



(22) Date de dépôt/Filing Date: 2011/05/27

(41) Mise à la disp. pub./Open to Public Insp.: 2012/11/27

(51) Cl.Int./Int.Cl. *G06F 17/30* (2006.01),
G06F 17/27 (2006.01)

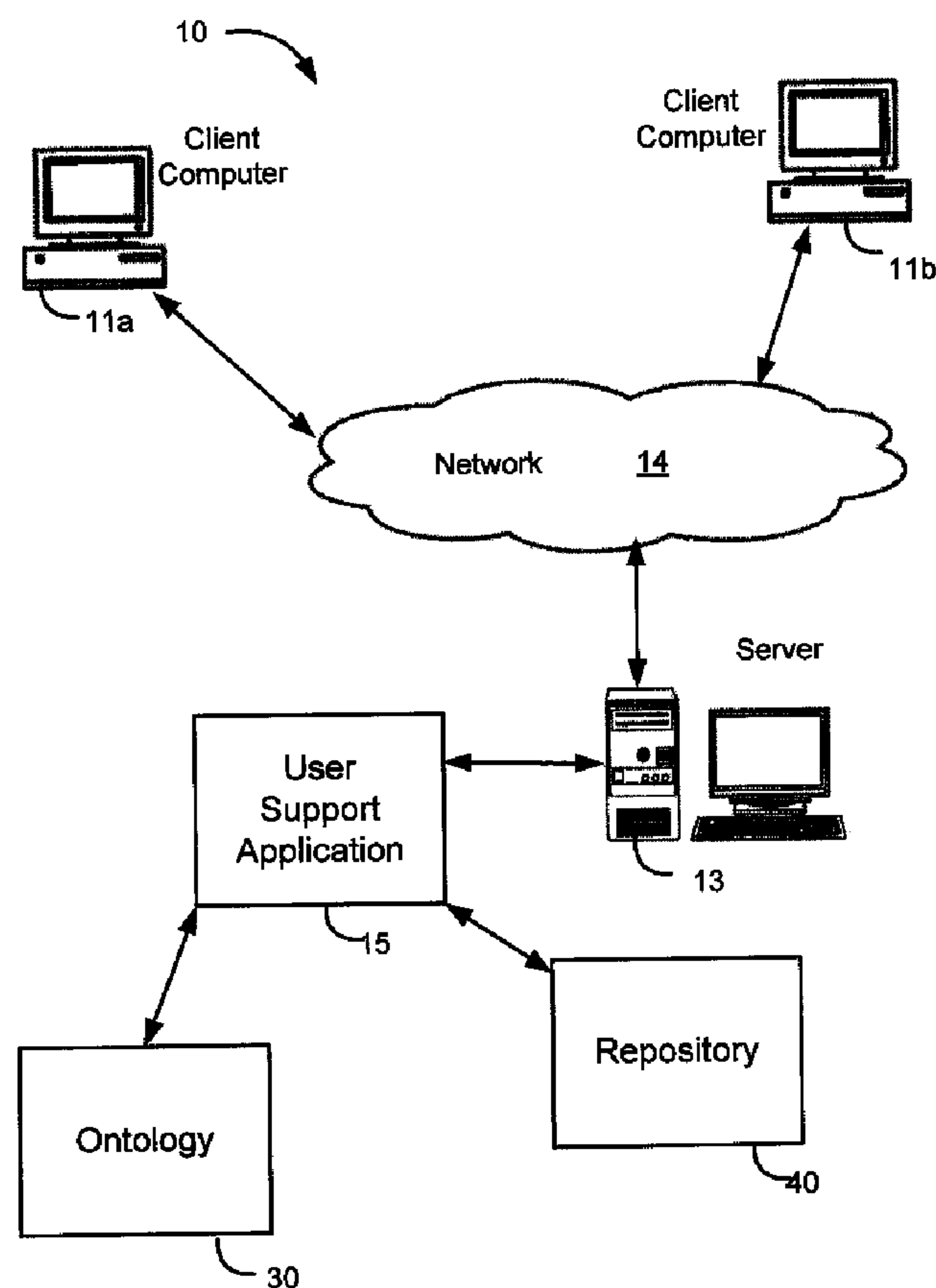
(71) Demandeur/Applicant:
IBM CANADA LIMITED - IBM CANADA LIMITEE, CA

(72) Inventeurs/Inventors:
BOYLE, P. CURRIE, CA;
JAVAHERY, HOMA, CA;
ROZSA, ENIKO I., CA;
YAMAMOTO, PAUL A., CA

(74) Agent: WANG, PETER

(54) Titre : ASSISTANCE D'UTILISATEUR DE LIBRE-SERVICE AUTOMATISE FONDE SUR L'ONTOLOGIE

(54) Title: AUTOMATED SELF-SERVICE USER SUPPORT BASED ON ONTOLOGY



(57) Abrégé/Abstract:

Embodiments of the disclosure relate to self-service user support applications based on analysis of an ontology on domain-specific information. An exemplary embodiment concerns a system for providing information in response to a user query that may comprise



(57) **Abrégé(suite)/Abstract(continued):**

a natural language parser for identifying relevant terms from the user query, and an ontology analyzer for matching the relevant terms to concepts in an ontology of information related to the user query. The system may include a query processor for refining the user query based on the matched query terms and ontology concepts, and a search engine for identifying from available repositories information relevant to the user query.

ABSTRACT OF THE DISCLOSURE

[0079] Embodiments of the disclosure relate to self-service user support applications based on analysis of an ontology on domain-specific information. An exemplary embodiment concerns a system for providing information in response to a user query that may comprise a natural language parser for identifying relevant terms from the user query, and an ontology analyzer for matching the relevant terms to concepts in an ontology of information related to the user query. The system may include a query processor for refining the user query based on the matched query terms and ontology concepts, and a search engine for identifying from available repositories information relevant to the user query.

AUTOMATED SELF-SERVICE USER SUPPORT BASED ON ONTOLOGY ANALYSIS

BACKGROUND

[0001] Embodiments of the disclosure relate generally to computer software, and more particularly, to automated self-service user support based on an ontology of domain-specific information.

[0002] Automated self-service software applications are commonly deployed by business enterprises to support customers with inquiries and problems concerning their products or services. Such applications may be integrated with call center utilities to minimize the need for live support staff. Self-service support applications may use formal categories to describe the domain in which anticipated inquiries and problems may occur, such as a domain on computer products or financial services. The self-service support applications often assume that inquiry and problem categories are well known and can easily be interpreted by the users. The applications may further assume that the formal categories have universal definitions. However, these definitions are often dependent on the underlying back-end support systems. In addition, most users would prefer to describe their problems or explain their needs in their own terms, using free-form text. The user terms may not align with the formal problem categories or descriptions maintained by the user support system.

[0003] For example, a user may explain a problem in the form of the statement “My laptop fails when I run program XYZ after I have started a backup using program MNO”. Whereas, the user's company back-end technical support systems are typically categorized using very specific terminology, e.g., laptop/desktop, operating system, CPU type, application, program, driver, storage, backup/restore, etc. The self-service applications are thus less effective when their user interfaces are based on system-centric terminology that does not match with the users' terminology.

[0004] Furthermore, even if the users know the specific system-centric terminology, they may not be able to formulate their questions to a degree where satisfactory results can be expected, unless they are aware what constitutes a complete description of a problem. This is due to the fact the users do not know the specific domain that support system uses and are not familiar with the terms, attributes, and relationships in this domain.

[0005] There is thus a need for improved systems and processes for assisting users to formulate self-service inquiries and effectively processing such self-service user inquiries.

BRIEF SUMMARY OF THE DISCLOSURE

[0006] Exemplary embodiments of the disclosure relate to self-service user support applications that are based on the analysis of ontologies on domain-related information. One aspect of the disclosure concerns a system for providing information relating to a user query. The user query may concern a problem that the user experienced with a product or service, a question about a product or service, or other customer/user needs. The system may comprise a natural language processor for identifying relevant terms in the user query and an ontology analyzer for matching the relevant terms to concepts in an ontology related to the user query. The system may further include a query processor for refining the user query using the matching relevant terms and ontology concepts, and a search engine for identifying from a database information relevant to the refined user query.

[0007] Another aspect of the disclosure concerns a computer implemented method for providing information relating to a user query. The method may comprise parsing the user query to identify relevant terms from the user query, matching the relevant terms to concepts in an ontology that are related to the user query, refining the query based on the matching relevant terms and ontology concepts, and searching a database for information relevant to the refined user query.

[0008] Still another aspect of the disclosure concerns a computer program product for providing information relating to a user query. The computer program product comprises a computer readable storage medium having computer readable program

code embodied therewith and configured to parse the user query to identify relevant terms from the user query, and match the relevant terms to concepts in an ontology that are related to the user query. The program code may further be configured to refine the user query based on the matching relevant terms and ontology concepts, and search a database for information relevant to the refined user query.

