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

(19) **World Intellectual Property Organization**
International Bureau

(43) **International Publication Date**
23 July 2009 (23.07.2009)

PCT

(10) **International Publication Number**
WO 2009/090498 A2

(51) **International Patent Classification:**
G06F 17/30 (2006.01)

(21) **International Application Number:**
PCT/IB2008/003982

(22) **International Filing Date:** 30 October 2008 (30.10.2008)

(25) **Filing Language:** English

(26) **Publication Language:** English

(30) **Priority Data:**
60/983,844 30 October 2007 (30.10.2007) US

(71) **Applicant** *(for all designated States except US)*: **TRANSFORMER SOFTWARE, LTD.** [IL/IL]; 7 Jabotinsky Street, 52520 Ramat Gan (IL).

(72) **Inventors; and**
(75) **Inventors/Applicants** *(for US only)*: **ZOARETS, Jacob** [IL/IL]; 3404 Street, 68076 Tel-Aviv (IL). **ZOARETS, Roy** [IL/IL]; 3 Zamenof Street, 59473 Bat-Yam (IL). **MI-UNI, Shalom** [IL/IL]; 8 Gadera Street, 49724 Petach-Tikva (IL).

(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, 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, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL,

[Continued on next page]

(54) **Title:** KEY SEMANTIC RELATIONS FOR TEXT PROCESSING

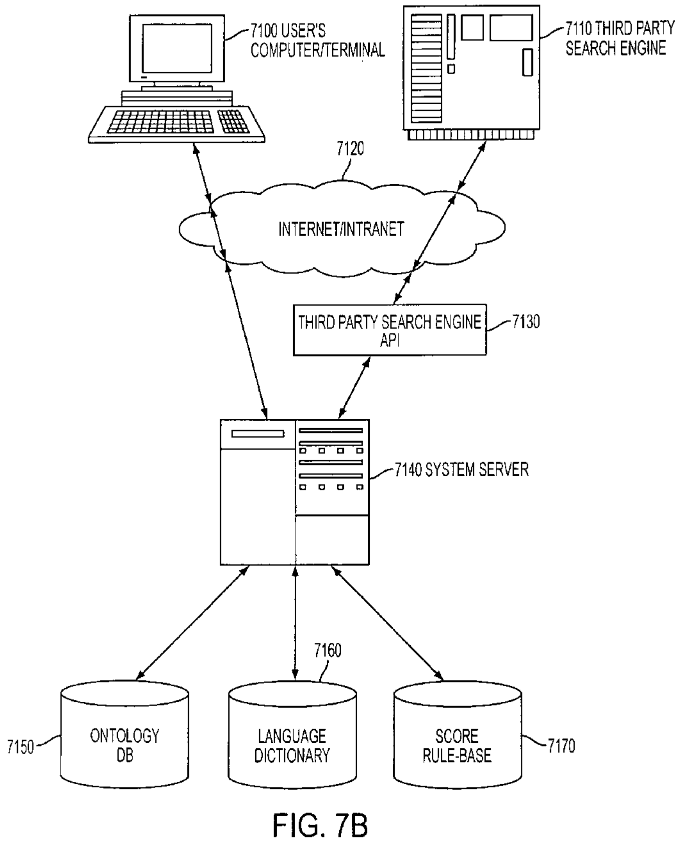


FIG. 7B

(57) **Abstract:** A computer system, method, and computer readable medium for semantically analyzing natural language sentences. The system parses a sentence utilizing semantics-based methodologies, including event-based semantics. The system identifies key semantic relations (KSRs) in the sentence and ranks them in order of semantic importance. An algorithm based on KSR matching is proposed, allowing for more accurate full-text searches (FTS). KSR matching has many potential applications, including various types of FTS, targeted contextual advertisements, text matching, helpdesks, and others.

NO, PL, PT, RO, SE, SI, SK, 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*

KEY SEMANTIC RELATIONS FOR TEXT PROCESSING

CROSS-REFERENCE TO RELATED PATENT APPLICATION

[01] This application claims the benefit of U.S. Provisional Application No. 60/983,844, filed on October 30, 2007, the disclosure of which is incorporated herein in its entirety by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[02] This invention broadly relates to linguistic analysis of text for enabling semantic matching among queries and text, with applications for performing full text searches and other tasks related to natural language processing.

2. Description of the Related Art

[03] A need for linguistic analysis of text arises in a broad range of applications, such as Internet full-text search (hereinafter "FTS"), enterprise FTS, speech recognition, desktop search, and online advertising. FTS, generally speaking, is a process that attempts to find matches between a query and a set of full-text documents.

