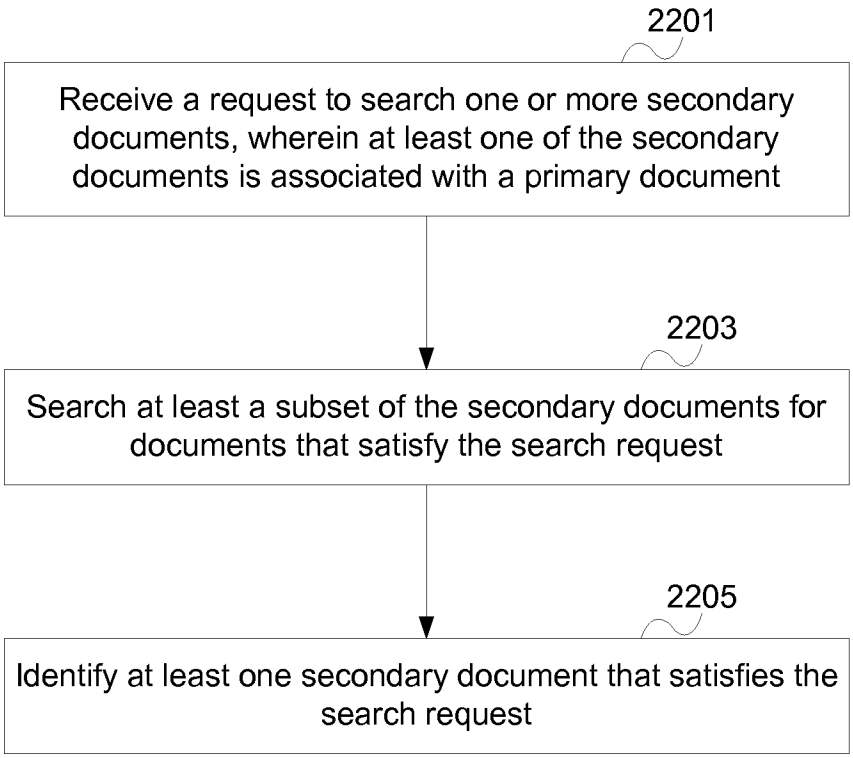

2201
Receive a request to search one or more secondary documents, wherein at least one of the secondary documents is associated with a primary document
2203
Search at least a subset of the secondary documents for documents that satisfy the search request
2205
Identify at least one secondary document that satisfies the search request

Figure 22A

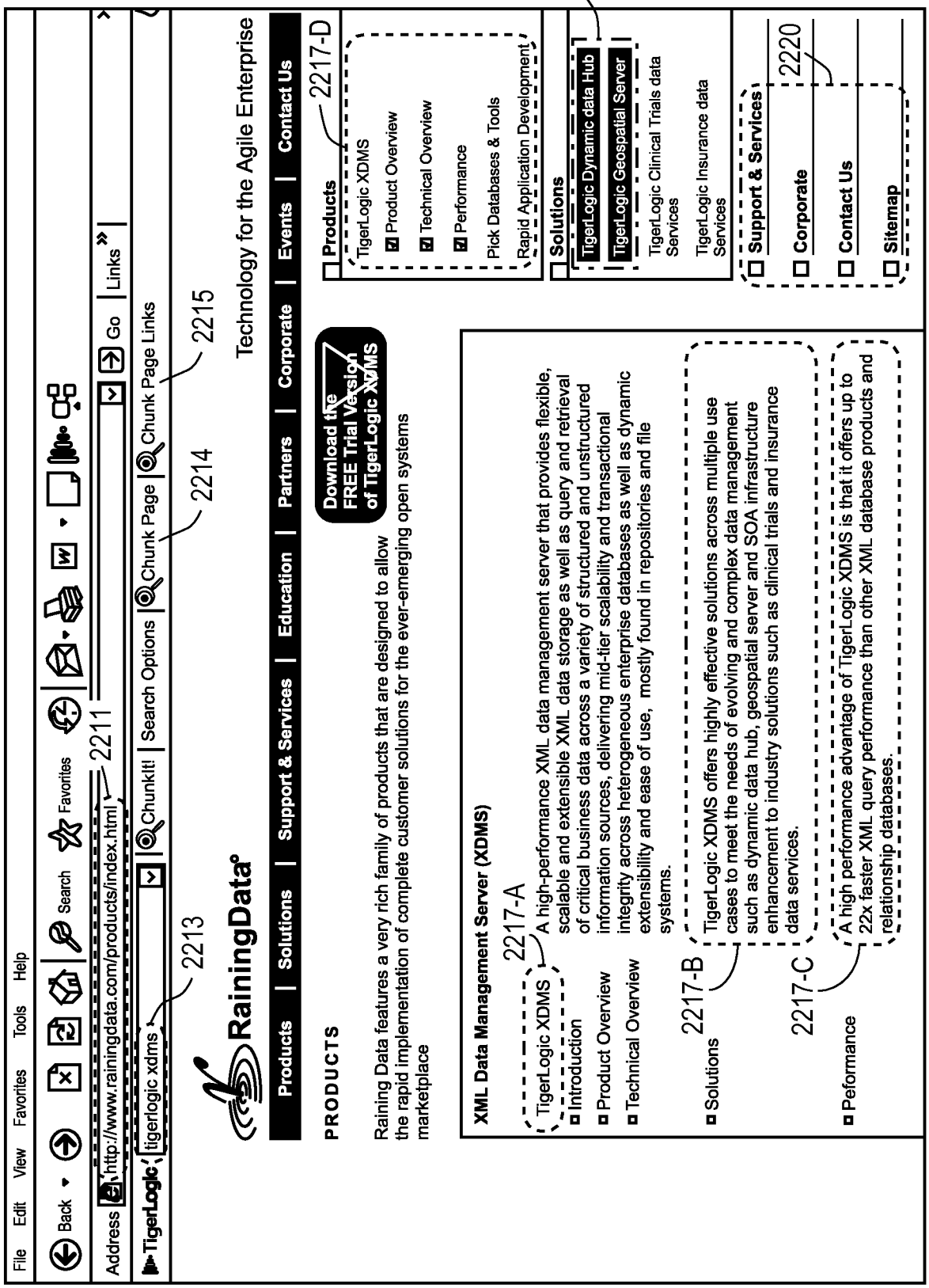

File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.rainingdata.com/products/index.html
2211
Go Links
TigerLogic tigerlogic xdms Chunkit! Search Options Chunk Page Chunk Page Links
2213
2214
2215
RainingData
Technology for the Agile Enterprise
Products | Solutions | Support & Services | Education | Partners | Corporate | Events | Contact Us
PRODUCTS
Raining Data features a very rich family of products that are designed to allow the rapid implementation of complete customer solutions for the ever-emerging open systems marketplace
Download the FREE Trial Version of TigerLogic XDMS
XML Data Management Server (XDMS)
TigerLogic XDMS
2217-A
A high-performance XML data management server that provides flexible, scalable and extensible XML data storage as well as query and retrieval of critical business data across a variety of structured and unstructured information sources, delivering mid-tier scalability and transactional integrity across heterogeneous enterprise databases as well as dynamic extensibility and ease of use, mostly found in repositories and file systems.
Introduction
Product Overview
Technical Overview
Solutions
2217-B
TigerLogic XDMS offers highly effective solutions across multiple use cases to meet the needs of evolving and complex data management such as dynamic data hub, geospatial server and SOA infrastructure enhancement to industry solutions such as clinical trials and insurance data services.
Peformance
2217-C
A high performance advantage of TigerLogic XDMS is that it offers up to 22x faster XML query performance than other XML database products and relationship databases.
Products
2217-D
TigerLogic XDMS
Product Overview
Technical Overview
Performance
Pick Databases & Tools
Rapid Application Development
Solutions
2219
TigerLogic Dynamic data Hub
TigerLogic Geospatial Server
TigerLogic Clinical Trials data Services
TigerLogic Insurance data Services
Support & Services
Corporate
2220
Contact Us
Sitemap

Figure 22B

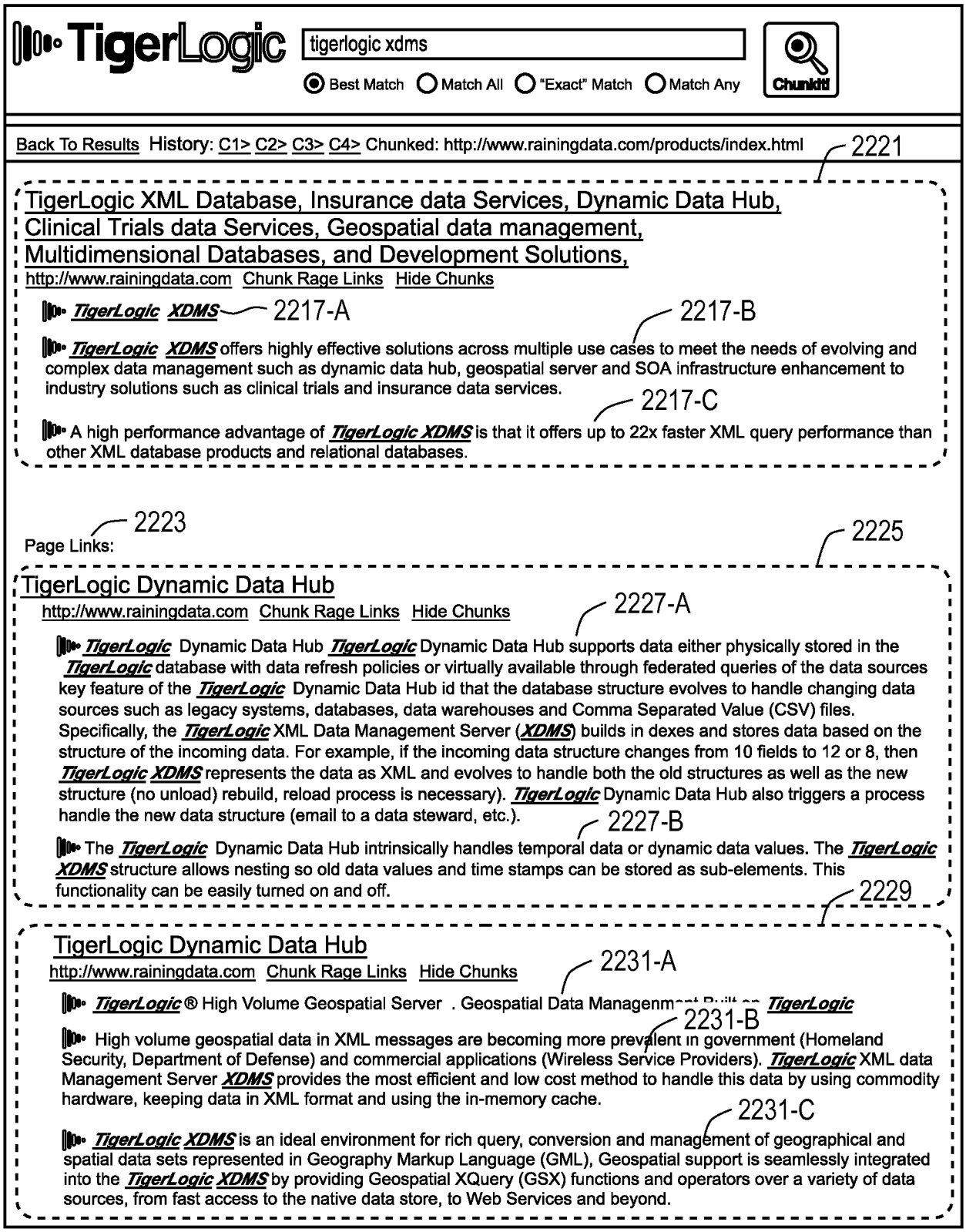

TigerLogic
tigerlogic xdms
Best Match
Match All
"Exact" Match
Match Any
Chunkit!
Back To Results History: C1> C2> C3> C4> Chunked: http://www.rainingdata.com/products/index.html
2221
TigerLogic XML Database, Insurance data Services, Dynamic Data Hub, Clinical Trials data Services, Geospatial data management, Multidimensional Databases, and Development Solutions,
http://www.rainingdata.com Chunk Rage Links Hide Chunks
TigerLogic XDMS
2217-A
2217-B
TigerLogic XDMS offers highly effective solutions across multiple use cases to meet the needs of evolving and complex data management such as dynamic data hub, geospatial server and SOA infrastructure enhancement to industry solutions such as clinical trials and insurance data services.
2217-C
A high performance advantage of TigerLogic XDMS is that it offers up to 22x faster XML query performance than other XML database products and relational databases.
2223
Page Links:
2225
TigerLogic Dynamic Data Hub
http://www.rainingdata.com Chunk Rage Links Hide Chunks
2227-A
TigerLogic Dynamic Data Hub TigerLogic Dynamic Data Hub supports data either physically stored in the TigerLogic database with data refresh policies or virtually available through federated queries of the data sources key feature of the TigerLogic Dynamic Data Hub id that the database structure evolves to handle changing data sources such as legacy systems, databases, data warehouses and Comma Separated Value (CSV) files. Specifically, the TigerLogic XML Data Management Server (XDMS) builds in dexes and stores data based on the structure of the incoming data. For example, if the incoming data structure changes from 10 fields to 12 or 8, then TigerLogic XDMS represents the data as XML and evolves to handle both the old structures as well as the new structure (no unload) rebuild, reload process is necessary). TigerLogic Dynamic Data Hub also triggers a process handle the new data structure (email to a data steward, etc.).
2227-B
The TigerLogic Dynamic Data Hub intrinsically handles temporal data or dynamic data values. The TigerLogic XDMS structure allows nesting so old data values and time stamps can be stored as sub-elements. This functionality can be easily turned on and off.
2229
TigerLogic Dynamic Data Hub
http://www.rainingdata.com Chunk Rage Links Hide Chunks
2231-A
TigerLogic® High Volume Geospatial Server . Geospatial Data Managenm
TigerLogic
2231-B
High volume geospatial data in XML messages are becoming more prevalent in government (Homeland Security, Department of Defense) and commercial applications (Wireless Service Providers). TigerLogic XML data Management Server XDMS provides the most efficient and low cost method to handle this data by using commodity hardware, keeping data in XML format and using the in-memory cache.
2231-C
TigerLogic XDMS is an ideal environment for rich query, conversion and management of geographical and spatial data sets represented in Geography Markup Language (GML), Geospatial support is seamlessly integrated into the TigerLogic XDMS by providing Geospatial XQuery (GSX) functions and operators over a variety of data sources, from fast access to the native data store, to Web Services and beyond.