[0009] The details of the exemplary embodiments of the invention, both as to their structure and operation, are described below in the Detailed Description section in reference to the accompanying drawings, in which like reference numerals refer to like parts. The Brief Summary section is intended to identify key features of the claimed subject matter, but it is not intended to be used to limit the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Figure 1 illustrates a block diagram of an exemplary user support configuration in which aspects of the disclosure may be provided.

[0011] Figure 2 illustrates a block diagram of a representative computer system that may be used in a computer-based user support system, such as the support configuration in Figure 1, for providing aspects of the disclosure.

[0012] Figure 3 illustrates a block diagram of a self-service support system for receiving and analyzing a user query based on an ontology of domain-specific information, and returning information relevant to the user query, according to an embodiment of the disclosure.

[0013] Figure 4 illustrates in more detail another self-service support system for receiving and analyzing a user query based on an ontology of domain-specific information, and returning information relevant to the user query, according to an embodiment of the disclosure.

[0014] Figure 5 illustrates an exemplary user interface menu for entering a user query that may be processed by an embodiment of a self-service support system of the disclosure.

[0015] Figures 6-8 illustrate additional exemplary user interface menus that a self-service support system may present to the user to obtain more details about a user query in order to effectively process the query and identify relevant results.

[0016] Figure 9 is a flowchart of a process for receiving and analyzing a user query based on an ontology of domain-specific information, and returning information relevant to the user query, according to an embodiment of the disclosure.

[0017] Figure 10 is a flowchart of a process that a natural language processor may follow for analyzing a user query to extract relevant terms and parameters, and providing them to an ontology analyzer, according to an embodiment of the disclosure.

[0018] Figure 11 is a flowchart of a process that an ontology analyzer may follow for matching relevant terms from a user query against an ontology of domain-specific information to generate a search query for a search engine, according to an embodiment of the disclosure.

[0019] Figure 12 is a flowchart of a query refining process to determine the completeness and specificity of a user query, further refine the query with user questions and answers, and determine follow-on actions, according to an embodiment of the disclosure.

DETAILED DESCRIPTION OF THE DISCLOSURE

[0020] Exemplary embodiments of the disclosure relate to self-service user support applications based on domain-specific information. As examples, the embodiments of the disclosure may be applicable to customer support systems in information technology (IT), financial services, health care, public sector information, legal services, education, and product marketing, among others. The embodiments may be provided as stand-alone product information or service support systems, or integrated with call center support applications. The embodiments allow a customer or user to enter a problem or need in free-text form. For example, in a financial services environment, a customer may enter an inquiry in the form of “How to set up transfers from a bank account to a brokerage account?”

[0021] The embodiments may receive a user query, identify relevant terms and details from the user query, and if necessary, generate context-specific user questions based on knowledge-driven understanding and intelligence leveraged from the ontology. The query may be refined with user answers to additional questions presented to the user. This is an iterative process in which the query may be re-processed based on the questions and answers. The embodiments may use relevant terms and details extracted from the user query to analyze an ontology of domain-specific information and generate a search query. The embodiments may use the search query to identify information related to the user query from available information repositories or suggest relevant follow-on actions, such as applicable tools or ancillary processes. The

information identified by the embodiments of the invention may include document titles, portions of documents, user manuals, web pages, tools, processes, and links to documents that are relevant to the user query.

[0022] Referring to Figure 1, there is illustrated a block diagram of an exemplary computer configuration in which aspects of the disclosure may be provided. Computer configuration 10 may include multiple client computers 11-12 for accessing a server 13 to receive user support through network 14. Server 13 may host a self-service user support application 15 for assisting customers with their product information or service problems, and providing answers to customer inquiries.

[0023] Figure 2 illustrates a block diagram of a representative computer system that may be used in a user support configuration, such as the configuration 10 in Figure 1, for providing aspects of the invention. Data processing system 200 may include a processor unit 211, a memory unit 212, a persistent storage 213, a communications unit 214, an input/output unit 215, a display 216, and system bus 217. Computer programs are typically stored in persistent storage 213 until they are needed for execution, at which time the programs are brought into memory unit 212 so that they can be directly accessed by processor unit 211. Processor 211 selects a part of memory 212 to read and/or write by using an address processor 211 gives to memory 212 along with a request to read and/or write. Usually, the reading and interpretation of an encoded instruction at an address causes processor 211 to fetch a subsequent instruction, either at a subsequent address or some other address.

[0024] Figure 3 illustrates a block diagram of a self-service support system 300 for receiving a user query, analyzing the query based on an ontology of domain-specific information, and returning information relevant to the user query, according to an exemplary embodiment of the disclosure. As examples, the domain may concern financial services or customer support for IT products. The system 300 may be a software application operating on server 13, and may comprise a user interface system 302 and a query logic system 303. User interface system 302 allows a user 301 to enter, for example, a problem, user need, or inquiry about a supported product or service. The user's problem, need or inquiry may be in the form of a user query statement 304.

[0025] Once the user query statement 304 has been analyzed and processed by the query logic system 303, the query logic system 303 may return to the user 301 documents or other information 311 that the query logic system 303 identifies as being relevant to the user query. The identified information 311 may be returned through user interface system 302. Exemplary user interface menus that the user interface system 302 may present to the user 301 are described below with reference to Figures 5-8.

[0026] The user interface system 302 may present the user with additional questions about the user query 304, as generated by the query logic system 303. The user interface system 302 may also forward user answers to the query logic system 303 during the processing of the user query. A function of the query logic system 303 may be to bridge between the user's free-form description of the user query 304 and the fixed back-end categories based on the use of ontologies, which are analyzed to clarify

the user query statement. The bridging process may include disambiguation, augmentation, and extension of the user query 304, using a domain-specific ontology, to improve specificity and completeness of the user query. The query logic system 303 may assess the resulting user query against knowledge bases, tools, processes, or assets which may support the user's self-service request. The resulting query may be further refined by a query processor based on questions for the user and user answers.

[0027] An ontology is a data structure that formally represents concepts and associated relationships in a technical support domain, e.g., banking, health care, computer products, etc. It may be used not only to define the domain and provide a shared vocabulary, but to provide properties of data in the domain. In the embodiments of the invention, the ontology may initially be prepared by domain and ontology specialists. The query logic system 303 may refine and extend the ontology over time through text mining and ontology matching of additional user inputs, as described below with reference to the ontology editor 409 and ontology builder 410 in Figure 4, and the ontology analysis process in Figure 11.

[0028] An ontology may comprise an ontology model and a knowledge store. The ontology model may be in the form of a Web Ontology Language (OWL) file containing the main domain concepts, which are relatively static. The knowledge store may be in the Resource Description Format (RDF) and conform to the OWL file. As an example of a domain use, the ontology model may capture four main elements of a “perfect” or “complete” query, which may include information on: (1) what the user's need or problem is (e.g., situation, symptoms); (2) where the need or problem occurs (e.g.,

affected components); (3) in what environment (e.g., affected product); and (4) what changed (e.g., user activity that led to the problem).

[0029] The query logic system 303 may deduce semantic meaning in the user query and analyze it against a domain representation. The query logic system 303 may present the user with a series of questions until a search query can be generated that would return a reasonable amount of results. The query logic system 303 may further expose the user to the structure of the ontology through its questions and refine the user query based on user answers to the questions. The refinement is not automated but rather involves the user. It allows a mixed-initiative interaction, where the user is contributing to the formulation of the refined query by answering questions or providing additional information. The search query may be used to perform a meta-search, where it may be sent to a single or multiple heterogeneous back-end databases, knowledge stores and available tools, through multiple search engines. The query logic system 303 may return relevant results in a unified, but categorized and filtered by the query input, list of information.

[0030] In an exemplary embodiment of the invention, the query logic system 303 may comprise a natural language processor 305, ontology analyzer 306, query processor 307, and search engine 308. The natural language processor 305 may analyze a query entered by the user to extract key details from the user query. Query details may include, for example, the type of user problem, what the user was doing when the problem occurred, the environment in which the problem occurred, affected product components, and conditions that have changed as a result of the problem. The output

from the natural language processor 305 may be in the form of relevant substrings (e.g., key terms) and annotations on the relevant substrings. The natural language processor 305 is described in detail below with reference to Figures 4 and 10.

[0031] An ontology analyzer 306 may receive relevant strings and annotations from the natural language processor 305 for analyzing an ontology of domain-specific information related to the user query, and identifying concepts and relationships in the ontology that match the user's problem or need. The ontology analyzer 306 is described in detail below with reference to Figures 3-4 and 11. The query logic system 303 may further comprise a query processor 307 for refining the user query in terms of completeness and specificity. As part of the user query refining process, the query processor 307 may generate additional questions about the user's problem or need that the user interface system 302 may ask the user, and process user answers to these questions. Details on the generation of user questions and the processing of user answers are described below with reference to Figures 4 and 12.