[04] Search is a gigantic business nowadays. The Internet has turned FTS into a ubiquitous household activity. But most people find it quite difficult to find the most relevant documents using the tools provided by current search engines. The same is true, though to a lesser degree, with professional searches in specialized corpuses (e.g., LexisNexis) and intra-enterprise databases.

[05] Currently, the great majority of FTS systems are based on “keyword” search. Keyword search is typically a single-word or single-term search, and the search retrieves those documents that contain one or more occurrences of the specified search terms. This is often enhanced by “logical operators” for precluding occurrences of words and for combining search criteria, and by “metrical operators” for demanding that different terms occur in proximity. Some systems are also capable of searching for phrases (e.g., “White House”) or synonyms of the specified keywords, but essentially the search is still limited to sets of a single unit of text.

[06] A complementary technology is text mining, wherein documents are analyzed to determine which subjects they deal with. Applications include (a) classification, that is, determining under which topics in some taxonomy the document should be listed; (b) extraction of important people, places, and other entities referred to in the document; and (c) clustering of a mass of documents into ad hoc “birds of a feather” groupings. These tools are used to enhance textual search by facilitating search by subject or retrieval of “similar” results.

[07] Keyword search is simple to use but does not allow a user to specify exactly what the user is looking for in the search. The average Internet search on Yahoo or Google is only two or three words long, though this number is growing slowly. The default for most search engines is to assume that queries and documents are just “bags” of words, with the word order—and any interrelation—assumed to be irrelevant. Thus, the two queries “Man bites dog” and “Dog bites man” would retrieve virtually the same set of documents, despite the fact that the difference between them can be very significant when looking for news stories or court cases.

[08] Searching for complete phrases, on the other hand, such as “Dog bites man” (placing the words within quotation marks, in some interfaces) is only of partial help, since very

many variations share the same meaning: "Man bitten by dog," "Man bitten by stray dog," "A German shepherd, abandoned by its owner, felt threatened earlier today and attacked an elderly gentleman, biting him on the thigh." Thus, phrase-based search is likely to miss many relevant documents. The crux of search difficulties is that information retrieval is plagued by two opposing types of failure: incomplete retrieval, where relevant documents are not found, or are found but are not placed at the forefront of the list of retrieved documents; and imprecise retrieval, where irrelevant items are returned and are placed before the more relevant ones. Any attempt to improve retrieval is likely to decrease precision, and vice-versa.

[09] Neither the "bag of words" approach nor an "exact phrase" option works well enough, often enough. Therefore, many search engines rank retrieved documents according to the frequency, proximity, and order of the occurrences of query keywords in the retrieved documents. Nonetheless, serious problems remain. Ranking based on keyword distances and order does not faithfully reflect the true relations sought in the query: the same three query words appear contiguously in both "Man bites dog" and "Dog bites man." Furthermore, the order—but not the meaning—is reversed when the text is written in passive voice, for example, "Man bitten by dog" and "Dog bitten by man."

[10] One of keyword search's main drawbacks is that it provides an often-overwhelming number of irrelevant hits (i.e., retrieved references). For example, in a typical Google® Internet search, one may find hundreds of thousands or even millions of matching documents. The huge number of hits makes it very difficult to pinpoint the most relevant documents, often leaving the end-user frustrated and confused by the search results.

[11] Thus, there is a need for a technological breakthrough to overcome the limitations of FTS (including keyword search and text mining technologies) that result from the ignoring of semantic relations in queries.

[12] Keyword-based search engines seek matches between keywords found in the query and a given database of documents. The database of documents is known as a corpus. A corpus is composed of three layers: a files/paragraphs/document layer, a sentences layer, and a words (keywords) layer. The lowest is the lexical (word) or phrase level. Most search technologies concentrate on word-level text processing (looking for “dog”), which can include stemming, looking for all words with the same base form (“dogs” and perhaps “dogged”), and/or expansion with synonyms and related words (e.g., “canine” and perhaps “collie”). But individual words are often ambiguous (e.g., “dog” is both a noun and a verb), and the same set of words can be used to express contrary ideas (as with “Man bites dog” vs. “Dog bites man”).

[13] At the other end of the spectrum, there is the document level. The meaning of a document (i.e., a newspaper article, technical report, or court decision) is richer than the sum of its constituent sentences and words. Tools such as document classification and text mining exist for textual analysis on this level, but though they may correctly capture the subject matter, they inevitably completely miss many embedded details.

[14] Between the word and document levels lies the sentential level. Sentences clearly have greater meaning than the sum of their constituent words, but at the same time have less meaning than the whole document.