Figure 22C

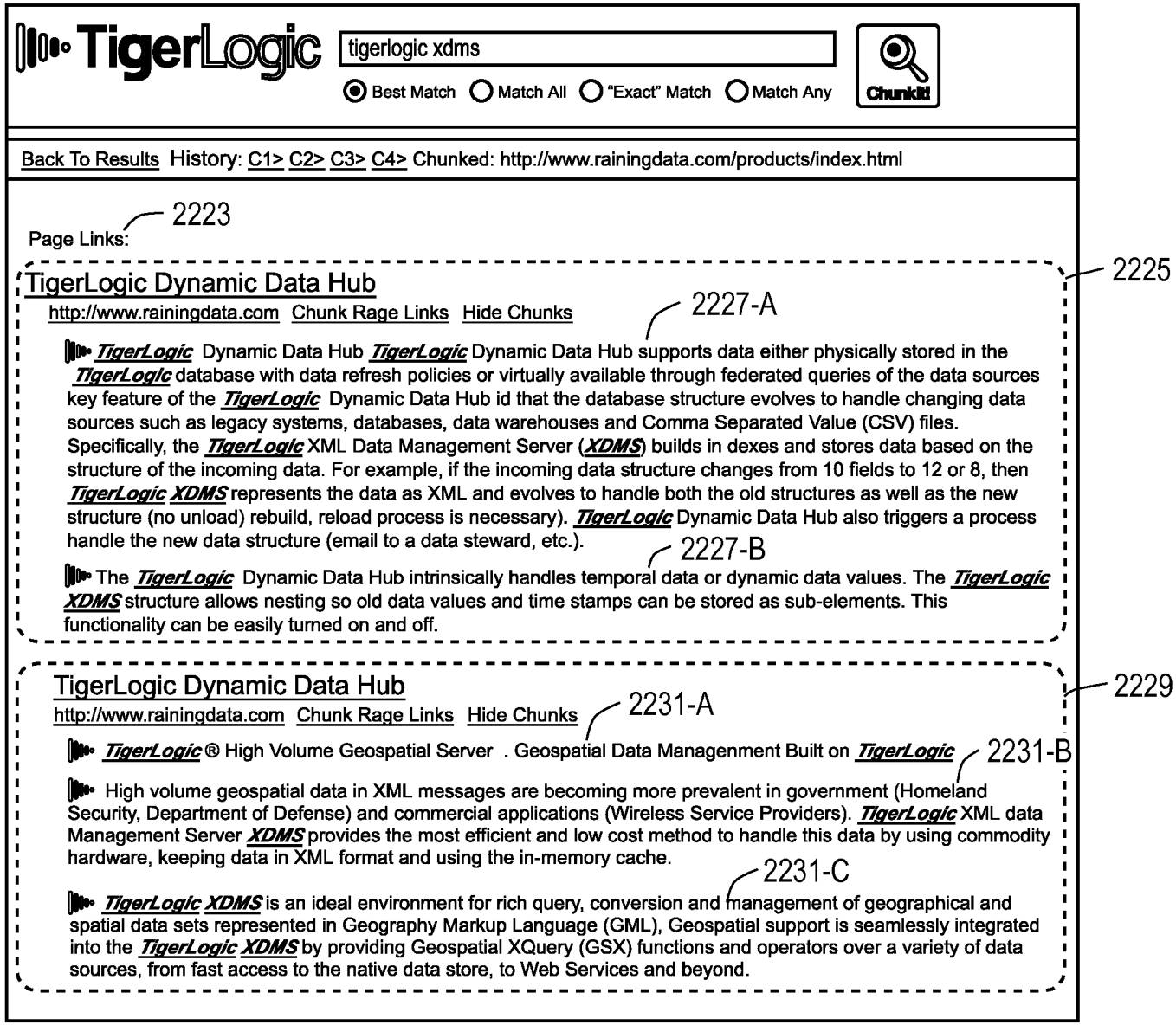

TigerLogic
tigerlogic xdms
Best Match
Match All
"Exact" Match
Match Any
ChunkIt!
Back To Results History: C1> C2> C3> C4> Chunked: http://www.rainingdata.com/products/index.html
Page Links:
2223
2225
TigerLogic Dynamic Data Hub
http://www.rainingdata.com Chunk Rage Links Hide Chunks
2227-A
TigerLogic Dynamic Data Hub TigerLogic Dynamic Data Hub supports data either physically stored in the TigerLogic database with data refresh policies or virtually available through federated queries of the data sources key feature of the TigerLogic Dynamic Data Hub id that the database structure evolves to handle changing data sources such as legacy systems, databases, data warehouses and Comma Separated Value (CSV) files. Specifically, the TigerLogic XML Data Management Server (XDMS) builds in dexes and stores data based on the structure of the incoming data. For example, if the incoming data structure changes from 10 fields to 12 or 8, then TigerLogic XDMS represents the data as XML and evolves to handle both the old structures as well as the new structure (no unload) rebuild, reload process is necessary). TigerLogic Dynamic Data Hub also triggers a process handle the new data structure (email to a data steward, etc.).
2227-B
The TigerLogic Dynamic Data Hub intrinsically handles temporal data or dynamic data values. The TigerLogic XDMS structure allows nesting so old data values and time stamps can be stored as sub-elements. This functionality can be easily turned on and off.
2229
TigerLogic Dynamic Data Hub
http://www.rainingdata.com Chunk Rage Links Hide Chunks
2231-A
TigerLogic® High Volume Geospatial Server . Geospatial Data Managenment Built on TigerLogic
2231-B
High volume geospatial data in XML messages are becoming more prevalent in government (Homeland Security, Department of Defense) and commercial applications (Wireless Service Providers). TigerLogic XML data Management Server XDMS provides the most efficient and low cost method to handle this data by using commodity hardware, keeping data in XML format and using the in-memory cache.
2231-C
TigerLogic XDMS is an ideal environment for rich query, conversion and management of geographical and spatial data sets represented in Geography Markup Language (GML), Geospatial support is seamlessly integrated into the TigerLogic XDMS by providing Geospatial XQuery (GSX) functions and operators over a variety of data sources, from fast access to the native data store, to Web Services and beyond.

Figure 22D

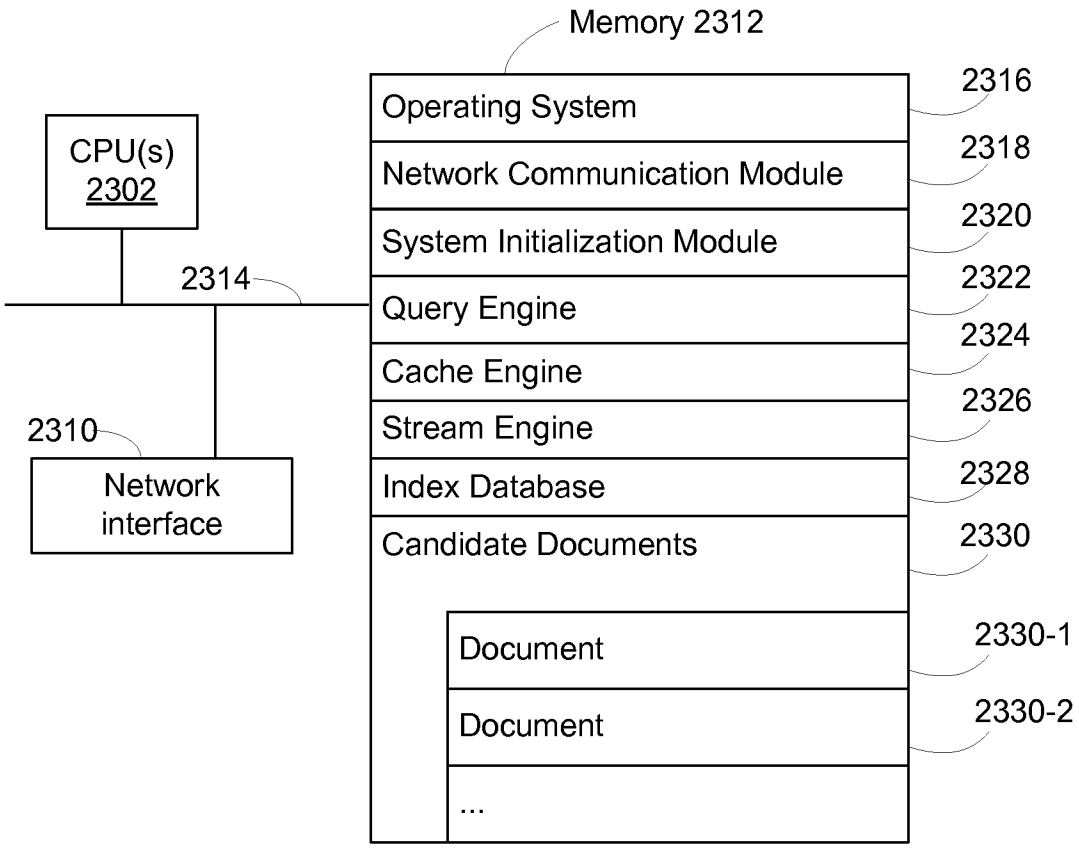

Document Search Server 2300
Memory 2312
Operating System
Network Communication Module
System Initialization Module
Query Engine
Cache Engine
Stream Engine
Index Database
Candidate Documents
Document
Document
...
2316
2318
2320
2322
2324
2326
2328
2330
2330-1
2330-2
CPU(s) 2302
2314
2310
Network interface

Figure 23

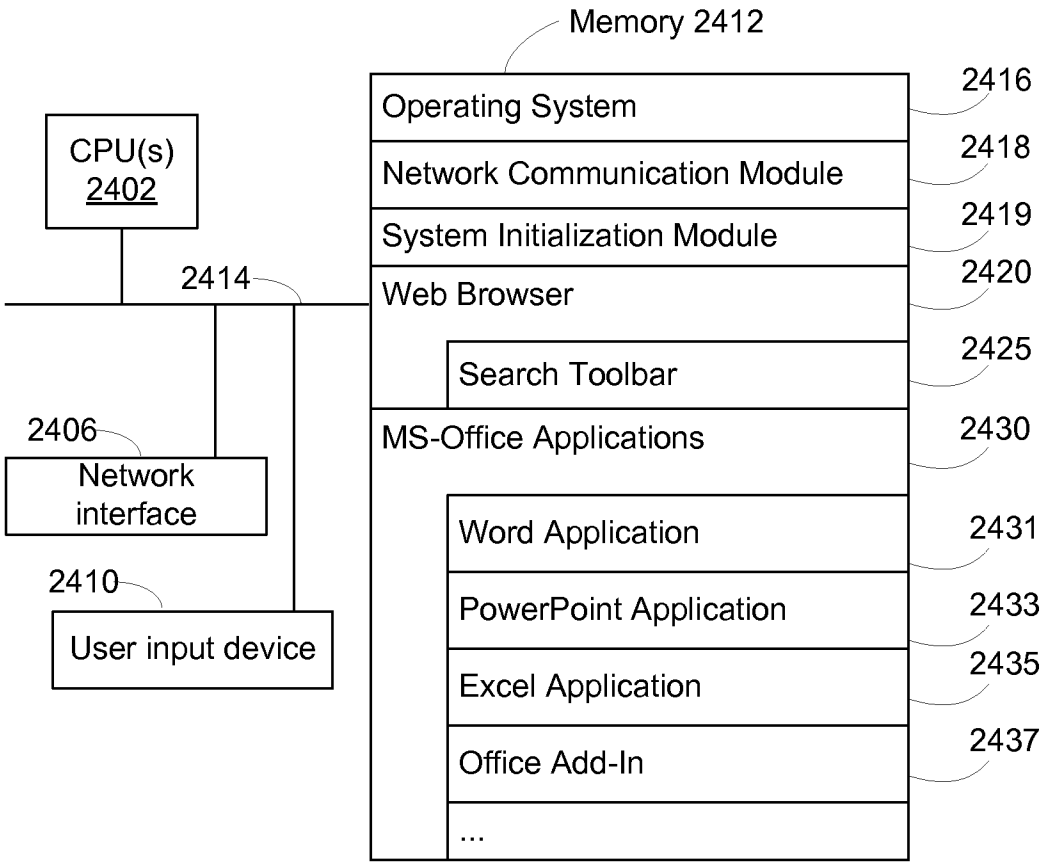

Client Computer 2400
Memory 2412
Operating System
2416
Network Communication Module
2418
System Initialization Module
2419
Web Browser
2420
Search Toolbar
2425
MS-Office Applications
2430
Word Application
2431
PowerPoint Application
2433
Excel Application
2435
Office Add-In
2437
...
CPU(s)
2402
2414
2406
Network interface
2410
User input device