[0032] The query processor 307 may further determine follow-on user service actions and present these service actions to the user, such as suggesting to the user to open a problem record or information request. As an output of the query refinement process, the query processor 307 may generate more specific terms, phrases, and additional information (if missing) that more accurately describe the user's problem or need. The query processor 307 may then provide these terms, phrases, and additional information to the search engine 308. The query processor 307 is further described below with reference to Figures 4 and 12.

[0033] The search engine 308 may identify information relevant to the user query 304 from databases 309 of product and service data, Internet and intranet 310, and other available repositories of information. The query logic system 303 may return the identified information to user 301 through user interface system 302. Search engine 308 may comprise a data search or data analytics program, such as GoogleTM search engine or IBM DB2 Intelligent MinerTM.

[0034] Figure 4 illustrates in more detail an exemplary embodiment of a query logic system 400 for receiving and analyzing a user query based on an ontology of domain-specific information, and returning relevant information to the user. The query logic system 400 may be implemented as layers where each layer is responsible for a set of related processing tasks. For example, a natural language processing layer 402 may be responsible for parsing a user query in natural language, e.g., English. The language processing layer 402 may include a natural language processor 403 for breaking the user query into tokens or key words relating to the problem, such as "failed", "program", "start up", "hang", etc. An example of the natural language processor 403 may be the IBM LanguageWareTM natural language processor. The natural language processor 403 may further parse the key words into a formal representation that is more readily utilized by a computer application.

[0035] In an embodiment of the invention, the natural language processor 403 may perform a lexical analysis of the user's description of a problem or need. It may initially parse the description into paragraphs, sentences and tokens using a break-rules dictionary. It may look up a token in one or more dictionaries to find out more

information about a word, for example, its part of speech (POS). The dictionaries may include both standard linguistic dictionaries containing all words in the selected language and custom dictionaries containing words from a specific domain of knowledge.

[0036] In addition, the natural language processor 403 may perform other types of analysis to determine the nature, format and meaning of the text being processed. For example, the natural language processor 403 may apply a language identification to a body of text to determine the language in which it was written. Lexical analysis may be used to identify words and their attributes as well as determine the part of speech (POS) of each word. Semantic analysis may be employed to determine the contextual meaning of words and phrases, through an understanding of the grammatical structural patterns of a language using a process of relating syntactic structures. Semantic analysis is a phase of natural language processing, following parsing, that involves extraction of context-independent aspects of a sentence's meaning, including the semantic roles of entities mentioned in the sentence, and quantification information, such as cardinality, iteration, and dependency.

[0037] As part of the natural language processing, natural language processor 403 may further include functions for spell-checking, POS disambiguation, normalization (i.e., determining the lemma or canonical form of a word, which is also known as 'morphological analysis'), and anaphora resolution. Normalization is the process of determining a single string representation for a word or term found in text. For normalization of inflectional variance (run, running, runs, etc.), this is traditionally called

the lemma, citation form, or canonical form. Part of Speech (POS) is the linguistic category of a word, such as noun (the run), verb (to run), adjective (runny honey), etc. POS disambiguation is the process of assigning the correct POS to a word and word sense (semantic) disambiguation is the process of identifying which sense of a word is used in any given sentence when the word has a number of distinct senses.

[0038] The natural language processor 403 may refer to a dictionary 404 to obtain the meaning of unfamiliar terms that the user enters. It may look up a thesaurus 405 for synonyms, antonyms, etc., and a lexicon 406 for expressions. A lexicon is a language's vocabulary that includes words as well as common expressions. It is a language's inventory of lexemes. The lexicon includes not only entries for words and phrases, but also lexical relations, syntactic argument structures, and grammatical relations. During the processing of the user query by natural language processor 403, the natural language processing layer 402 may extract key substrings from the user query and provide them to an ontology layer 407. The ontology layer 407 may match these substrings against an ontology of domain-specific information that is related to the user's need or problem.

[0039] As an example, the user may input a query as "Instollation problem on UNIX". The natural language processing layer 402 may perform the following tasks:

- Identify the language of the text as English.
- Recognize the misspelling of "installation" (Instollation).
- Determine the canonical form of "installation" (install).

- Recognize a technical support domain term (UNIX).
- Semantic recognition of an incident (Installation problem).

[0040] Once the natural language process is completed, the ontology layer 407 of the query logic system 400 may begin analyzing the user query against a related domain ontology. The query logic system 400 may iteratively refine the user query based on concepts and relationships that the ontology layer 407 identifies from the ontology, with the goal of increasing the relevance of search results.

[0041] The ontology layer 407 may include an ontology concept matcher 408 for examining terms and relationships in the ontology and matching them against the substrings that were extracted from the user query. The ontology may be visualized as a tree structure where each node in the tree is associated with a term and a connection between two nodes represents a relationship between the terms associated with the connected nodes. Based on the analysis of the ontology, the ontology concept matcher 408 may provide a set of terms from the ontology, and their relationships, that match the relevant substrings extracted from the user query. The matched terms and relationships may be forwarded to the query processing layer 416 for continued processing by the query logic system 400.

[0042] The ontology concept matcher 408 may match a token extracted from the user query to each concept in an ontology (e.g., a node in an ontology structure), and attributes and relationships associated with the concept. Attributes may include sub-components, acronyms, and synonyms of the concept. If there is a match between the

token and a concept in the ontology, the natural language processing layer 402 may annotate the token with the matching concept and its associated pillars. The associated pillars may include situations, activities, products, IT components, etc., that mirror requirements of a “perfect” query.

[0043] If the ontology concept matcher 408 identifies a partial match between a token and an ontology concept, the natural language processing layer 402 may annotate the matching token, and if necessary the query logic system 400 may confirm the partial ontology match with the user. In case the ontology concept matcher 408 identifies multiple ontology concepts that match a token, the query logic system 400 may ask the user to clarify and select the best ontology match through questions for the user. The user can choose the correct words based on the context, pillar and description of the matching token and ontology concepts.

[0044] The ontology layer 407 may further comprise an ontology editor 409 and an ontology builder 410. The ontology editor 409 allows an ontology specialist to edit and create an ontology for a particular domain. An example ontology editor is the open-source Protege editor. Using the Protege editor, an ontology specialist can edit and create an ontology in RDF and OWL script languages. The ontology builder 410 allows an ontology to be updated with additional terms and relationships that the query logic system 400 may identify while processing user queries. The ontology builder 410 thus extends the ontology and refines its contents over time in terms of completeness and accuracy, based on actual user needs and problems and information identified in response to user queries.

[0045] Once the ontology-matching process is completed, the query logic system 400 may forward matched ontology concepts to the query processing layer 416 to assess whether the user query is complete and specific enough for processing or needs refinement with additional user input. The query processing layer 416 may include a term checker 411 for determining the specificity of the key terms extracted from the user query. For each part of the user query, the term checker 411 may determine whether the returned ontology match is sufficiently specific for a search. If the ontology match is not specific enough, the query processing layer 416 may ask the user additional questions to improve the specificity of the ontology match.

[0046] The query processing layer 416 may include a completeness checker 412 to assess whether the user query is sufficiently complete for processing. The completeness checker 412 may determine whether each part of a “perfect” query is satisfied. In an exemplary embodiment, completeness may mean that the query logic system 400 has sufficient information to allow an expert in the field to respond to the user. For example, the query logic system 400 may need information on: a) what the user was trying to do, b) what problem the user encountered, and c) what product or service the user was using. All three aspects of the problem description would be needed to satisfy completeness. For any missing elements, the query processing layer 416 may ask the user further questions with the goal of satisfying each part of a perfect query.

[0047] The query processing layer 416 may include a user question and answer processor 413 for generating user questions and obtaining additional details about the

user's problem or need. Information obtained from the user questions and answers is used to refine the user query, as described above. In an exemplary embodiment, the questions may address areas that would help the system “understand” the problem better, such as “what the problem is?”, “where did the problem occur?”, “in what environment?”, and “what changed?”. The question and answer processor 413 may not always ask all the questions. It may determine which questions to ask depending of the level of specificity and granularity of the user query, to allow the query logic system 400 to reasonably return relevant results. The results may include suitable documents from a search on the user's need or problem in various domains, or relevant follow-on actions such as applicable tools and services.

[0048] In an example embodiment, the user question and answer processor 413 may combine information in the domain ontology with the user's specific problem or need to explain a condition that might have caused the user's problem and to improve the user's trust and relationship. For example, in response to a customer's problem concerning a financial transaction, the question and answer processor 413 may inform the customer of a recent system upgrade, offer to assist the customer by telephone, and provide incentives that may be of value to the customer.