[15] Existing technologies concentrate on word and document analysis, leaving aside sentence analysis. It has become clear, however, that this missing dimension is of paramount

importance for the ultimate ability to cover the whole spectrum of information contained in a text. On the sentence level, there are both syntactic and semantic elements that come into play. Even trivial textual analysis can help. If there is semicolon or parenthesis between two words in a phrase, then that occurrence is irrelevant. That alone would go a long way to the frustratingly many highly ranked irrelevant results returned by search engines like Google® when presented with a phrasal query.

[16] With syntactic analysis, much more can be achieved. Even with just "part of speech tagging" and "shallow parsing," it is readily apparent that "intermediate" does not modify "machine" in the query, "Optimization of intermediate and machine code." By matching syntax trees of a clause containing the query and of a collection of fully parsed documents, it is possible to achieve much more precise search results.

[17] Unfortunately, syntax-based search has not lived up to hopes. Though it helps for noun phrases and the like, it is of limited value when queries involve predicates.

[18] In the following, a few examples are presented demonstrating sentence analysis of queries and the related relevant queries suggested by sentence analysis.

[19] First, consider the following search query: "pupils punish their teachers." This search query is an example of a verb phrase (hereinafter "VP"). Similar queries include "pupils/students punish *their* teachers," "how *can I/one/we* punish *our* teachers *for* ill behavior," "teacher punished *for* ill behavior," "punished teachers," "teachers punished *by* *," "* punish teachers," and "misbehavior *of* teachers." Keyword-based search results returned by Google®, MSN®, and Yahoo® for queries such as these include "Teachers punish students...," "How to deal with misbehaving children," "Pupils with handicaps are being punished...," "Child abuse,"

and "Punish by reward." The keyword search engine recalls/retrieves all the documents that include the query's keywords and ranks the documents by a certain FTS logic/algorithm. Frequently, numerous irrelevant hits are returned, as in the example above.

[20] Second, consider the following search query: "optimization of intermediate and machine code." This search query is an example of a noun phrase (hereinafter "NP"). In this query, certain combinations of keywords have a significant meaning, while other combinations are irrelevant to the meaning behind the query.

[21] For example, the combination "optimization code" is relevant. There is a significant semantic attribute-head relationship between keywords "optimization" and "code." This is despite the fact that the FTS distance, measured by the number of separating words, within the query sentence is maximal.

[22] Now consider the combination "intermediate machines." This combination is irrelevant; there is neither an attribute-head relation nor any other key semantic relation (hereinafter "KSR") in the query between the keywords "intermediate" and "machine." This is despite the fact that the FTS distance within the query sentence is minimal. In the above example, it is shown that pairs or larger sets of meaningful words often may be located in different parts of the sentence. Thus, the proximity assumption typically used in existing keyword-based search engines is not always accurate. Proximate combinations of keywords are not always meaningful, while distant combinations of keywords sometimes have great relevance.

[23] The best technology available today for formal sentence-based analysis is *semantic parsing*, which converts a sentence into some relatively language independent representation of its meaning (an "interlingua").

[24] Once the query and the sentences in the corpus have been semantically parsed, this data can be used for search. Ideally, "semantic matchers" would be designed that would search a corpus for sentences that are semantically similar to a query, that is, those which share a lot of structure. But there are simpler, more localized ways in which the semantic information embedded in the semantically-parsed documents can be exploited to greatly enhance the quality of search results.

SUMMARY OF THE INVENTION

[25] Exemplary embodiments of the present invention may overcome the above disadvantages and other disadvantages not described above. The present invention is not necessarily required to overcome at least some of the disadvantages described above, and the exemplary embodiments of the present invention may not overcome at least some of the problems described above. The appended claims should be consulted to ascertain the true scope of the invention.

[26] Exemplary embodiments of the present invention relate to linguistic analysis of text for purposes of improving natural language processing tasks, including FTS.

[27] According to an exemplary embodiment of the present invention, sentence-level linguistic analysis of queries is performed in order to improve the precision of FTS or other natural language processing tasks.

[28] According to an exemplary embodiment of the present invention, a method for using key semantic relations in full text search is provided. The method includes performing a keyword search using a search query, identifying at least one key semantic relation in the search query, identifying at least one key semantic relation in at least one document or document

snippet in output of the keyword search that matches a key semantic relation identified in the search query, and re-ranking the keyword search output by using the at least one identified matching key semantic relation in the at least one document or document snippet.

[29] According to yet another exemplary embodiment of the present invention, a computer program product for enabling a computer to use key semantic relations in full text search is provided. The computer program product includes software instructions for enabling the computer to perform predetermined operations and a computer readable medium bearing the software instructions. The predetermined operations include performing a keyword search using a search query, identifying at least one key semantic relation in the search query, and identifying at least one key semantic relation in at least one document or document snippet in output of the keyword search that matches a key semantic relation identified in the search query.