Figure 24

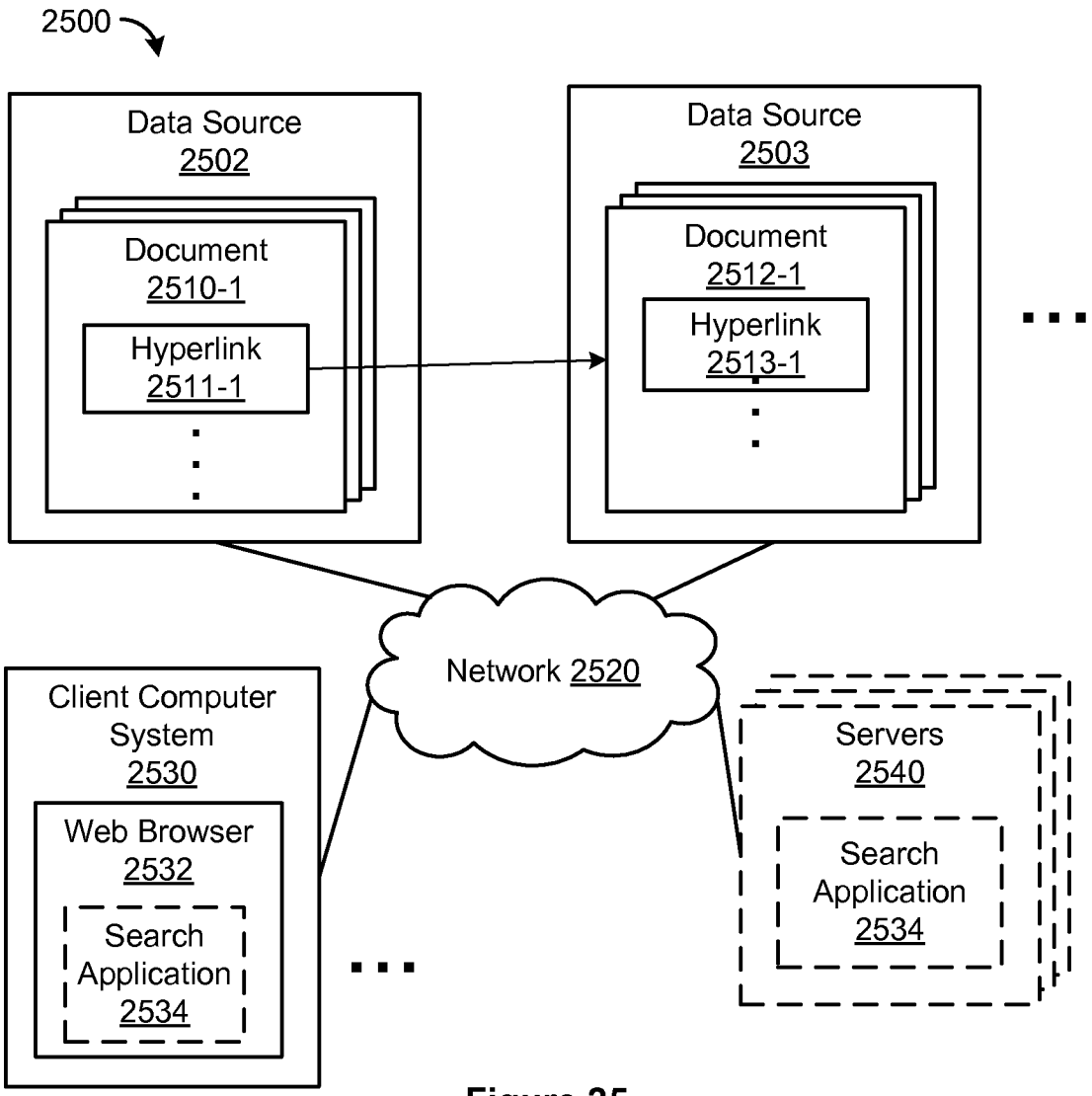



**Figure 25**

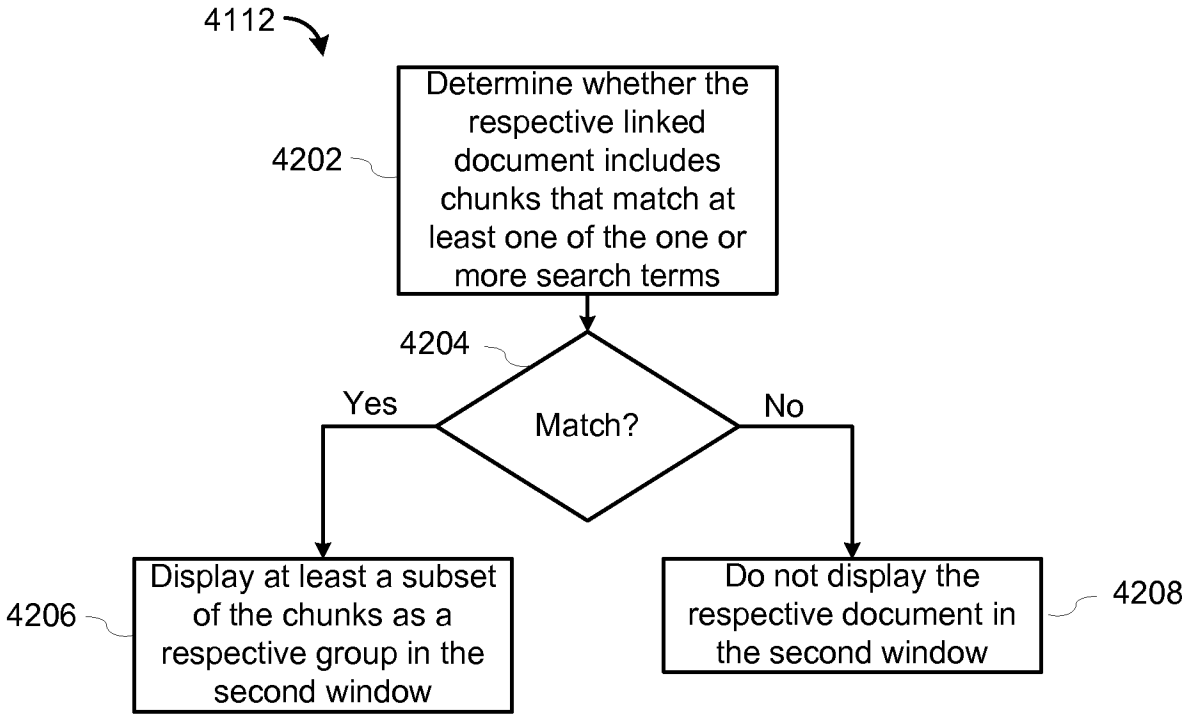



**Figure 42**

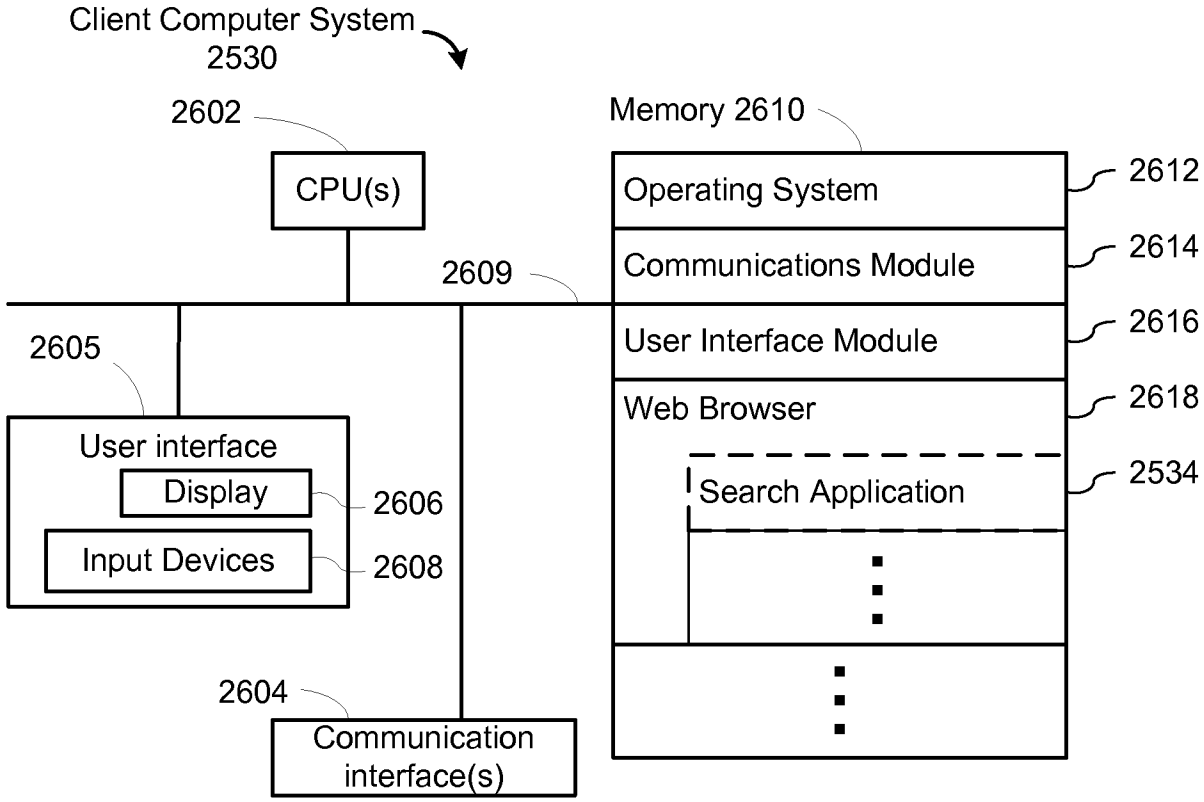



**Figure 26**

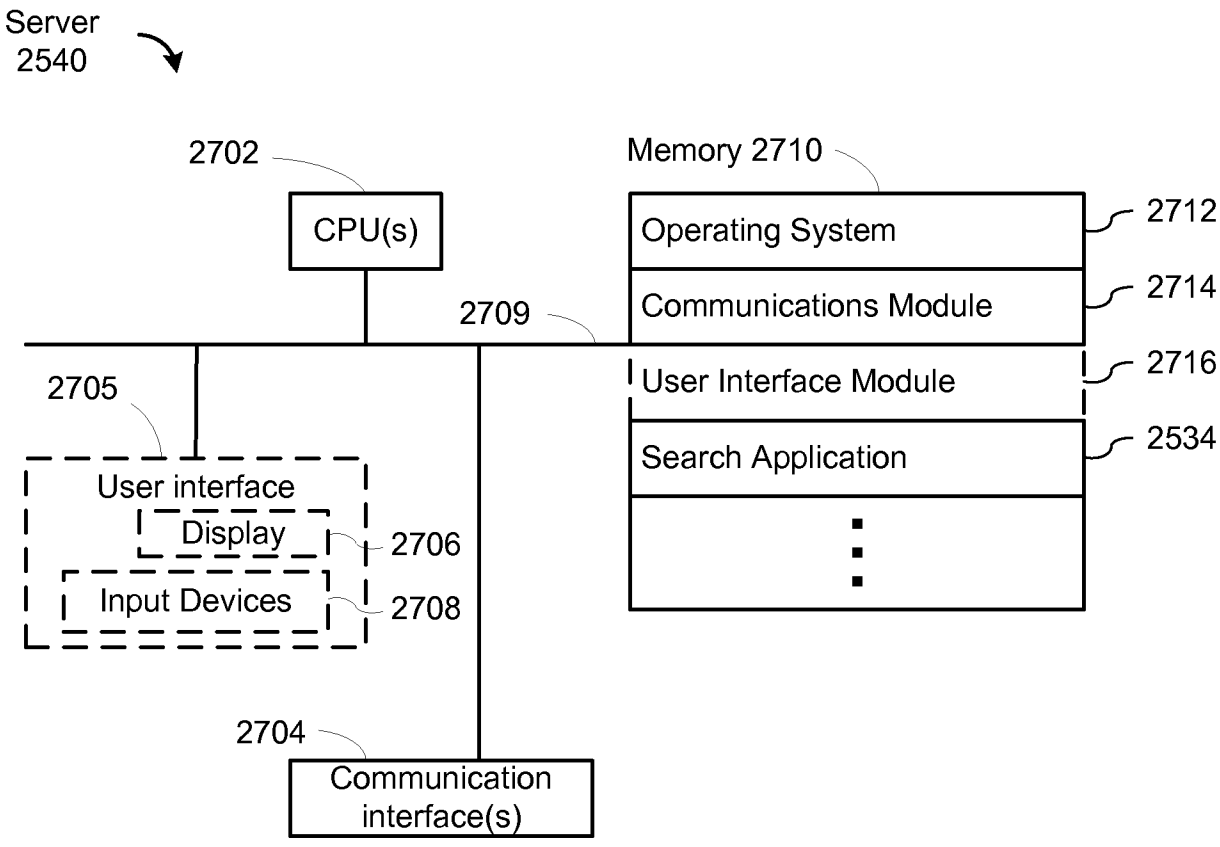



**Figure 27**

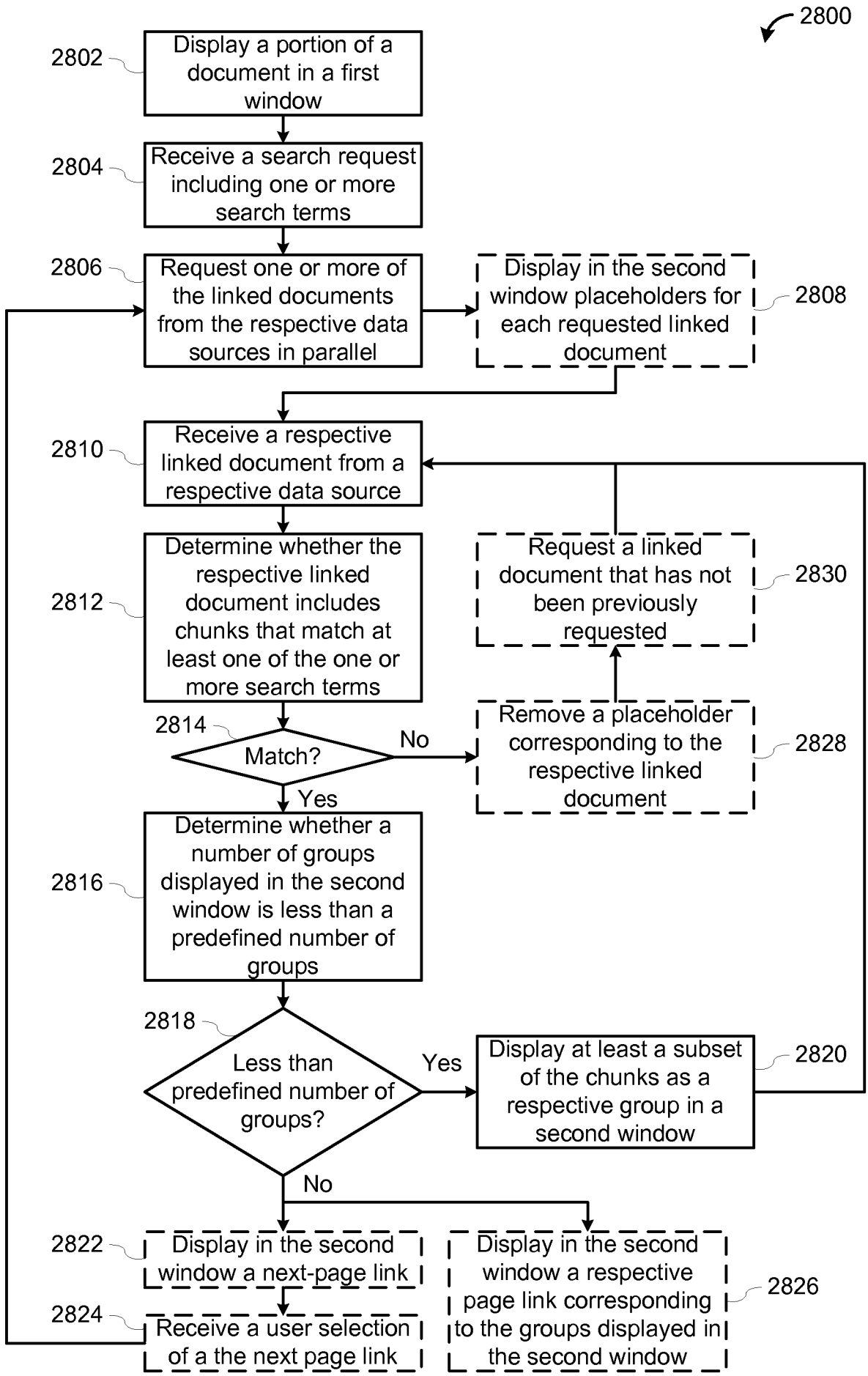

2800
2802
Display a portion of a document in a first window
2804
Receive a search request including one or more search terms
2806
Request one or more of the linked documents from the respective data sources in parallel
Display in the second window placeholders for each requested linked document
2808
2810
Receive a respective linked document from a respective data source
2812
Determine whether the respective linked document includes chunks that match at least one of the one or more search terms
Request a linked document that has not been previously requested
2830
2814
Match?
No
Remove a placeholder corresponding to the respective linked document
2828
Yes
2816
Determine whether a number of groups displayed in the second window is less than a predefined number of groups
2818
Less than predefined number of groups?
Yes
Display at least a subset of the chunks as a respective group in a second window
2820
No
2822
Display in the second window a next-page link
2824
Receive a user selection of a the next page link
Display in the second window a respective page link corresponding to the groups displayed in the second window
2826