[0049] The question and answer processor 413 may further tailor the questions to display specific words that are appropriate for the user's situation. The questions may include substituted words to allow the system to interact in the context relevant to the user. For example, the user may state in the user query that “the notebook computer failed to boot”. In generating questions for the user, the question and answer processor

413 may substitute the word “laptop” in its repository of questions with the word “notebook”. Through refinement questions for the user, the question and answer processor 413 can expose the ontology structure to the user, thereby allowing the user to learn more about the particular domain that the query logic system 400 is using. Based on the user's answers, the question and answer processor 413 may generate additional user questions to further refine the user query.

[0050] The user question and answer interaction is thus an iterative process for refining the user query with user input. The question and answer processor 413 may capture significant elements of the user query, but may ask further questions and provide answer suggestions based on the domain-specific ontology until the user query is close to a “perfect” query. The query logic system 400 is providing a learning experience to the user to formulate better questions, while the user is potentially providing extra, not yet captured domain-specific knowledge to be added over time to the formal representation of the domain (ontology). Specifically, the question and answer processor 413 refines and optimizes the user's free-form text entry with the purpose of describing the user's need or problem with sufficient specificity and completeness. It leverages concepts and relationships from the ontology to determine the questions and the sequence of presenting the questions to the user.

[0051] The query processing layer 416 may further include a previous query checker 414 to determine whether the user query is similar to a query that has previously been processed by the query logic system 400. Previously handled queries and their solutions may be maintained in a knowledge repository which the query logic system

400 can access. For example, the user query may concern a computer's failure to boot that was caused by the installation of a particular program, and this problem has previously been processed by the query logic system 400 and stored in a database. The previous query checker 414 may look up information relating to this particular problem in the repository and respond to the user with the identified procedure for fixing the problem, without re-processing the user query.

[0052] The query processing layer 416 may comprise other services 415 such as functions to determine follow-on actions for the user or request the user to create a problem report, a purchase request, or an online process/activity. Other services 415 may determine that the user query is more suitable for a service rather than a search. User services may include analyzing system logs, cataloging symptoms, or checking the compatibility of products, models, release versions, etc., for either sales or support functions.

[0053] Once the query processing layer 416 has checked the completeness and specificity of the user query, and refined the user query if necessary, it may send relevant key terms and relationships about the user's need or problem to a search layer 419. The search layer 419 may include a search engine or data analytics program 417 with access to relevant information sources 418 such as databases of product documents. The search engine or data analytics program 417 may also access the Internet or a company's intranet to search for information relevant to the query, in both public and private domains. The search layer 419 may employ a result-ranking utility for ranking the information sources that best match the key terms of the user query and

returning a set of best-matching results. The search layer 419 may then return the best-matching results to the user through the user interface 302.

[0054] Figure 5 illustrates an exemplary user screen of a self-service support system, e.g., system 300, to allow a user to enter a user query or question about products or services, according to an embodiment of the invention. The user screen 501 may include a user interface component 502 in which the user may enter a question or describe a service need or problem. Upon selecting submit button 503, the self-service support system 300 may display a follow-on screen, as shown in Figure 6, which may prompt the user for additional information about the user query. For example, if the user had entered "Change disk drive" in the user interface component 502, then the support system 300 may ask the user to select whether the disk drive is for a logical volume or a physical volume, as illustrated by user choices 604 in Figure 6. In addition, the user interface 302 may display a user interface component where the user does not have to select one of the system choices, but can enter the user's own terms. These terms may then be used for extending the ontology by the ontology builder 410.

[0055] The support system 300 may continue to prompt the user for additional details about the user's need or problem by presenting other user screens until it could determine that the description of the user query is sufficiently complete and specific for a data search. For example, in the disk drive change scenario, the support system 300 may ask the user to specify the type of disk drive change as illustrated by user question 705 in Figure 7. Once a query logic system 400 has a reasonably complete description of the user's need or problem, its components would process the user query, as

described with reference to Figure 4. The results from a search for information relevant to the user query may be displayed to the user as document list 806 in Figure 8.

[0056] Figure 9 is a flowchart of a process for receiving and analyzing a user query based on a domain-specific ontology, and returning information relevant to the user query, according to an embodiment of the disclosure. The process may start at step 901 where a user enters a query or question in free-text form, such as the phrase “change disk drive”. At step 902, a natural language processor, such as natural language processing layer 402 of the query logic system 400, may extract key terms about the user's need or problem from the query, e.g., “change” and “disk drive”.

[0057] An ontology analyzer, such as the ontology layer 407 in the query logic system 400, may analyze a related domain ontology and match the extracted key terms against concepts and relationships in the ontology, at step 903. If the query logic system 400 determines that the user query is not complete or specific enough for a search, then the query processing layer 416 may generate additional questions about the user's need or problem, per step 904. The query processing layer 416 may further refine the user query, at step 905, to make it more complete and specific for a search, based on the user's answers to these questions. The query processing layer 416 may reprocess the refined user query as shown by the loop back from step 905 to step 902.

[0058] At step 906, the query processing layer 416 in the query logic system 400 may generate a search query that includes search terms, concepts and annotations, based on the refined user query. The query logic system 400 may provide the search query to

a search layer 419 for identifying information relevant to the user's problem or need, using the search query, at step 907. The search layer 419 may use a search engine or data analytics program 417 to search repositories 418 of product and service information. In addition to presenting the search results to the user at step 908, the query logic system 400 may solicit user feedback concerning the relevancy of the resulting information, per step 909.

[0059] Figure 10 is a flowchart of an exemplary process that a natural language processing layer 402 may follow for analyzing a user query to extract relevant terms and details about the user's need or problem, and providing them to an ontology layer 407. The natural language processing layer 402 may start at step 101 by determining the language of the user query, e.g., English. It may parse the user query to extract verbs and nouns from the query, at step 102. The verbs generally relate to the actions that the user is interested in performing, and the nouns generally correspond to the objects involved (such as a particular product, situation, or a technology component). The natural language processing layer 402 may ignore connecting words in the statement, e.g., "about", "in" and "by" (step 103). It may perform other tasks on the user query such as text segmentation, tokenization, disambiguation, spell-checking and normalization, per step 104. These tasks were previously described with reference to Figure 4.

[0060] The natural language processing layer 402 may identify entities (e.g., key terms) and relationships in the user query, at step 105 (for example, "failed", "after", and "installation"). If there are terms in the user query that the natural language processing

layer 402 does not recognize, it may refer to a dictionary, thesaurus, or lexicon at step 106 to help determine their meaning and the user's intent. Once the natural language processing layer 402 has extracted relevant terms from the user query, it may forward these terms to an ontology analyzer (such as ontology layer 407) for analyzing a domain-specific ontology related to the user query, and matching the terms against concepts in the ontology, per step 107.

[0061] Figure 11 is a flowchart of an exemplary process that an ontology layer 407 may follow for matching relevant terms from a user query against concepts and relationships in a domain-specific ontology. The process may start at step 111 where an ontology concept matcher 408 matches a token extracted from the user query to each concept in an ontology (e.g., a node in an ontology structure), and attributes and relationships associated with the concept. Attributes may include sub-components, acronyms, and synonyms of the concept. If there is a match between the token and a concept in the ontology, the natural language processing layer 402 may annotate the token with the matching concept and its associated pillars, per step 112. The associated pillars may include situations, activities, products, IT components, etc., that mirror requirements of a "perfect" query.

[0062] If the ontology concept matcher 408 identifies a partial match between a token and an ontology concept, the natural language processing layer 402 may annotate the matching token, and if necessary the query logic system 400 may confirm the partial ontology match with the user, per step 113. In case the ontology concept matcher 408 identifies multiple ontology concepts that match a token, the query logic system 400

may ask the user to clarify and select the best ontology match through questions for the user, at step 114. The user can choose the correct words based on the context, pillar and description of the matching token and ontology concepts, per step 115.

[0063] The query logic system 400 may perform the process illustrated in Figure 11 for each token extracted from the user query and may iteratively refine the user query based on concepts and relationships from the ontology, with the goal of increasing the relevance of search results. Based on the analysis of the ontology, the ontology concept matcher 408 may output a set of terms from the ontology, and their relationships, that match the relevant substrings extracted from the user query. The ontology layer 407 may provide the matched terms and relationships the query processing layer 416 for continued processing by the query logic system 400, at step 116.

[0064] As described above with reference to Figure 4, the ontology layer 407 may comprise an ontology builder 410 for updating a domain ontology with terms and relationships that the query logic system 400 identifies while processing user queries, per step 117. The ontology builder 410 thus extends the ontology and refines its contents over time in terms of completeness and accuracy, based on actual user needs and problems and information identified in response to user queries. Once the ontology-matching process is completed, the query logic system 400 may forward matched ontology concepts to a query processing layer 416 to refine the query with additional user input if necessary.

[0065] Figure 12 is a flowchart of an exemplary query refining process that a query processing layer 416 may follow to determine the completeness and specificity of a user query. The process may refine the query with additional user questions and answers, and determine follow-on user actions. A query processing layer 416 in the query logic system 400 may start at step 121 to determine whether ontology matches for a user query, as returned from an ontology concept matcher 408, are specific enough for a search of related information. If the matches are not sufficiently specific, the question and answer processor 413 may generate and ask the user additional questions to clarify the user's need or problem, as previously described with reference to Figure 4.

[0066] The query processing layer 416 may further determine whether the user query is sufficiently complete for processing, per step 122. For example, the completeness checker 412 of the query logic system 400 may determine whether each part of a “perfect” query is present in the user query. For any missing key descriptors, the query logic system 400 may ask the user additional questions with the goal of satisfying each part of a “perfect” query, at step 123. The query processing layer 416 may refine, at step 124, the specificity and completeness of the key descriptors with the additional information that the user supplies in response to the questions.

[0067] At step 125, the query processing layer 416 may conclude, based on information extracted from the user query, that the user query is more suitable for a service rather than a search for information, e.g., a product replacement due to a defect. In that case, the query logic system 400 may direct the user to a service handling system rather than continuing with the information search. Furthermore, if the query

processing layer 416 determines that user actions are needed to accurately determine the user's need or problem, or relevant information, it may formulate follow-on actions and present them to the user at step 126.

[0068] In case the query processing layer 416 determines that the user query descriptors are sufficiently complete and specific, it may forward the descriptors to the search layer 419 in the query logic system 400, at step 127. The search layer 419 may use a search engine or data analytics program 417 to search information sources 418 such as databases, intranets, or the Internet. The search engine or data analytics program 417 may identify from the sources 418 information related to the user query and return the identified information to the user through user interface 302.

[0069] The subject matter described above is provided by way of illustration only and should not be construed as limiting. Various modifications and substitutions of the described components and operations can be made by those skilled in the art without departing from the spirit and scope of the disclosure defined in the following claims, the scope of which is to be accorded the broadest interpretation so as to encompass such modifications and equivalent structures. As will be appreciated by those skilled in the art, the systems, methods, and procedures described herein can be embodied in a programmable computer, computer executable software, or digital circuitry. The software can be stored on computer readable media. For example, computer readable media can include a floppy disk, RAM, ROM, hard disk, removable media, flash memory, a "memory stick", optical media, magneto-optical media, CD-ROM, etc.

[0070] Accordingly, aspects of the disclosure may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a “circuit,” “module” or “system.” Furthermore, aspects of the disclosure may take the form of a computer program product embodied in one or more computer readable medium(s) having computer readable program code embodied thereon.

[0071] Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples (a non-exhaustive list) of the computer readable storage medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination of the foregoing. In the context of this document, a computer readable storage medium may be any tangible medium that can contain, or store a program for use by or in connection with an instruction execution system, apparatus, or device.

[0072] A computer readable signal medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electro-magnetic, optical, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device.

[0073] Program code embodied on a computer readable medium may be transmitted using any appropriate medium, including but not limited to wireless, wireline, optical fiber cable, RF, etc., or any suitable combination of the foregoing.

[0074] Computer program code for carrying out operations for aspects of the disclosure may be written in any combination of one or more programming languages, including an object oriented programming language such as Java, Smalltalk, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an

external computer (for example, through the Internet using an Internet Service Provider).

[0075] Aspects of the disclosure are described above with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the disclosure. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0076] These computer program instructions may also be stored in a computer readable medium that can direct a computer, other programmable data processing apparatus, or other devices to function in a particular manner, such that the instructions stored in the computer readable medium produce an article of manufacture including instructions which implement the function/act specified in the flowchart and/or block diagram block or blocks.

[0077] The computer program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other devices to cause a series of

operational steps to be performed on the computer, other programmable apparatus or other devices to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0078] The flowchart and block diagrams in the figures described above illustrate the architecture, functionality, and operation of possible implementations of systems, methods and computer program products according to various embodiments of the disclosure. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that, in some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

CLAIMS

What is claimed is:

1. A system for providing information related to a user query, comprising:
a natural language processor for identifying relevant terms from the user query;
an ontology analyzer for matching the relevant terms to concepts in an ontology related to the user query;
a query processor for refining the user query based on the matching relevant terms and ontology concepts; and
a search engine for identifying from a database information relevant to the refined user query.
2. The system of claim 1, wherein the natural language processor dissects the user query into tokens for processing by a computer application.
3. The system of claim 1, wherein the user query comprises words and phrases, and the natural language processor determines contextual meaning of the words and phrases based on a semantic analysis.
4. The system of claim 1, wherein the natural language processor performs at least one task of language identification, text segmentation, tokenization, disambiguation, spell-checking, and normalization on the user query.

5. The system of claim 1, wherein the ontology analyzer annotates the matched relevant terms with ontology information and confirms partial and multiple ontology matches with the user.

6. The system of claim 1, wherein the ontology analyzer comprises an ontology builder for extending the ontology with relevant terms and relationships identified by the system in processing the user query.

7. The system of claim 1, wherein the query processor determines whether the user query is complete and specific, and refines the user query with user questions and answers to make the user query more complete and specific.

8. A computer implemented method for providing information related to a user query, the method comprising:

parsing the user query to identify relevant terms from the user query;

matching the relevant terms to concepts in an ontology related to the user query;

refining the user query based on the matching relevant terms and ontology concepts; and

searching a database for information relevant to the refined user query.

9. The method of claim 8, further comprising asking the user questions to achieve a complete query, and refining the query based on user answers to the questions.

10. The method of claim 9, wherein the questions and answers are based on concepts and relationships in the ontology.
11. The method of claim 8, wherein refining the user query comprises determining whether the user query is complete and specific.
12. The method of claim 8, wherein refining the user query comprises determining information on the nature, environment and components affected by the user need.
13. The method of claim 8, further comprising generating a search query based on the refined user query and processing the search query by a search engine.
14. The method of claim 8, further comprising determining follow-on user actions associated with the user query.

15. A computer program product for providing information related to a user query, comprising a computer readable storage medium having computer readable program code embodied therewith, the computer readable program code comprising:

program code configured to parse a user query to identify relevant terms from the user query;

program code configured to match the relevant terms to concepts in an ontology related to the user query;

program code configured to refine the user query based on the matching relevant terms and ontology concepts; and

program code configured to search a database for information relevant to the refined user query.

16. The computer program product of claim 15, wherein the computer readable program code comprises program code configured to ask the user questions to achieve a complete query, and refine the user query based on user answers to the questions.

17. The computer program product of claim 15, wherein the computer readable program code comprises program code configured to determine information on the nature, environment and affected components concerning the user query.