**Figure 28**

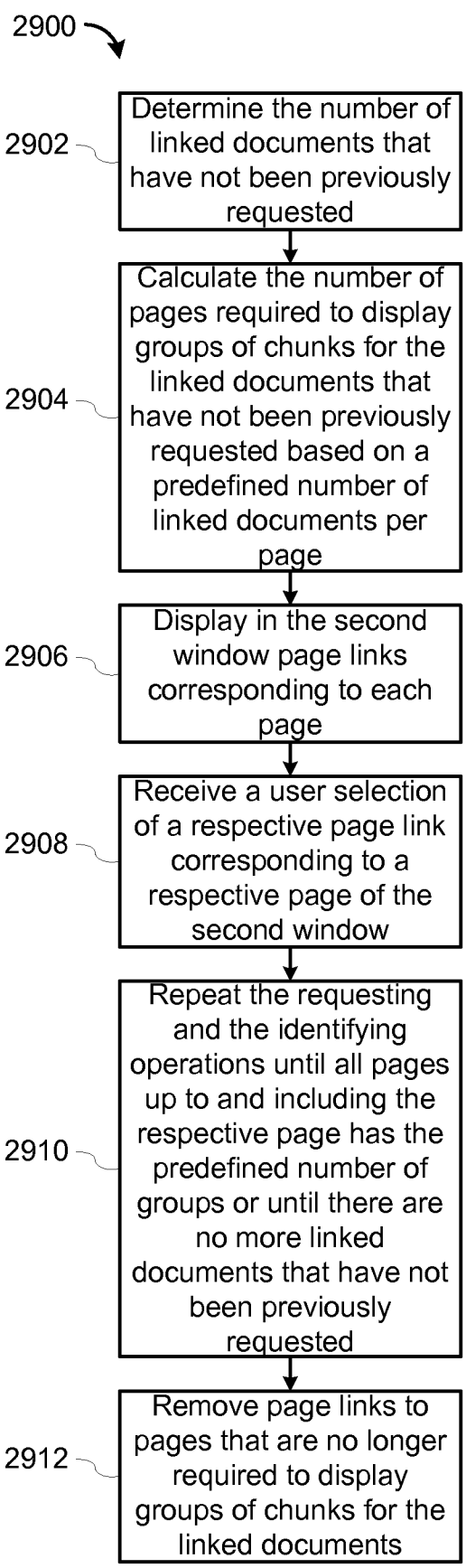



Figure 29

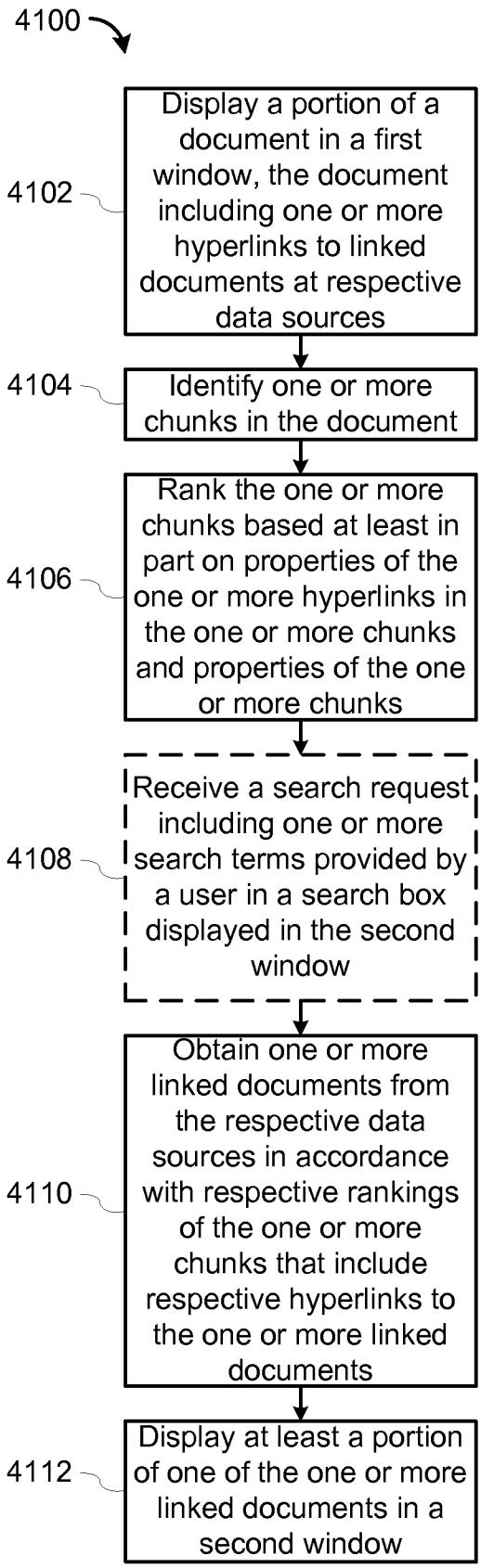



Figure 41

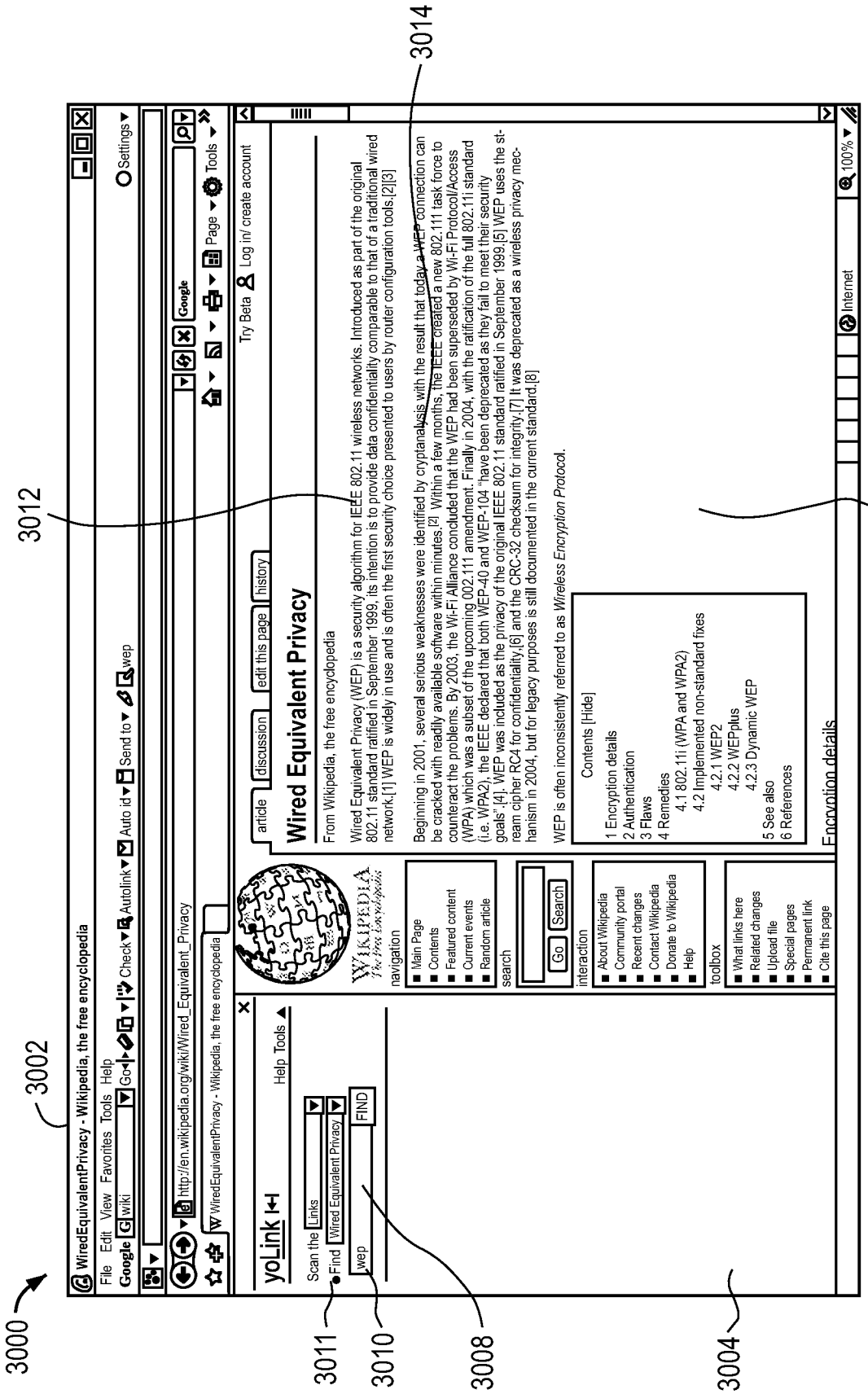

3000
3002
3004
3006
3008
3010
3011
3012
3014
WiredEquivalentPrivacy - Wikipedia, the free encyclopedia
Wired Equivalent Privacy
From Wikipedia, the free encyclopedia
Wired Equivalent Privacy (WEP) is a security algorithm for IEEE 802.11 wireless networks. Introduced as part of the original 802.11 standard ratified in September 1999, its intention is to provide data confidentiality comparable to that of a traditional wired network.[1] WEP is widely in use and is often the first security choice presented to users by router configuration tools.[2][3]
Beginning in 2001, several serious weaknesses were identified by cryptanalysis with the result that today a WEP connection can be cracked with readily available software within minutes.[2] Within a few months, the IEEE created a new 802.111 task force to counteract the problems. By 2003, the Wi-Fi Alliance concluded that the WEP had been superseded by Wi-Fi Protocol/Access (WPA) which was a subset of the upcoming 002.111 amendment. Finally in 2004, with the ratification of the full 802.11i standard (i.e. WPA2), the IEEE declared that both WEP-40 and WEP-104 "have been deprecated as they fail to meet their security goals".[4]. WEP was included as the privacy of the original IEEE 802.11 standard ratified in September 1999.[5] WEP uses the stream cipher RC4 for confidentiality,[6] and the CRC-32 checksum for integrity.[7] It was deprecated as a wireless privacy mechanism in 2004, but for legacy purposes is still documented in the current standard.[8]
WEP is often inconsistently referred to as Wireless Encryption Protocol.
Contents [Hide]
1 Encryption details
2 Authentication
3 Flaws
4 Remedies
4.1 802.11i (WPA and WPA2)
4.2 Implemented non-standard fixes
4.2.1 WEP2
4.2.2 WEPplus
4.2.3 Dynamic WEP
5 See also
6 References
Encryption details
navigation
search
interaction
toolbox