18. The computer program product of claim 15, wherein the user query is in a natural language.

19. The computer program product of claim 15, wherein the user query concerns a product.

20. The computer program product of claim 15, wherein the user query concerns a service.

Page 1 of 10

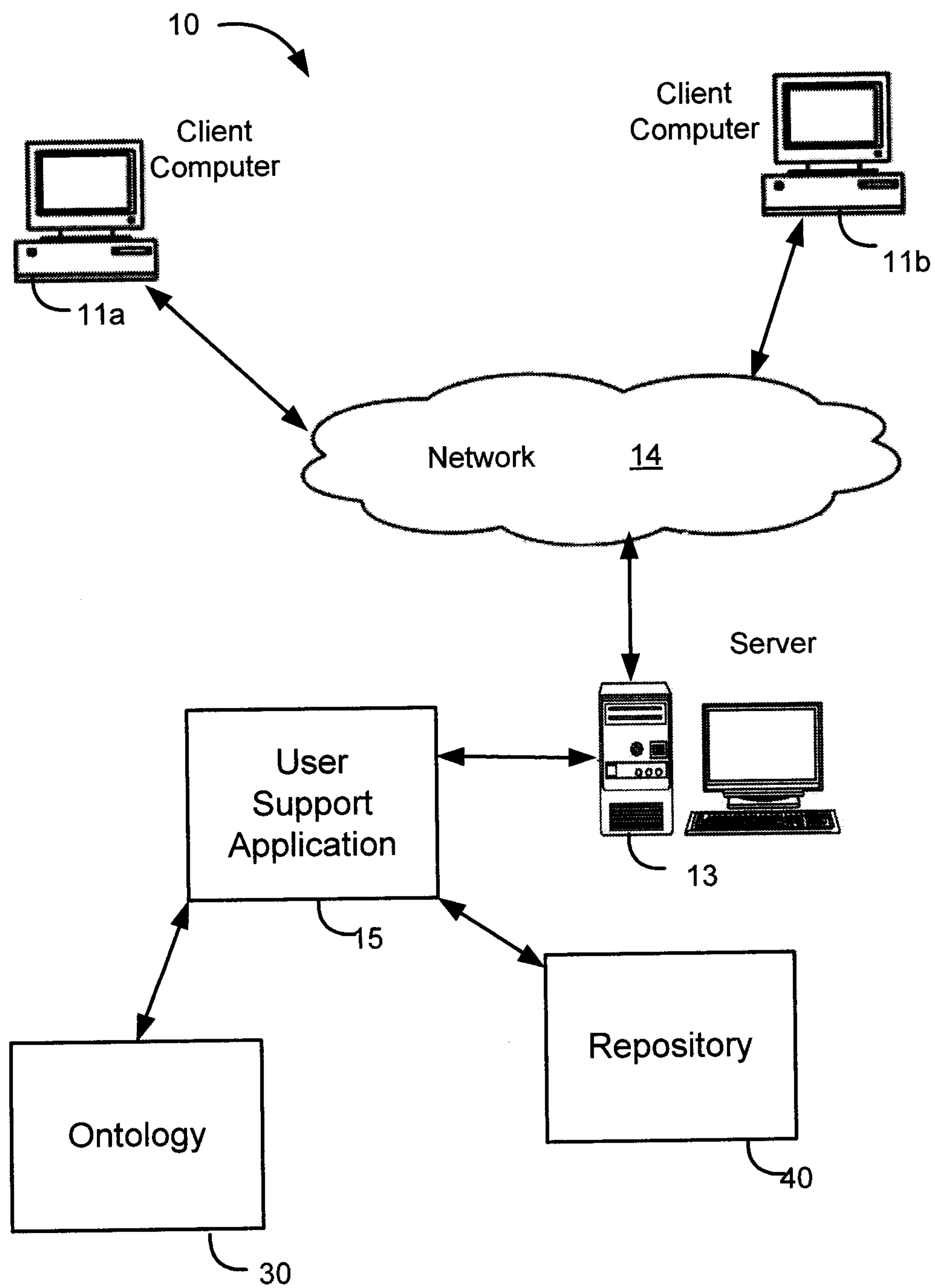


Figure 1

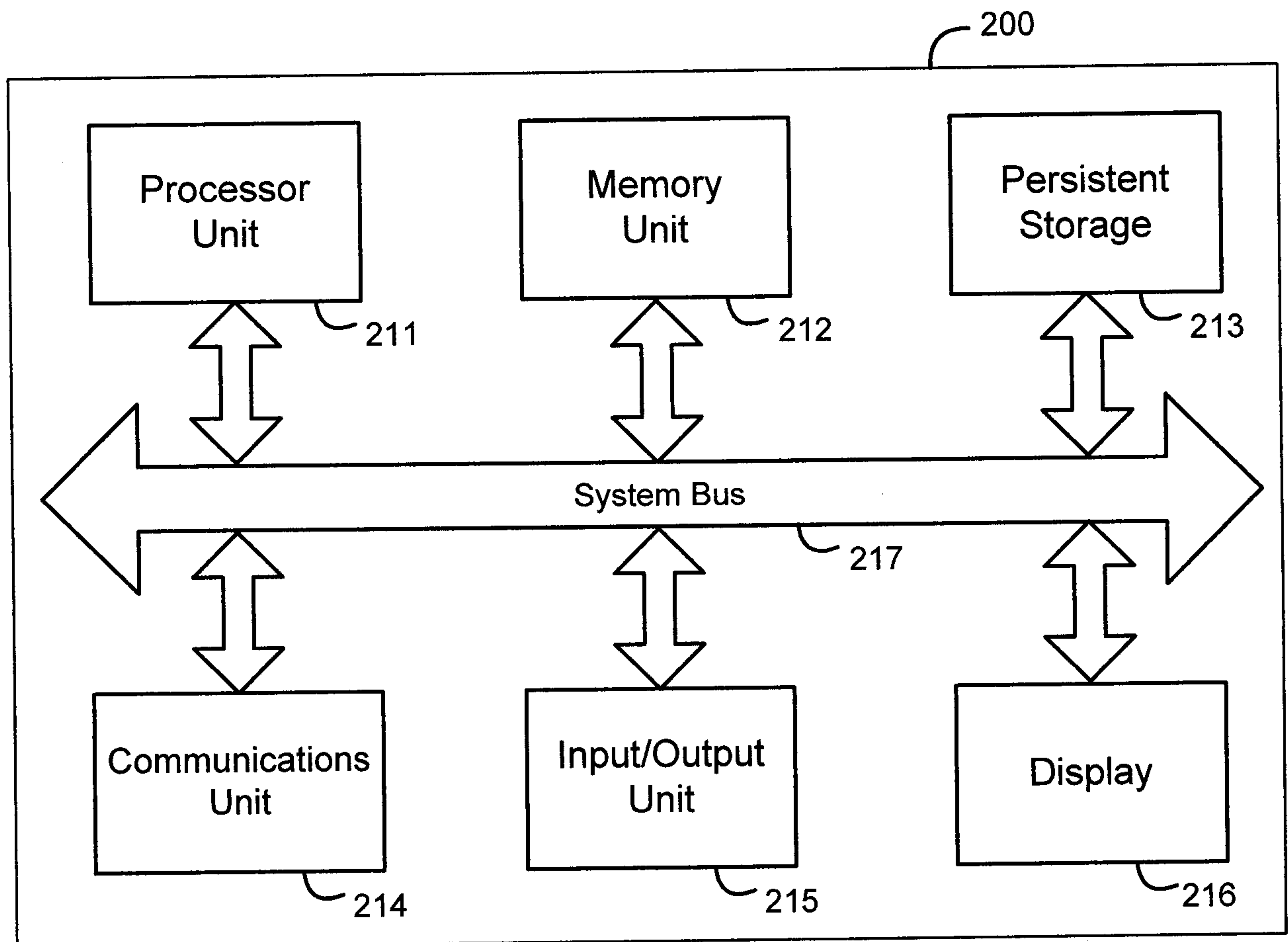


Figure 2

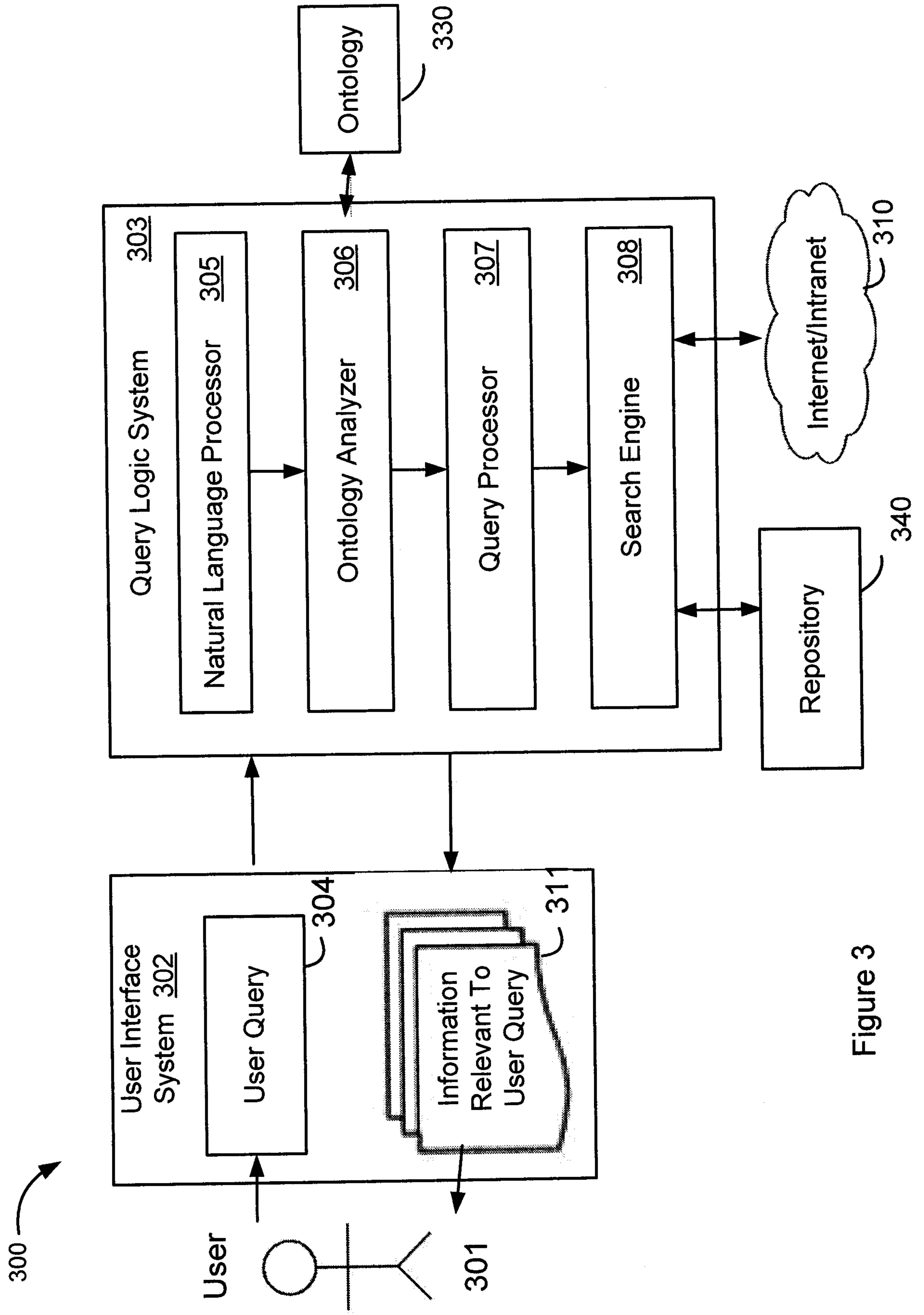


Figure 3



User Support System 501

Enter your question or problem

Change disk drive _

Submit

502

503

Figure 5

User Support System 601

Enter your question or problem

Change disk drive _

Submit

602

603

IT Component

You said "disk drive". Are you dealing with

☐ Logical volume

☐ Physical volume

☐ Other

)

605

Figure 6

User Support System 701

Enter your question or problem

Change disk drive _

Activity

What type of change?

☐ Adding disk drive

☐ Removing disk drive

☐ Replacing disk drive

☐ Other

)

705

Figure 7

User Support System 801

Activity

What type of change?

☐ Adding disk drive

☐ Removing disk drive

☐ Replacing disk drive

☐ Other

)

805

Recommended Documents:

- Removing and Replacing a Fixed Disk
- Synchronizing Disk Names

)

806

Figure 8

Page 7 of 10

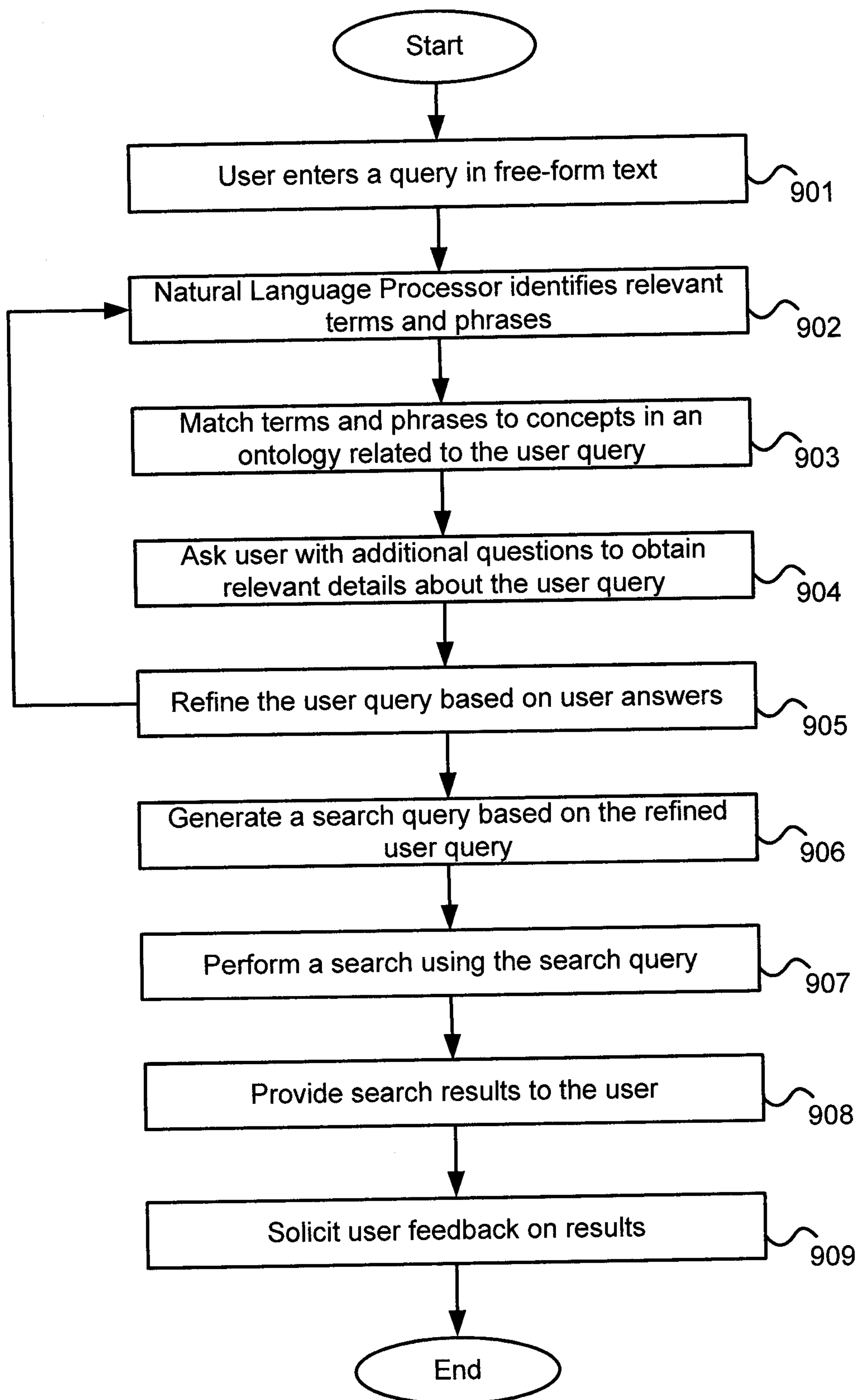


Figure 9

Page 8 of 10

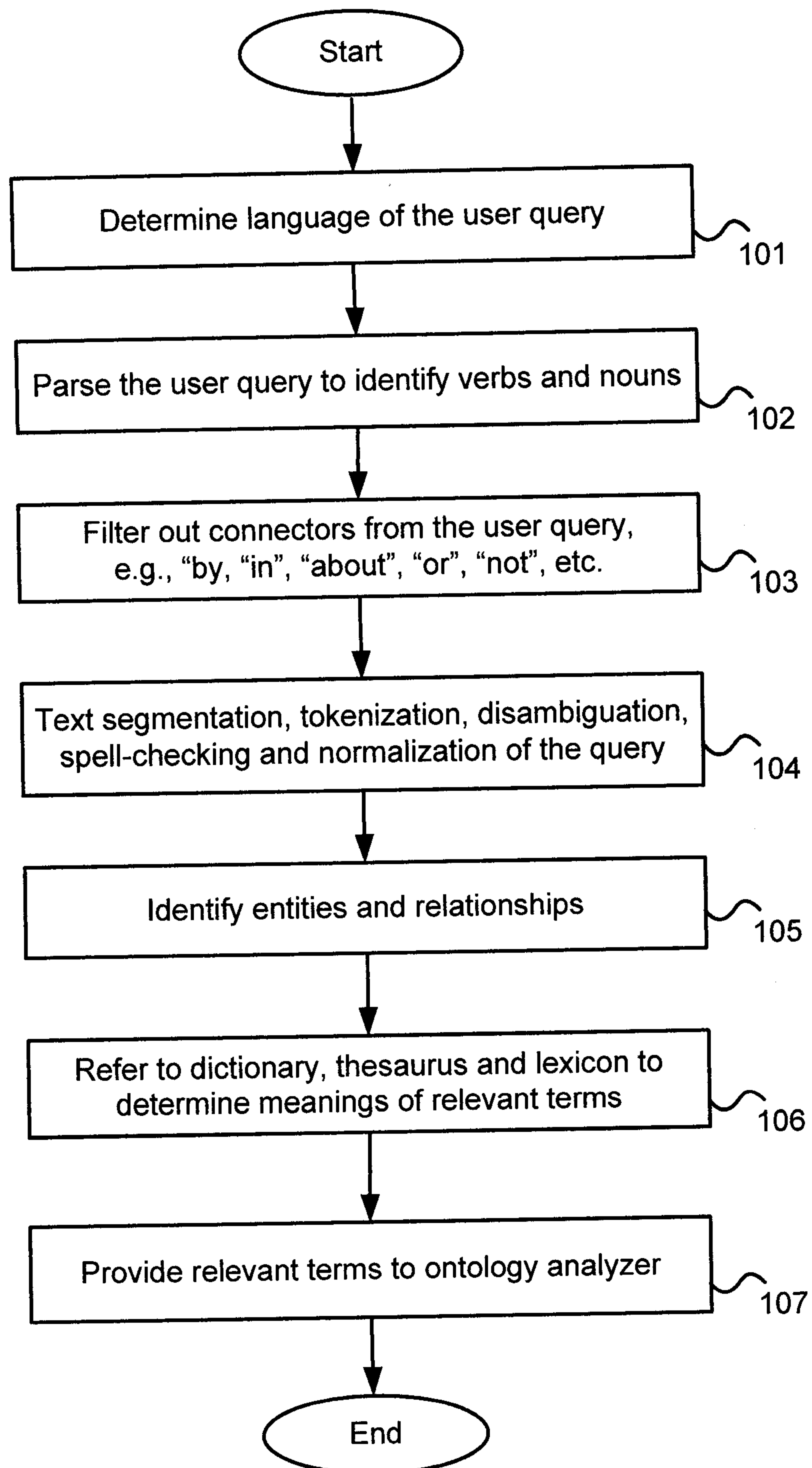