Figure 30

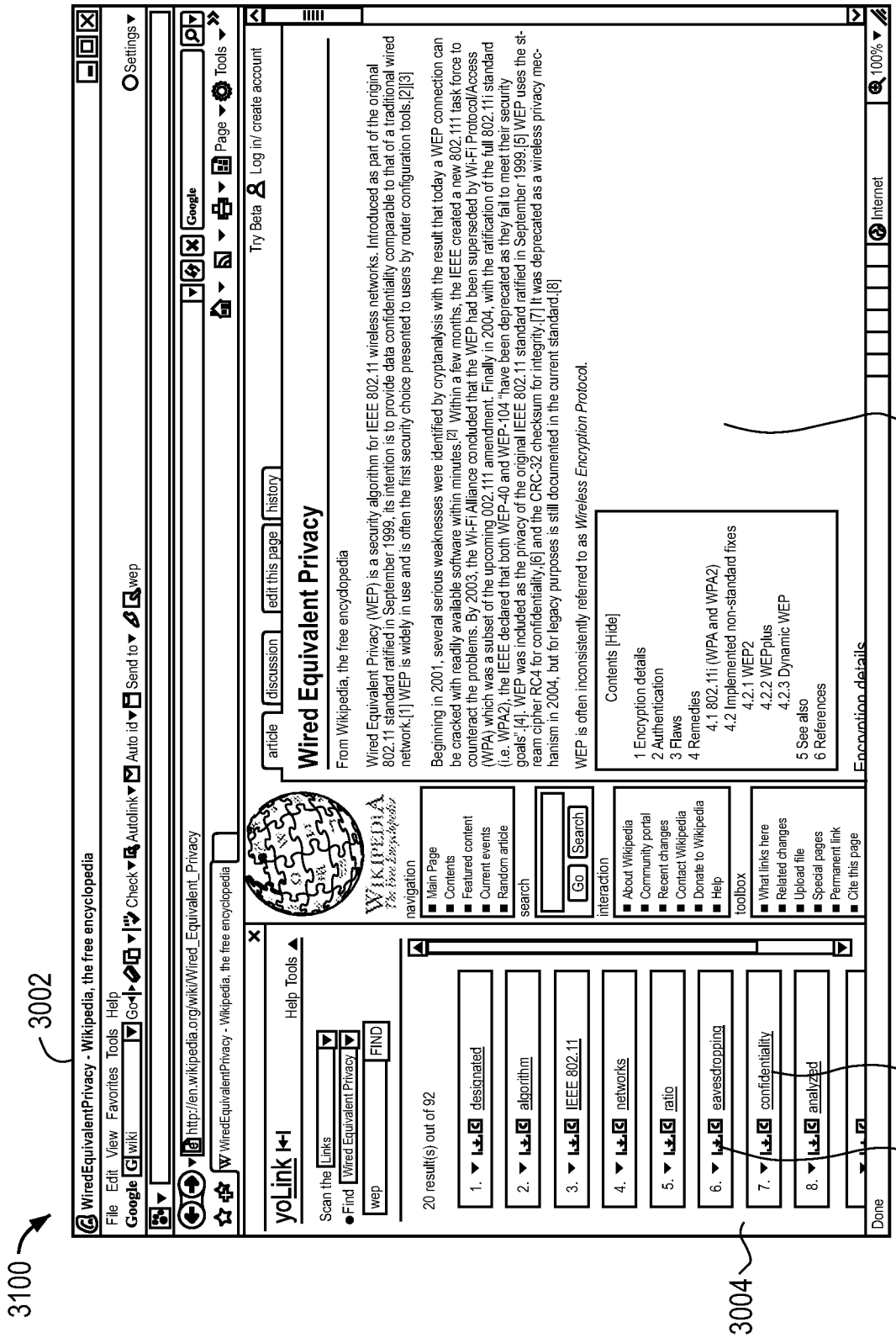

3100
3002
WiredEquivalentPrivacy - Wikipedia, the free encyclopedia
File Edit View Favorites Tools Help
http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy
article discussion edit this page history
Wired Equivalent Privacy
From Wikipedia, the free encyclopedia
Wired Equivalent Privacy (WEP), is a security algorithm for IEEE 802.11 wireless networks. Introduced as part of the original 802.11 standard ratified in September 1999, its intention is to provide data confidentiality comparable to that of a traditional wired network.[1] WEP is widely in use and is often the first security choice presented to users by router configuration tools.[2][3]
Beginning in 2001, several serious weaknesses were identified by cryptanalysis with the result that today a WEP connection can be cracked with readily available software within minutes.[2] Within a few months, the IEEE created a new 802.111 task force to counteract the problems. By 2003, the Wi-Fi Alliance concluded that the WEP had been superseded by Wi-Fi Protocol/Access (WPA) which was a subset of the upcoming 002.111 amendment. Finally in 2004, with the ratification of the full 802.11i standard (i.e. WPA2), the IEEE declared that both WEP-40 and WEP-104 "have been deprecated as they fail to meet their security goals".[4]. WEP was included as the privacy of the original IEEE 802.11 standard ratified in September 1999.[5] WEP uses the stream cipher RC4 for confidentiality,[6] and the CRC-32 checksum for integrity.[7] It was deprecated as a wireless privacy mechanism in 2004, but for legacy purposes is still documented in the current standard.[8]
WEP is often inconsistently referred to as *Wireless Encryption Protocol*.
Contents [Hide]
1 Encryption details
2 Authentication
3 Flaws
4 Remedies
4.1 802.11i (WPA and WPA2)
4.2 Implemented non-standard fixes
4.2.1 WEP2
4.2.2 WEPplus
4.2.3 Dynamic WEP
5 See also
6 References
Encryption details
yoLink
Scan the Links
Find Wired Equivalent Privacy
wep
FIND
20 result(s) out of 92
1. designated
2. algorithm
3. IEEE 802.11
4. networks
5. ratio
6. eavesdropping
7. confidentiality
8. analyzed
navigation
Main Page
Contents
Featured content
Current events
Random article
search
Go Search
interaction
About Wikipedia
Community portal
Recent changes
Contact Wikipedia
Donate to Wikipedia
Help
toolbox
What links here
Related changes
Upload file
Special pages
Permanent link
Cite this page
3004
3104
3106
3006
Done
Internet
100%

Figure 31

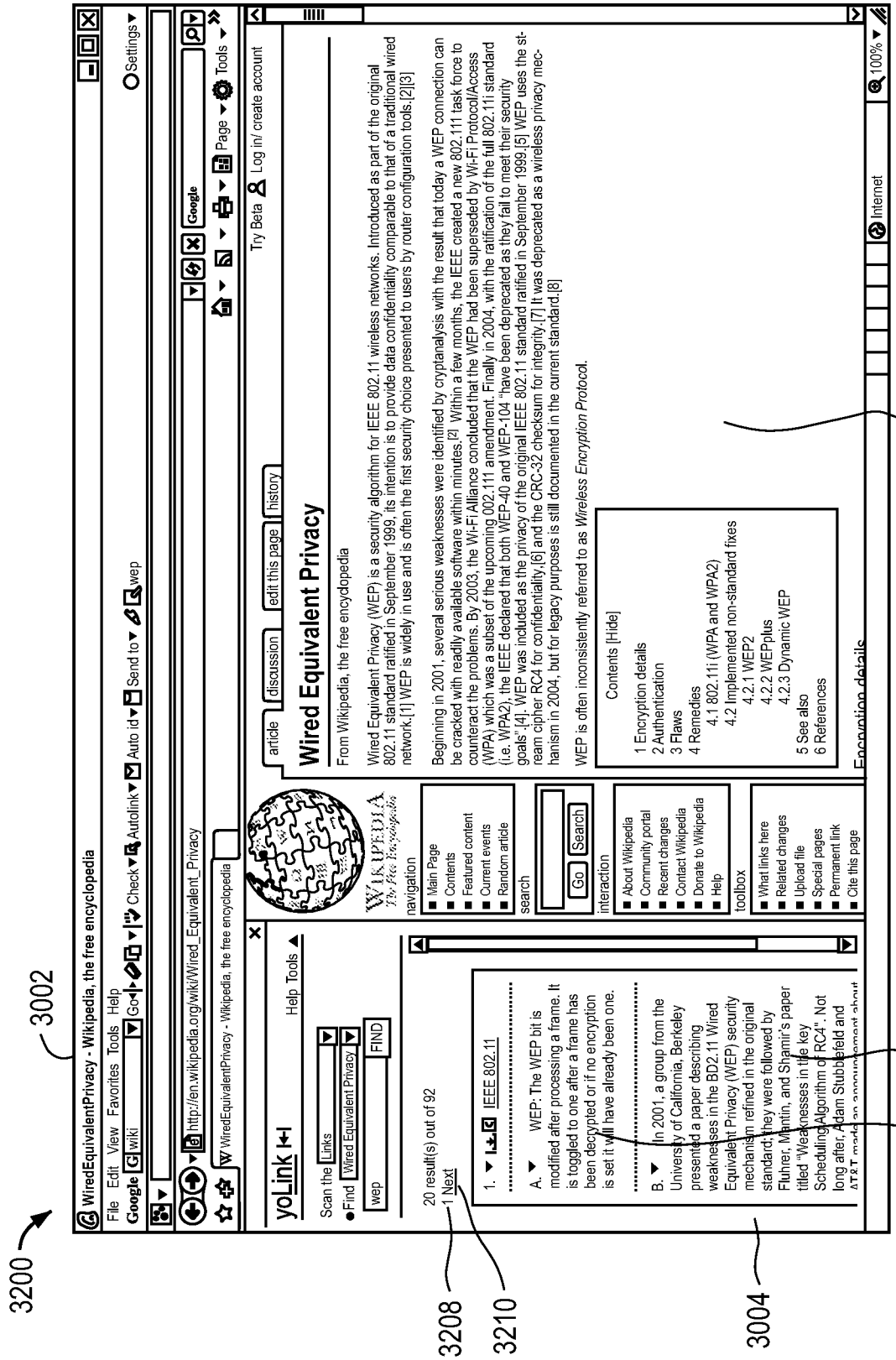

3200
3002
WiredEquivalentPrivacy - Wikipedia, the free encyclopedia
File Edit View Favorites Tools Help
http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy
yoLink
Scan the Links
Find Wired Equivalent Privacy
wep
FIND
Help Tools
20 result(s) out of 92
1 Next
1. IEEE 802.11
A. WEP: The WEP bit is modified after processing a frame. It is toggled to one after a frame has been decrypted or if no encryption is set it will have already been one.
B. In 2001, a group from the University of California, Berkeley presented a paper describing weaknesses in the BD2.11 Wired Equivalent Privacy (WEP) security mechanism refined in the original standard; they were followed by Fluhrer, Mantin, and Shamir's paper titled "Weaknesses in the key Scheduling Algorithm of RC4". Not long after, Adam Stubblefield and
article discussion edit this page history
Wired Equivalent Privacy
From Wikipedia, the free encyclopedia
Wired Equivalent Privacy (WEP) is a security algorithm for IEEE 802.11 wireless networks. Introduced as part of the original 802.11 standard ratified in September 1999, its intention is to provide data confidentiality comparable to that of a traditional wired network.[1] WEP is widely in use and is often the first security choice presented to users by router configuration tools.[2][3]
Beginning in 2001, several serious weaknesses were identified by cryptanalysis with the result that today a WEP connection can be cracked with readily available software within minutes.[2] Within a few months, the IEEE created a new 802.111 task force to counteract the problems. By 2003, the Wi-Fi Alliance concluded that the WEP had been superseded by Wi-Fi Protocol/Access (WPA) which was a subset of the upcoming 002.111 amendment. Finally in 2004, with the ratification of the full 802.11i standard (i.e. WPA2), the IEEE declared that both WEP-40 and WEP-104 "have been deprecated as they fail to meet their security goals".[4] WEP was included as the privacy of the original IEEE 802.11 standard ratified in September 1999.[5] WEP uses the stream cipher RC4 for confidentiality.[6] and the CRC-32 checksum for integrity.[7] It was deprecated as a wireless privacy mechanism in 2004, but for legacy purposes is still documented in the current standard.[8]
WEP is often inconsistently referred to as *Wireless Encryption Protocol*.
Contents [Hide]
1 Encryption details
2 Authentication
3 Flaws
4 Remedies
4.1 802.11i (WPA and WPA2)
4.2 Implemented non-standard fixes
4.2.1 WEP2
4.2.2 WEPplus
4.2.3 Dynamic WEP
5 See also
6 References
Encryption details
navigation
Main Page
Contents
Featured content
Current events
Random article
search
Go Search
interaction
About Wikipedia
Community portal
Recent changes
Contact Wikipedia
Donate to Wikipedia
Help
toolbox
What links here
Related changes
Upload file
Special pages
Permanent link
Cite this page
Try Beta Log in/ create account
Internet
100%
3208
3210
3004
3204
3206
3006

Figure 32

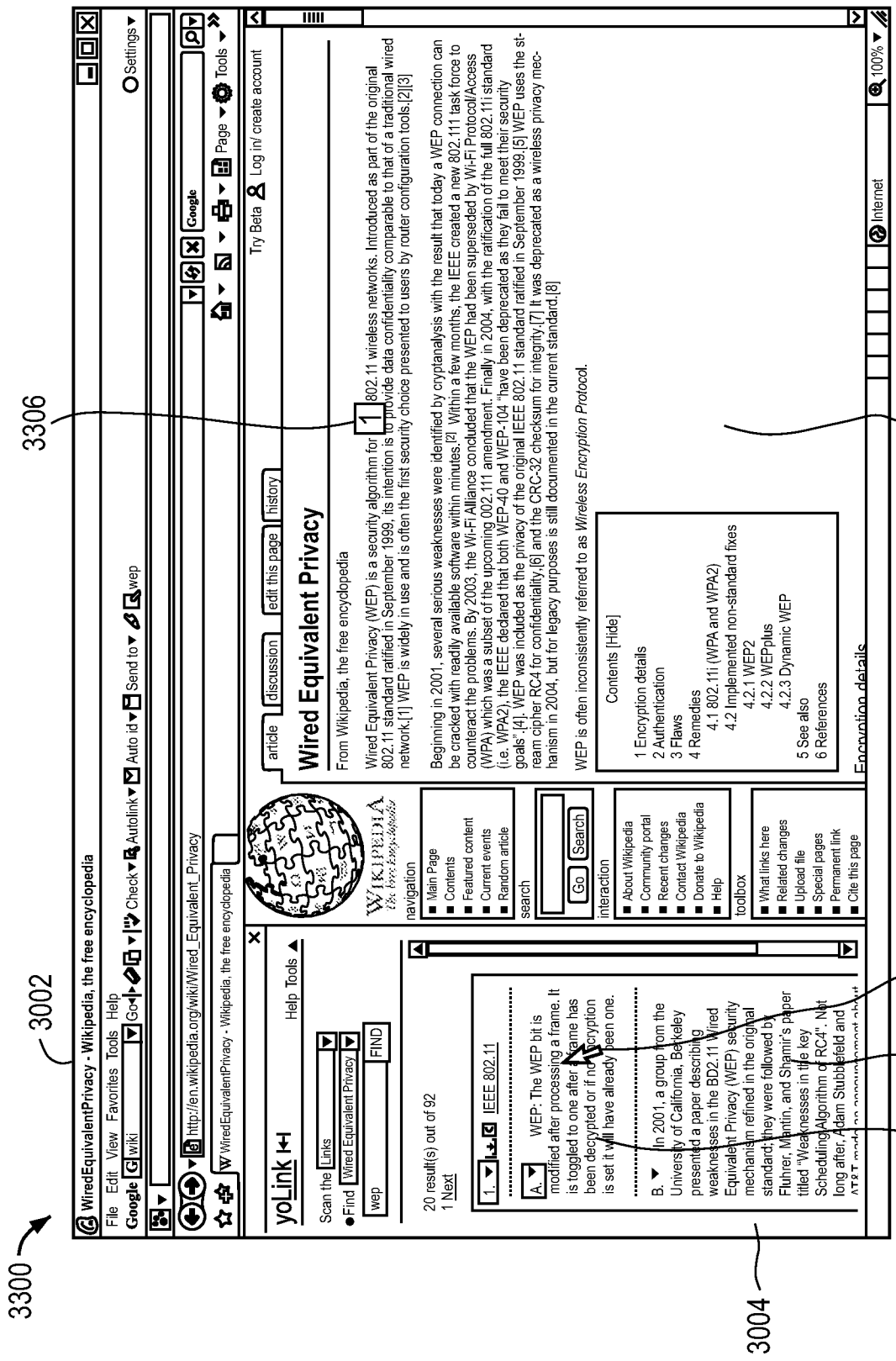

3300
3002
3306
3004
3204
3206
3304
3006
WiredEquivalentPrivacy - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy
Wired Equivalent Privacy
From Wikipedia, the free encyclopedia
Wired Equivalent Privacy (WEP) is a security algorithm for 802.11 wireless networks. Introduced as part of the original 802.11 standard ratified in September 1999, its intention is to provide data confidentiality comparable to that of a traditional wired network.[1] WEP is widely in use and is often the first security choice presented to users by router configuration tools.[2][3]
Beginning in 2001, several serious weaknesses were identified by cryptanalysis with the result that today a WEP connection can be cracked with readily available software within minutes.[2] Within a few months, the IEEE created a new 802.111 task force to counteract the problems. By 2003, the Wi-Fi Alliance concluded that the WEP had been superseded by Wi-Fi Protocol/Access (WPA) which was a subset of the upcoming 002.111 amendment. Finally in 2004, with the ratification of the full 802.11i standard (i.e. WPA2), the IEEE declared that both WEP-40 and WEP-104 "have been deprecated as they fail to meet their security goals".[4]. WEP was included as the privacy of the original IEEE 802.11 standard ratified in September 1999.[5] WEP uses the stream cipher RC4 for confidentiality,[6] and the CRC-32 checksum for integrity.[7] It was deprecated as a wireless privacy mechanism in 2004, but for legacy purposes is still documented in the current standard.[8]
WEP is often inconsistently referred to as Wireless Encryption Protocol.
Contents [Hide]
1 Encryption details
2 Authentication
3 Flaws
4 Remedies
4.1 802.11i (WPA and WPA2)
4.2 Implemented non-standard fixes
4.2.1 WEP2
4.2.2 WEPplus
4.2.3 Dynamic WEP
5 See also
6 References
Encryption details
navigation
Main Page
Contents
Featured content
Current events
Random article
search
Go
Search
interaction
About Wikipedia
Community portal
Recent changes
Contact Wikipedia
Donate to Wikipedia
Help
toolbox
What links here
Related changes
Upload file
Special pages
Permanent link
Cite this page
article
discussion
edit this page
history
Try Beta
Log in/ create account
yoLink
Help
Tools
Scan the Links
Find Wired Equivalent Privacy
wep
FIND
20 result(s) out of 92
1 Next
1. IEEE 802.11
A. WEP: The WEP bit is modified after processing a frame. It is toggled to one after a frame has been decrypted or if no encryption is set it will have already been one.
B. In 2001, a group from the University of California, Berkeley presented a paper describing weaknesses in the BD2.11 Wired Equivalent Privacy (WEP) security mechanism refined in the original standard; they were followed by Fluhrer, Mantin, and Shamir's paper titled "Weaknesses in the key Scheduling Algorithm of RC4". Not long after, Adam Stubblefield and
Internet
100%

Figure 33

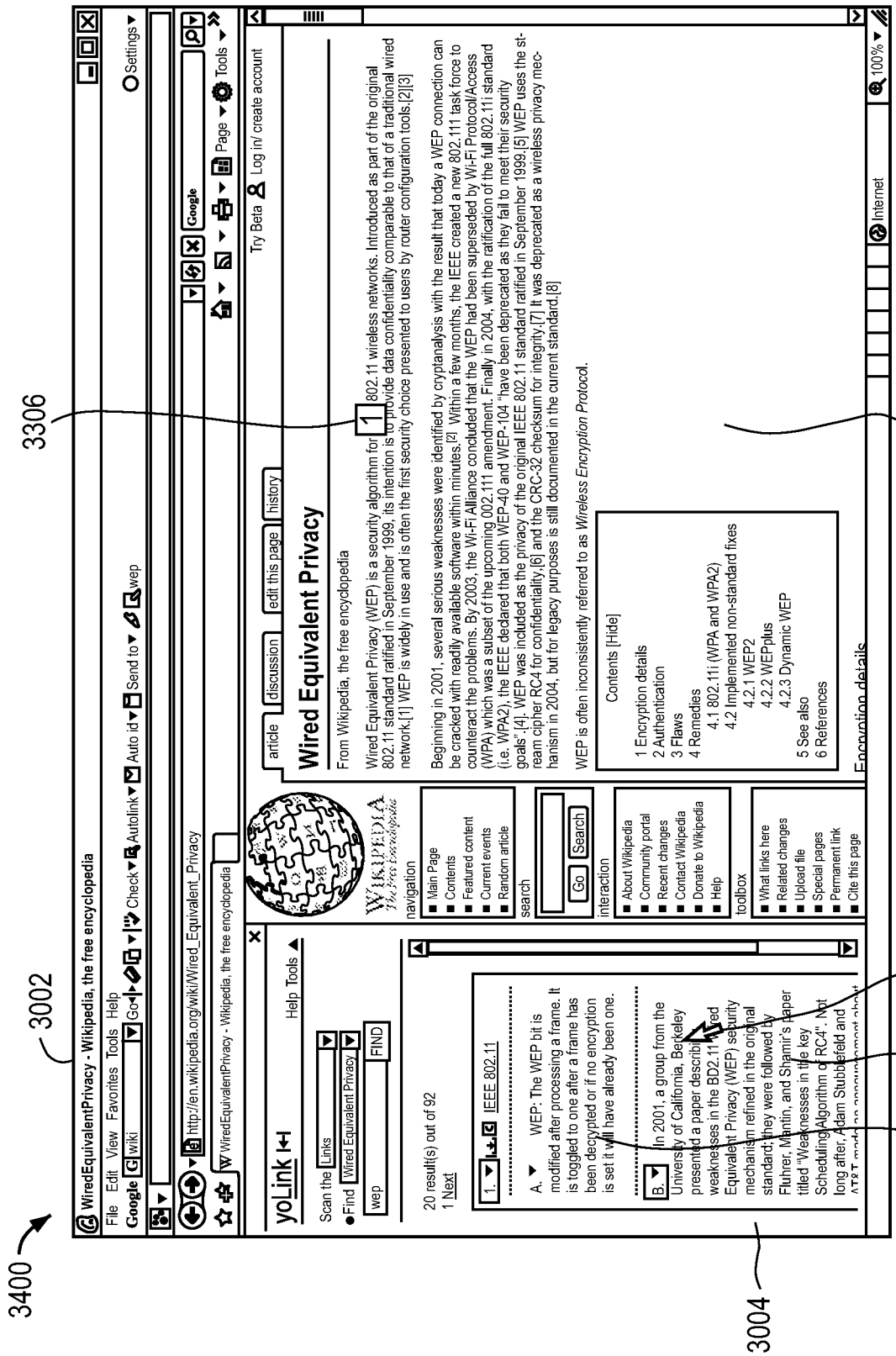

3400
3002
3306
3004
3204
3206
3304
3006
WiredEquivalentPrivacy - Wikipedia, the free encyclopedia
Wired Equivalent Privacy
From Wikipedia, the free encyclopedia
Wired Equivalent Privacy (WEP) is a security algorithm for 802.11 wireless networks. Introduced as part of the original 802.11 standard ratified in September 1999, its intention is to provide data confidentiality comparable to that of a traditional wired network.[1] WEP is widely in use and is often the first security choice presented to users by router configuration tools.[2][3]
Beginning in 2001, several serious weaknesses were identified by cryptanalysis with the result that today a WEP connection can be cracked with readily available software within minutes.[2] Within a few months, the IEEE created a new 802.11i task force to counteract the problems. By 2003, the Wi-Fi Alliance concluded that the WEP had been superseded by Wi-Fi Protocol/Access (WPA) which was a subset of the upcoming 002.111 amendment. Finally in 2004, with the ratification of the full 802.11i standard (i.e. WPA2), the IEEE declared that both WEP-40 and WEP-104 "have been deprecated as they fail to meet their security goals".[4]. WEP was included as the privacy of the original IEEE 802.11 standard ratified in September 1999.[5] WEP uses the stream cipher RC4 for confidentiality,[6] and the CRC-32 checksum for integrity.[7] It was deprecated as a wireless privacy mechanism in 2004, but for legacy purposes is still documented in the current standard.[8]
WEP is often inconsistently referred to as Wireless Encryption Protocol.
Contents [Hide]
1 Encryption details
2 Authentication
3 Flaws
4 Remedies
4.1 802.11i (WPA and WPA2)
4.2 Implemented non-standard fixes
4.2.1 WEP2
4.2.2 WEPplus
4.2.3 Dynamic WEP
5 See also
6 References
Encryption details
20 result(s) out of 92
1 Next
1. IEEE 802.11
A. WEP: The WEP bit is modified after processing a frame. It is toggled to one after a frame has been decrypted or if no encryption is set it will have already been one.
B. In 2001, a group from the University of California, Berkeley presented a paper describing weaknesses in the 802.11 Wired Equivalent Privacy (WEP) security mechanism refined in the original standard; they were followed by Fluhrer, Mantin, and Shamir's paper titled "Weaknesses in the key Scheduling Algorithm of RC4". Not long after, Adam Stubblefield and
navigation
Main Page
Contents
Featured content
Current events
Random article
search
Go
Search
interaction
About Wikipedia
Community portal
Recent changes
Contact Wikipedia
Donate to Wikipedia
Help
toolbox
What links here
Related changes
Upload file
Special pages
Permanent link
Cite this page
article
discussion
edit this page
history
Try Beta
Log in/ create account
yoLink
Scan the Links
Find Wired Equivalent Privacy
wep
FIND
Help Tools
Internet
100%

Figure 34

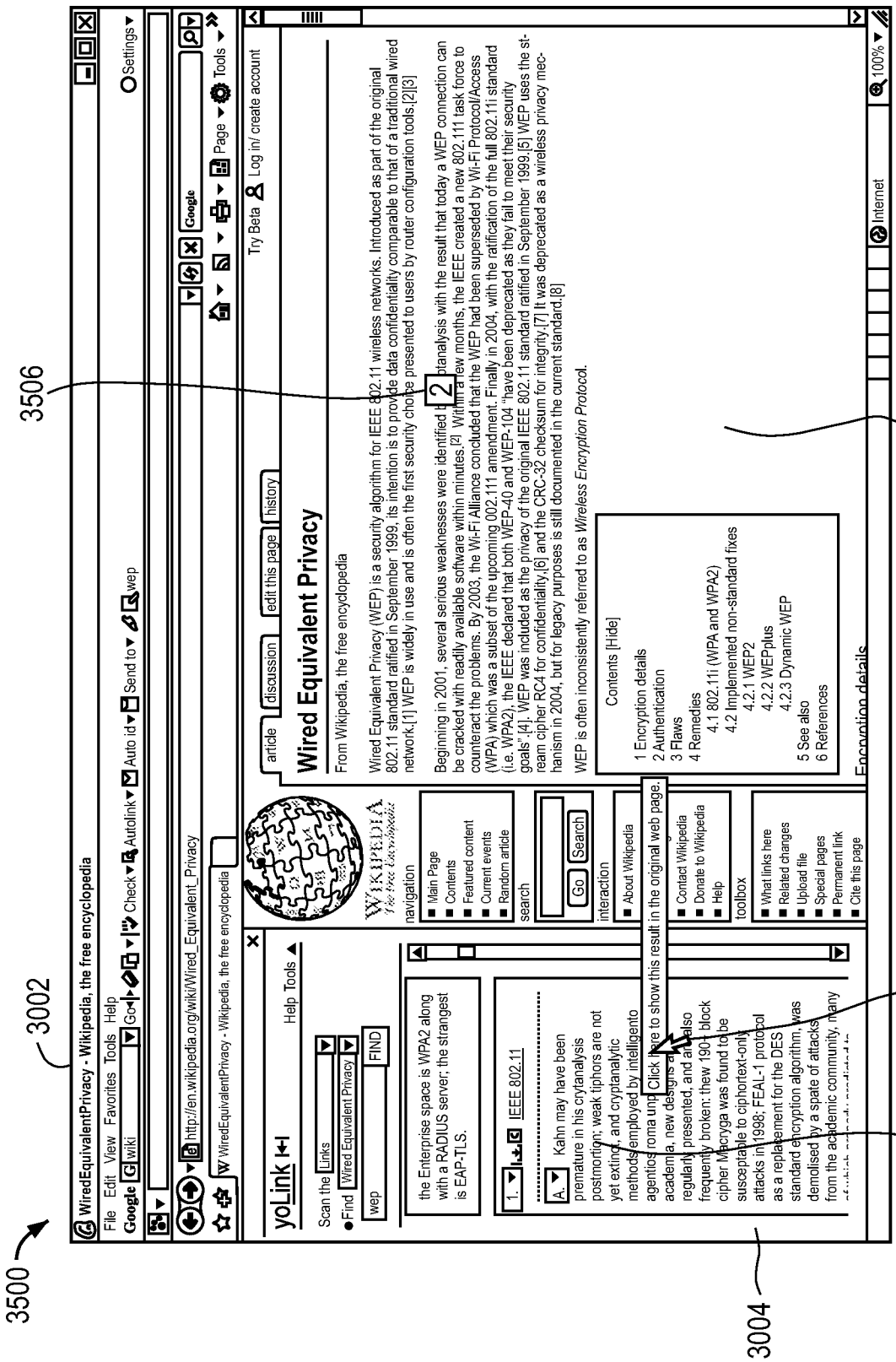

3500
3002
3506
3504
3304
3004
3006
WiredEquivalentPrivacy - Wikipedia, the free encyclopedia
Wired Equivalent Privacy
From Wikipedia, the free encyclopedia
Contents [Hide]
1 Encryption details
2 Authentication
3 Flaws
4 Remedies
4.1 802.11i (WPA and WPA2)
4.2 Implemented non-standard fixes
4.2.1 WEP2
4.2.2 WEPplus
4.2.3 Dynamic WEP
5 See also
6 References
Encryption details