Figure 10

Page 9 of 10

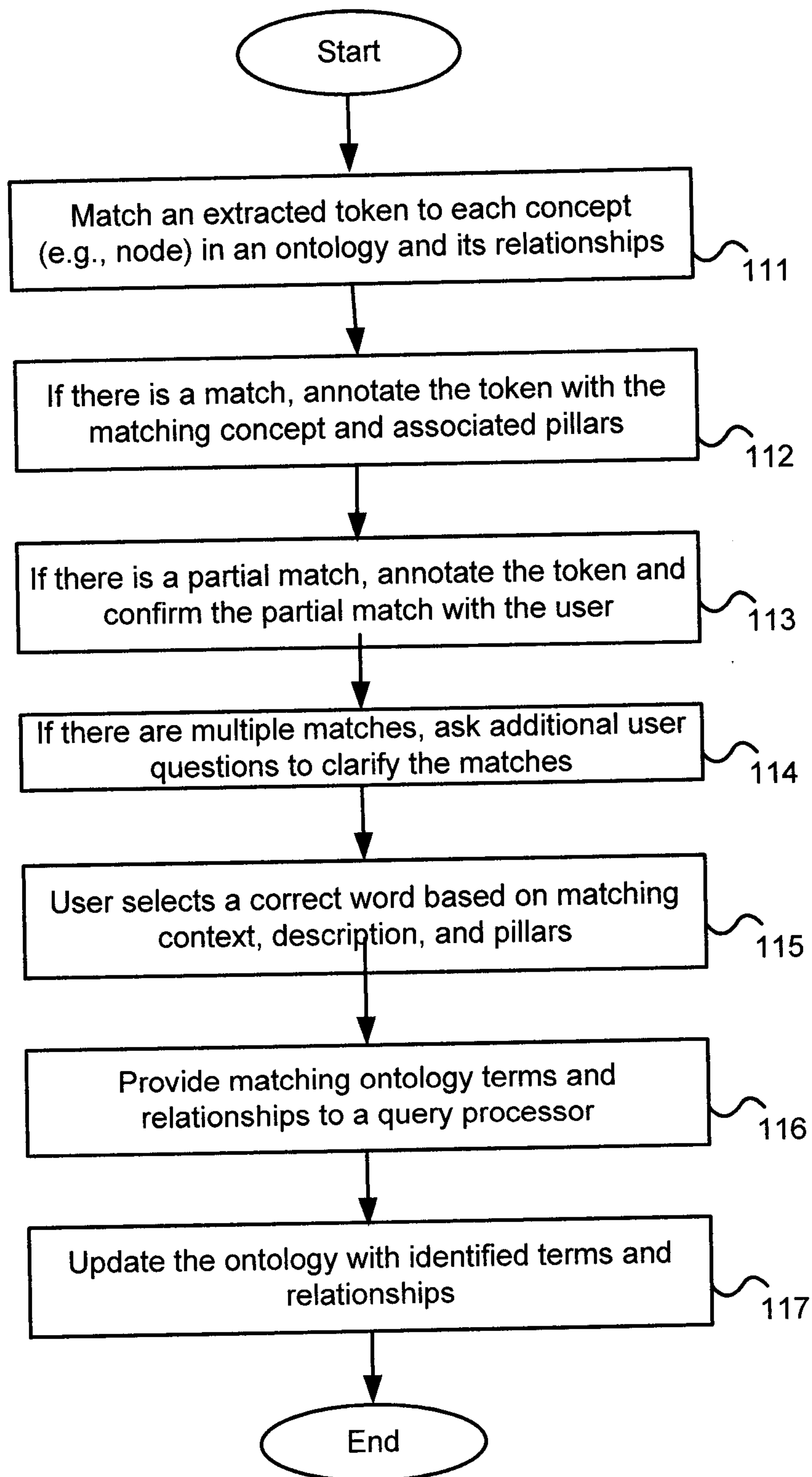


Figure 11

Page 10 of 10

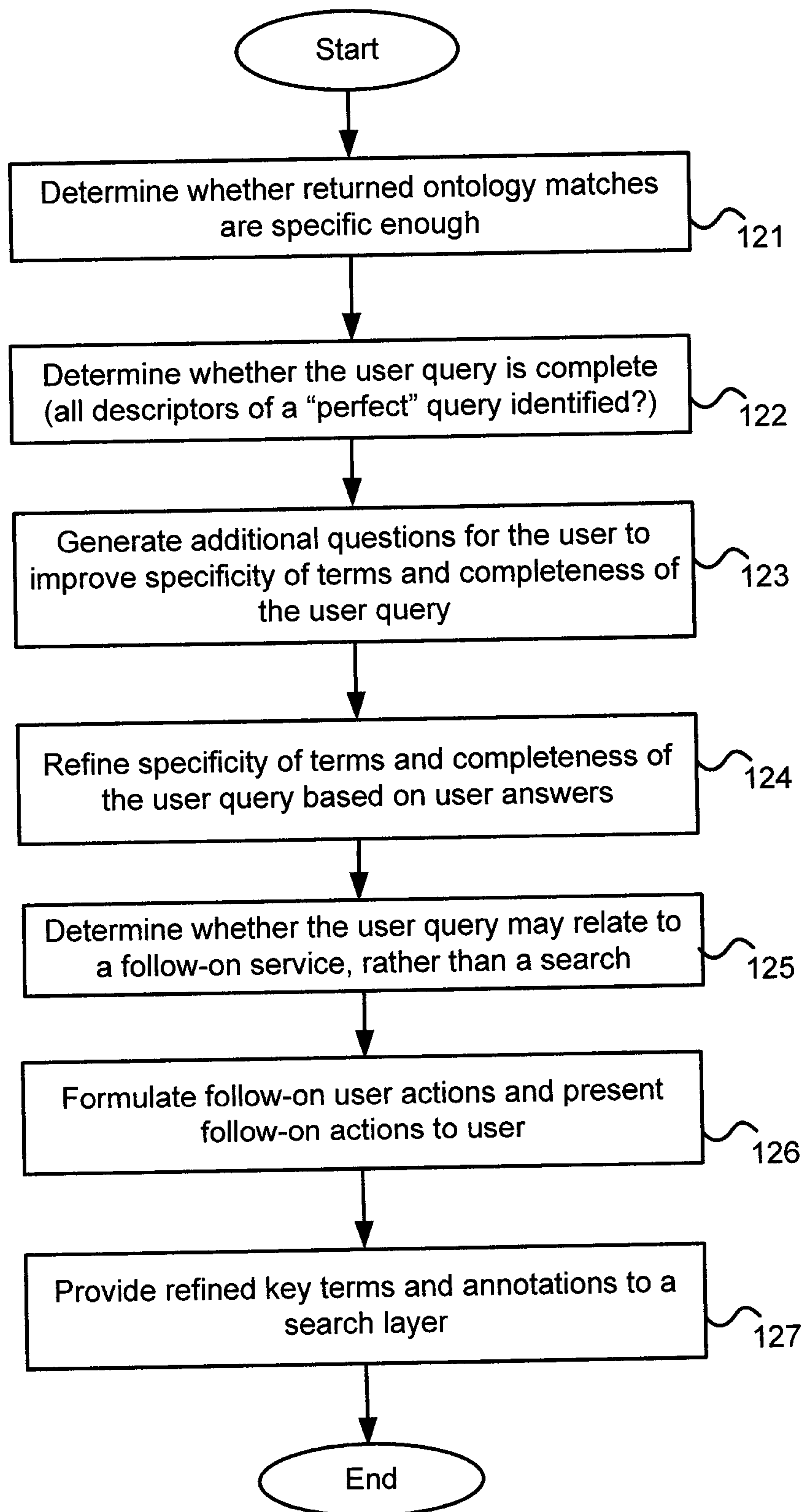


Figure 12