Figure 35

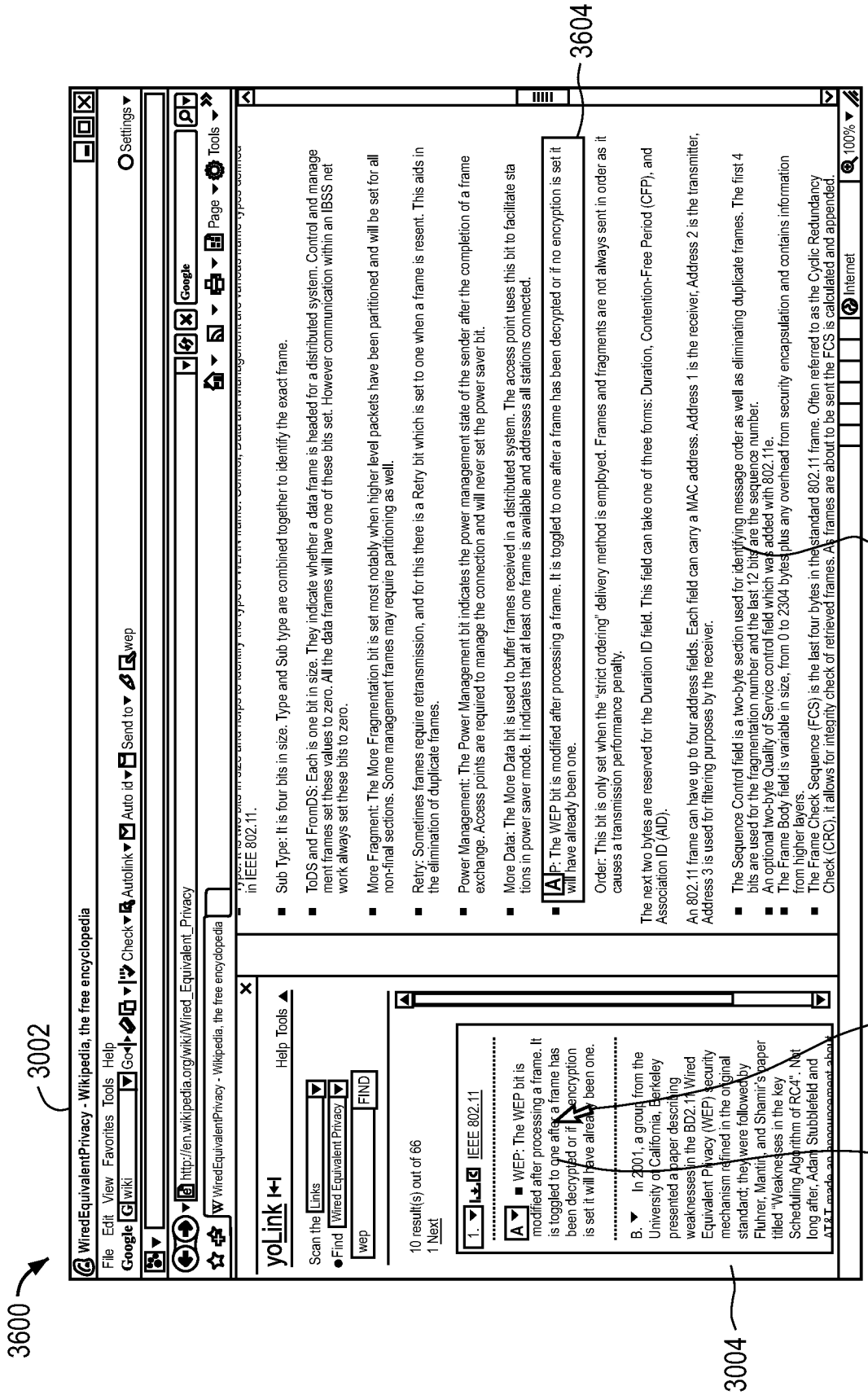

3600
3002
3004
3204
3304
3604
3006
WiredEquivalentPrivacy - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy
yoLink
Scan the Links
Find Wired Equivalent Privacy
wep
FIND
10 result(s) out of 66
1 Next
IEEE 802.11
WEP: The WEP bit is modified after processing a frame. It is toggled to one after a frame has been decrypted or if no encryption is set it will have already been one.
Sub Type: It is four bits in size. Type and Sub type are combined together to identify the exact frame.
ToDS and FromDS: Each is one bit in size. They indicate whether a data frame is headed for a distributed system. Control and management frames set these values to zero. All the data frames will have one of these bits set. However communication within an IBSS network always set these bits to zero.
More Fragment: The More Fragmentation bit is set most notably when higher level packets have been partitioned and will be set for all non-final sections. Some management frames may require partitioning as well.
Retry: Sometimes frames require retransmission, and for this there is a Retry bit which is set to one when a frame is resent. This aids in the elimination of duplicate frames.
Power Management: The Power Management bit indicates the power management state of the sender after the completion of a frame exchange. Access points are required to manage the connection and will never set the power saver bit.
More Data: The More Data bit is used to buffer frames received in a distributed system. The access point uses this bit to facilitate stations in power saver mode. It indicates that at least one frame is available and addresses all stations connected.
P: The WEP bit is modified after processing a frame. It is toggled to one after a frame has been decrypted or if no encryption is set it will have already been one.
Order: This bit is only set when the "strict ordering" delivery method is employed. Frames and fragments are not always sent in order as it causes a transmission performance penalty.
The next two bytes are reserved for the Duration ID field. This field can take one of three forms: Duration, Contention-Free Period (CFP), and Association ID (AID).
An 802.11 frame can have up to four address fields. Each field can carry a MAC address. Address 1 is the receiver, Address 2 is the transmitter, Address 3 is used for filtering purposes by the receiver.
The Sequence Control field is a two-byte section used for identifying message order as well as eliminating duplicate frames. The first 4 bits are used for the fragmentation number and the last 12 bits are the sequence number.
An optional two-byte Quality of Service control field which was added with 802.11e.
The Frame Body field is variable in size, from 0 to 2304 bytes plus any overhead from security encapsulation and contains information from higher layers.
The Frame Check Sequence (FCS) is the last four bytes in the standard 802.11 frame. Often referred to as the Cyclic Redundancy Check (CRC), it allows for integrity check of retrieved frames. As frames are about to be sent the FCS is calculated and appended.
B. In 2001, a group from the University of California, Berkeley presented a paper describing weaknesses in the BD2.11 Wired Equivalent Privacy (WEP) security mechanism defined in the original standard; they were followed by Fluhrer, Mantin, and Shamir's paper titled "Weaknesses in the Key Scheduling Algorithm of RC4". Not long after, Adam Stubblefield and
Internet
100%

Figure 36

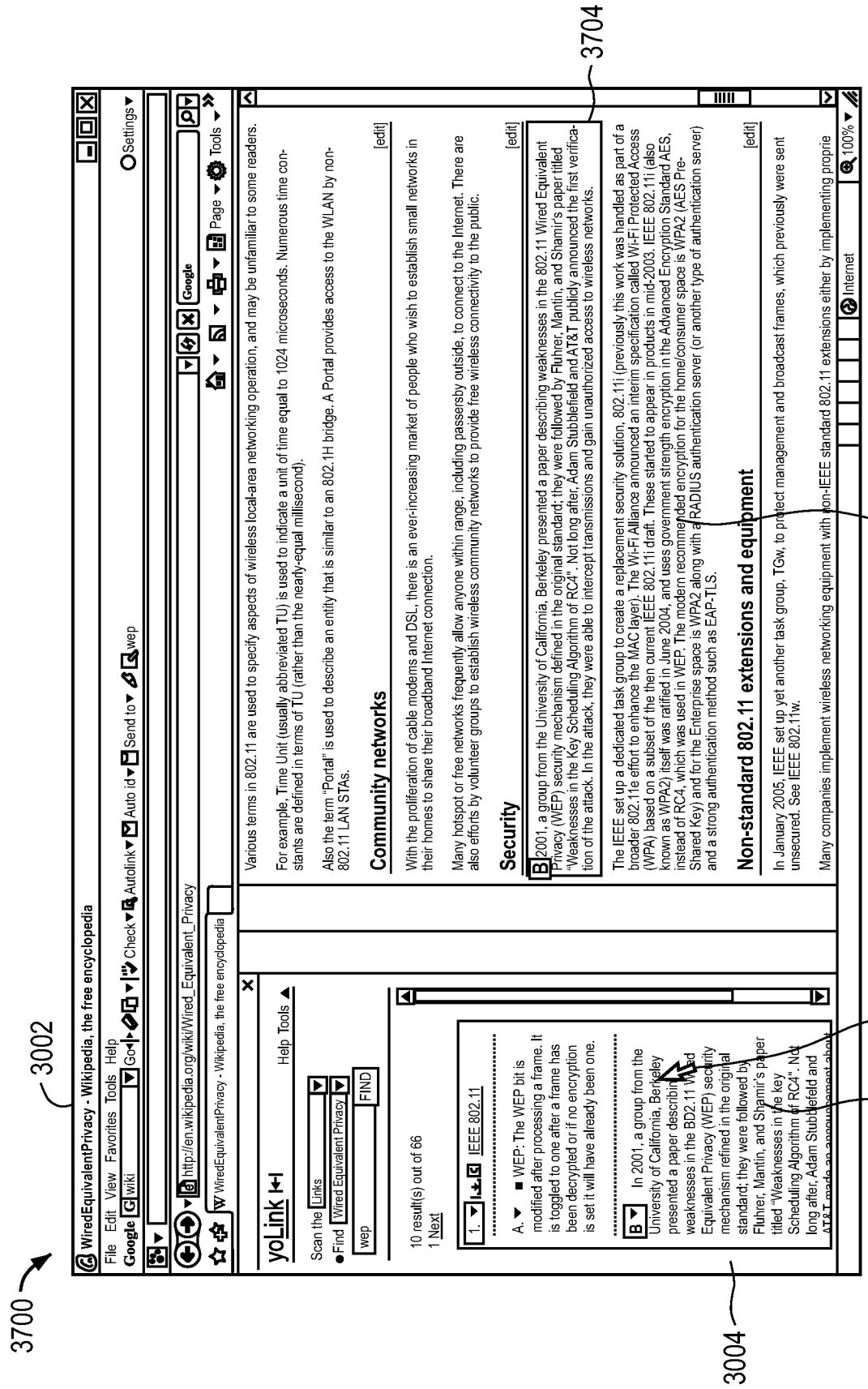

3700
3002
3004
3206
3304
3704
3006
WiredEquivalentPrivacy - Wikipedia, the free encyclopedia
File Edit View Favorites Tools Help
http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy
yoLink
Help Tools
Scan the Links
Find Wired Equivalent Privacy
wep
FIND
10 result(s) out of 66
1 Next
1. IEEE 802.11
A. WEP: The WEP bit is modified after processing a frame. It is toggled to one after a frame has been decrypted or if no encryption is set it will have already been one.
B In 2001, a group from the University of California, Berkeley presented a paper describing weaknesses in the 802.11 Wired Equivalent Privacy (WEP) security mechanism refined in the original standard; they were followed by Fluhrer, Mantin, and Shamir's paper titled "Weaknesses in the Key Scheduling Algorithm of RC4". Not long after, Adam Stubblefield and
Various terms in 802.11 are used to specify aspects of wireless local-area networking operation, and may be unfamiliar to some readers.
For example, Time Unit (usually abbreviated TU) is used to indicate a unit of time equal to 1024 microseconds. Numerous time constants are defined in terms of TU (rather than the nearly-equal millisecond).
Also the term "Portal" is used to describe an entity that is similar to an 802.1H bridge. A Portal provides access to the WLAN by non-802.11 LAN STAs.
Community networks [edit]
With the proliferation of cable modems and DSL, there is an ever-increasing market of people who wish to establish small networks in their homes to share their broadband Internet connection.
Many hotspot or free networks frequently allow anyone within range, including passersby outside, to connect to the Internet. There are also efforts by volunteer groups to establish wireless community networks to provide free wireless connectivity to the public.
Security [edit]
B 2001, a group from the University of California, Berkeley presented a paper describing weaknesses in the 802.11 Wired Equivalent Privacy (WEP) security mechanism defined in the original standard; they were followed by Fluhrer, Mantin, and Shamir's paper titled "Weaknesses in the Key Scheduling Algorithm of RC4". Not long after, Adam Stubblefield and AT&T publicly announced the first verification of the attack. In the attack, they were able to intercept transmissions and gain unauthorized access to wireless networks.
The IEEE set up a dedicated task group to create a replacement security solution, 802.11i (previously this work was handled as part of a broader 802.11e effort to enhance the MAC layer). The Wi-Fi Alliance announced an interim specification called Wi-Fi Protected Access (WPA) based on a subset of the then current IEEE 802.11i draft. These started to appear in products in mid-2003. IEEE 802.11i (also known as WPA2) itself was ratified in June 2004, and uses government strength encryption in the Advanced Encryption Standard AES, instead of RC4, which was used in WEP. The modern recommended encryption for the home/consumer space is WPA2 (AES Pre-Shared Key) and for the Enterprise space is WPA2 along with a RADIUS authentication server (or another type of authentication server) and a strong authentication method such as EAP-TLS.
Non-standard 802.11 extensions and equipment [edit]
In January 2005, IEEE set up yet another task group, TGw, to protect management and broadcast frames, which previously were sent unsecured. See IEEE 802.11w.
Many companies implement wireless networking equipment with non-IEEE standard 802.11 extensions either by implementing proprie
Internet
100%

Figure 37

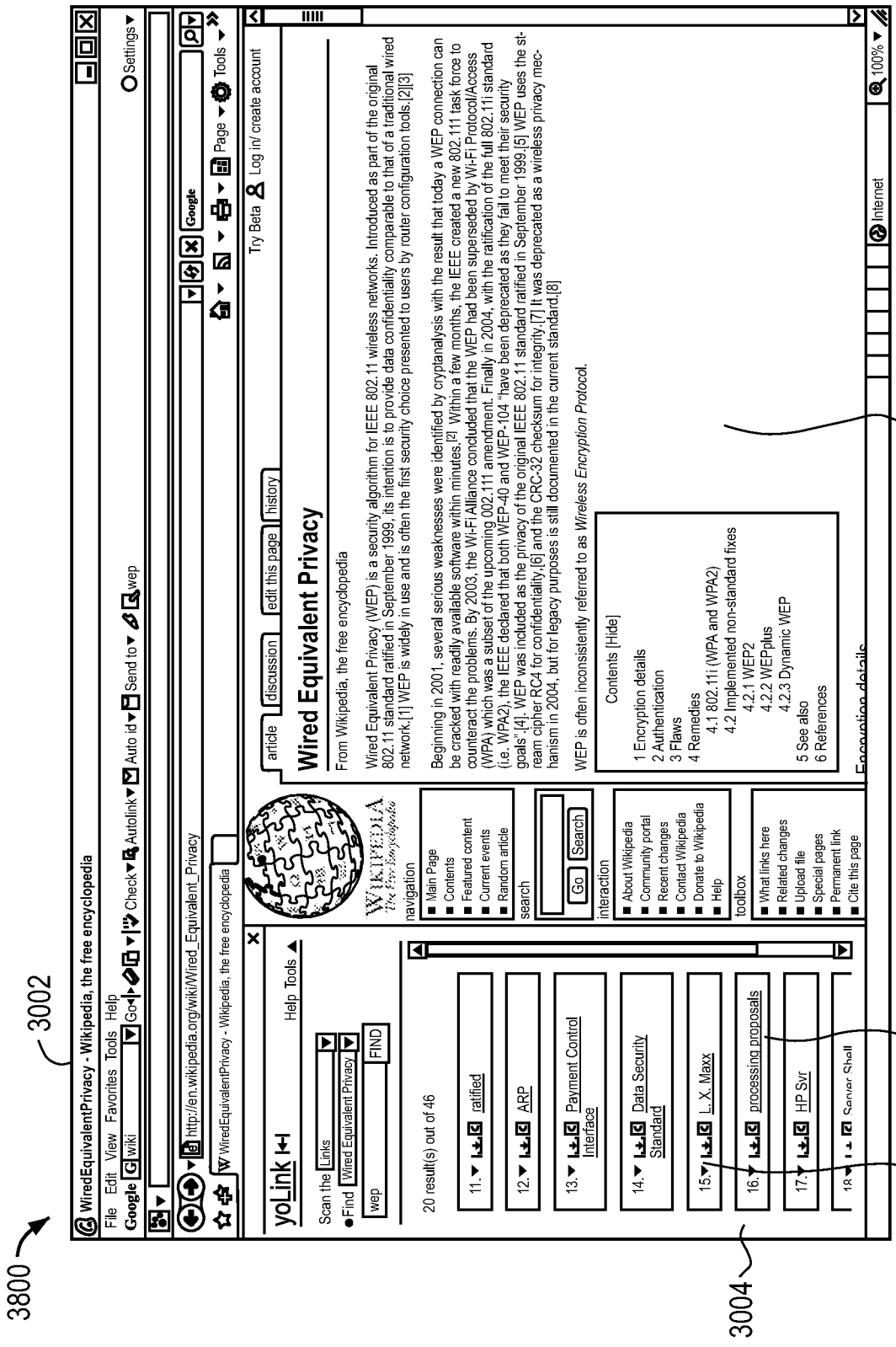

3800
3002
3004
3804
3806
3006
WiredEquivalentPrivacy - Wikipedia, the free encyclopedia
File Edit View Favorites Tools Help
http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy
yoLink
Help Tools
Scan the Links
Find Wired Equivalent Privacy
wep
FIND
20 result(s) out of 46
11. ratified
12. ARP
13. Payment Control Interface
14. Data Security Standard
15. L. X. Maxx
16. processing proposals
17. HP Svr
18. Server Shell
article discussion edit this page history
Try Beta Log in/ create account
Wired Equivalent Privacy
From Wikipedia, the free encyclopedia
Wired Equivalent Privacy (WEP) is a security algorithm for IEEE 802.11 wireless networks. Introduced as part of the original 802.11 standard ratified in September 1999, its intention is to provide data confidentiality comparable to that of a traditional wired network.[1] WEP is widely in use and is often the first security choice presented to users by router configuration tools.[2][3]
Beginning in 2001, several serious weaknesses were identified by cryptanalysis with the result that today a WEP connection can be cracked with readily available software within minutes.[2] Within a few months, the IEEE created a new 802.111 task force to counteract the problems. By 2003, the Wi-Fi Alliance concluded that the WEP had been superseded by Wi-Fi Protocol/Access (WPA) which was a subset of the upcoming 002.111 amendment. Finally in 2004, with the ratification of the full 802.11i standard (i.e. WPA2), the IEEE declared that both WEP-40 and WEP-104 "have been deprecated as they fail to meet their security goals".[4]. WEP was included as the privacy of the original IEEE 802.11 standard ratified in September 1999.[5] WEP uses the stream cipher RC4 for confidentiality,[6] and the CRC-32 checksum for integrity.[7] It was deprecated as a wireless privacy mechanism in 2004, but for legacy purposes is still documented in the current standard.[8]
WEP is often inconsistently referred to as *Wireless Encryption Protocol*.
Contents [Hide]
1 Encryption details
2 Authentication
3 Flaws
4 Remedies
4.1 802.11i (WPA and WPA2)
4.2 Implemented non-standard fixes
4.2.1 WEP2
4.2.2 WEPplus
4.2.3 Dynamic WEP
5 See also
6 References
Encryption details
navigation
Main Page
Contents
Featured content
Current events
Random article
search
Go Search
interaction
About Wikipedia
Community portal
Recent changes
Contact Wikipedia
Donate to Wikipedia
Help
toolbox
What links here
Related changes
Upload file
Special pages
Permanent link
Cite this page
Internet
100%

Figure 38

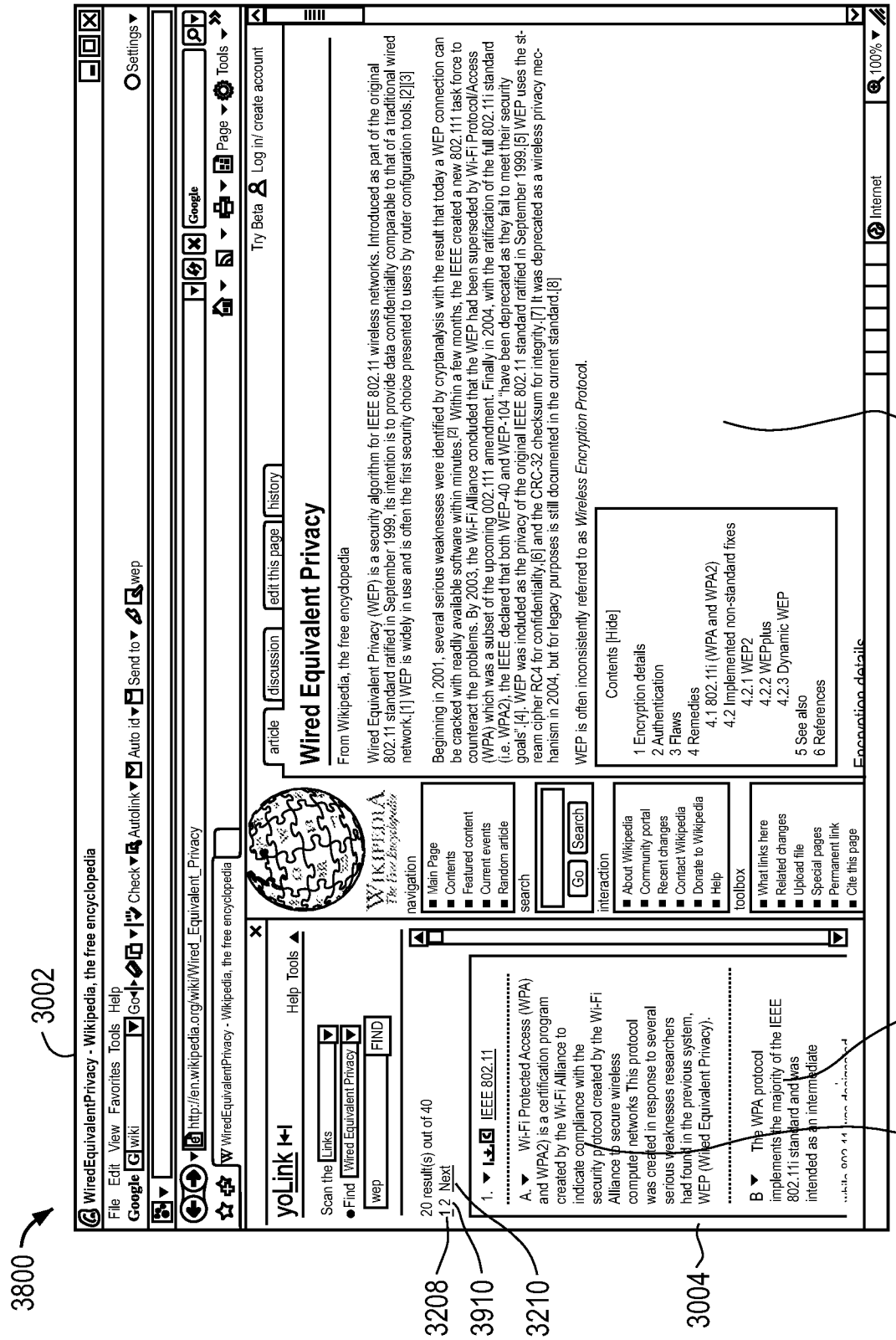

3800
3002
WiredEquivalentPrivacy - Wikipedia, the free encyclopedia
File Edit View Favorites Tools Help
Google wiki Go Check Autolink Auto id Send to wep Settings
http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy
WiredEquivalentPrivacy - Wikipedia, the free encyclopedia
Page Tools
yoLink Help Tools
Scan the Links
Find Wired Equivalent Privacy
wep FIND
20 result(s) out of 40
1 2 Next
1. IEEE 802.11
A. Wi-Fi Protected Access (WPA) and WPA2) is a certification program created by the Wi-Fi Alliance to indicate compliance with the security protocol created by the Wi-Fi Alliance to secure wireless computer networks This protocol was created in response to several serious weaknesses researchers had found in the previous system, WEP (Wired Equivalent Privacy).
B The WPA protocol implements the majority of the IEEE 802.11i standard and was intended as an intermediate
3208
3910
3210
3004
3904
3906
article discussion edit this page history
Try Beta Log in/ create account
Wired Equivalent Privacy
From Wikipedia, the free encyclopedia
Wired Equivalent Privacy (WEP) is a security algorithm for IEEE 802.11 wireless networks. Introduced as part of the original 802.11 standard ratified in September 1999, its intention is to provide data confidentiality comparable to that of a traditional wired network.[1] WEP is widely in use and is often the first security choice presented to users by router configuration tools.[2][3]
Beginning in 2001, several serious weaknesses were identified by cryptanalysis with the result that today a WEP connection can be cracked with readily available software within minutes.[2] Within a few months, the IEEE created a new 802.111 task force to counteract the problems. By 2003, the Wi-Fi Alliance concluded that the WEP had been superseded by Wi-Fi Protocol/Access (WPA) which was a subset of the upcoming 002.111 amendment. Finally in 2004, with the ratification of the full 802.11i standard (i.e. WPA2), the IEEE declared that both WEP-40 and WEP-104 "have been deprecated as they fail to meet their security goals".[4]. WEP was included as the privacy of the original IEEE 802.11 standard ratified in September 1999.[5] WEP uses the stream cipher RC4 for confidentiality,[6] and the CRC-32 checksum for integrity.[7] It was deprecated as a wireless privacy mechanism in 2004, but for legacy purposes is still documented in the current standard.[8]
WEP is often inconsistently referred to as *Wireless Encryption Protocol*.
Contents [Hide]
1 Encryption details
2 Authentication
3 Flaws
4 Remedies
4.1 802.11i (WPA and WPA2)
4.2 Implemented non-standard fixes
4.2.1 WEP2
4.2.2 WEPplus
4.2.3 Dynamic WEP
5 See also
6 References
Encryption details
navigation
Main Page
Contents
Featured content
Current events
Random article
search
Go Search
interaction
About Wikipedia
Community portal
Recent changes
Contact Wikipedia
Donate to Wikipedia
Help
toolbox
What links here
Related changes
Upload file
Special pages
Permanent link
Cite this page
Internet 100%
3006

Figure 39

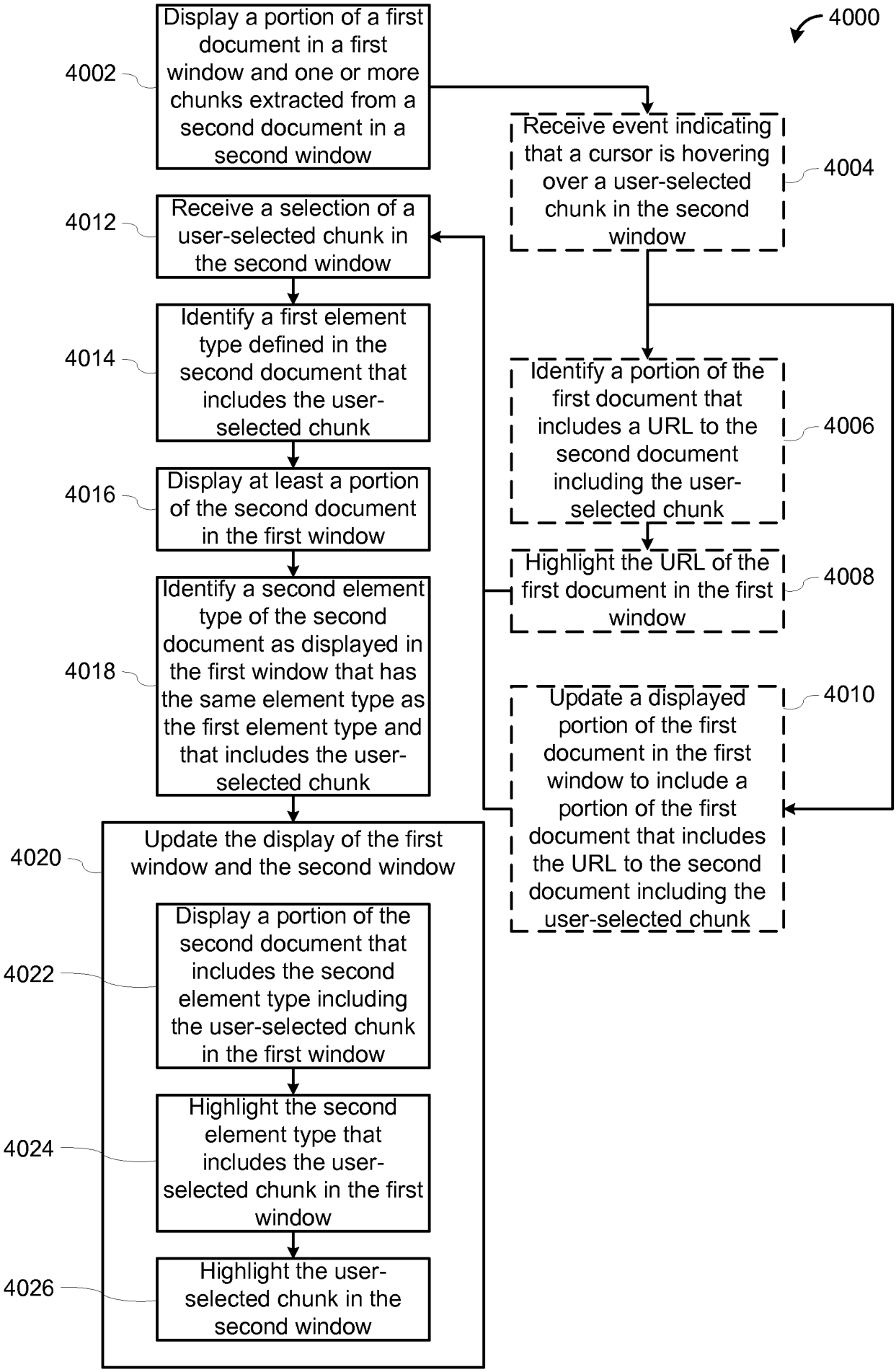

4000
4002
Display a portion of a first document in a first window and one or more chunks extracted from a second document in a second window
4004
Receive event indicating that a cursor is hovering over a user-selected chunk in the second window
4012
Receive a selection of a user-selected chunk in the second window
4014
Identify a first element type defined in the second document that includes the user-selected chunk
4016
Display at least a portion of the second document in the first window
4018
Identify a second element type of the second document as displayed in the first window that has the same element type as the first element type and that includes the user-selected chunk
4006
Identify a portion of the first document that includes a URL to the second document including the user-selected chunk
4008
Highlight the URL of the first document in the first window
4010
Update a displayed portion of the first document in the first window to include a portion of the first document that includes the URL to the second document including the user-selected chunk
4020
Update the display of the first window and the second window
4022
Display a portion of the second document that includes the second element type including the user-selected chunk in the first window
4024
Highlight the second element type that includes the user-selected chunk in the first window
4026
Highlight the user-selected chunk in the second window

Figure 40