[30] According to another exemplary embodiment of the present invention, a computer system adapted to use key semantic relations in full text search is provided. The computer system includes a processor and a memory including software instructions. The software instructions cause the computer system to perform a keyword search using a search query, to perform an identification of at least one key semantic relation in the search query, and to perform an identification of at least one key semantic relation in at least one document or document snippet in output of the keyword search that matches a key semantic relation identified in the search query.

[31] According to still yet another exemplary embodiment of the invention, a method for using key semantic relations in full text search is provided. The method includes identifying at least one keyword and at least one key semantic relation in a search corpus, creating an index

of features using the identified at least one keyword from the search corpus and the identified at least one key semantic relation from the search corpus, identifying at least one keyword and at least one key semantic relation in a search query, and performing a search using the index of features and the identified at least one keyword from the search query and the identified at least one key semantic relation from the search query.

BRIEF DESCRIPTION OF THE DRAWINGS

[32] The above and other aspects of the present invention will become more apparent by describing in detail exemplary embodiments thereof with reference to the attached drawings in which:

[33] **FIG. 1** shows a parsing tree for the query, “pupils punish teachers,” using event-based semantic parsing in accordance with the Theta-Grid system according to an exemplary embodiment of the present invention;

[34] **FIG. 2** shows a parsing tree having associated KSRs according to an exemplary embodiment of the present invention;

[35] **FIG. 3** shows a parsing tree for another query, “optimization of intermediate and machine code” according to an exemplary embodiment of the present invention;

[36] **FIG. 4** shows a parsing tree for yet another query, “Services of optimization for intermediate and machine code” according to an exemplary embodiment of the present invention;

[37] **FIG. 5** shows a table that includes all possible two-keyword KSRs and their classifications and weightings for the query, “Services of optimization for intermediate and machine code” according to an exemplary embodiment of the present invention;

[38] **FIG. 6A** shows the operations performed by a system according to an exemplary embodiment of the invention;

[39] **FIG. 6B** shows the operations performed by a search algorithm according to an exemplary embodiment of the invention;

[40] **FIG. 7A** is a flow chart depicting the search algorithm's flow according to an exemplary embodiment of the invention employing a post-processor;

[41] **FIG. 7B** is a schematic diagram depicting an exemplary embodiment of the invention employing a post-processor;

[42] **FIG. 7C** is a schematic diagram depicting an exemplary embodiment of the post-processor;

[43] **FIG. 8** shows an exemplary user interface of a sentence based semantic services (hereinafter "S3") application; and

[44] **FIG. 9** shows a web page displayed in a conventional keyword search engine.

DETAILED DESCRIPTION OF THE EXEMPLARY EMBODIMENTS

[45] Text searches are accomplished by extracting significant features from a query that represent the meaning behind the query and matching the extracted features against features found in the corpus. The most common approach for keyword-based search suffers from the major drawback of numerous irrelevant hits, as discussed above. An exemplary embodiment of the invention improves the quality of FTS. Exemplary embodiments of the invention can be used for search queries and corpuses in any number of languages. But for purposes of simplicity, and without loss of generality, exemplary embodiments of the invention as they relate to the English language will be more particularly described below.

[46] According to an exemplary embodiment of the invention, a semantic parser extracts event-based semantic relations from sentences. These relations, known as KSRs, serve as powerful textual features for search and other text-processing tasks. The simplest relations are binary (dyadic), involving two components of a sentence. Components are entities such as: action, agent, recipient, and instrument. Roles include: spatial, temporal, process, transfer, and ambient. More complex relations, such as agent-action-recipient, can be extracted as well, or they can be broken up into pairs such as agent-action and recipient-action.

[47] According to an exemplary embodiment of the invention, KSRs can be used to improve search, either by incorporating them as search features of a standalone system or by using them in a post-processor for re-ranking the results of a standard FTS engine. In the direct approach, semantic relations are used just like other textual features in feature vectors, which are the standard tool in document retrieval.

[48] In the simplest formulation, queries and corpus documents are represented as vectors containing the number of occurrences of each word in some corpus-dependent lexicon. Generally, rather than words themselves, occurrences of their base forms are counted. In an exemplary embodiment of the invention, the occurrence of query KSRs is added to the feature vectors used to retrieve relevant documents.

[49] According to an exemplary embodiment of the invention, documents can then be ranked based on weighted occurrences of KSRs. If no KSRs are found in the query, documents can then be ranked based on weighted occurrences of keywords.

[50] According to an exemplary embodiment of the invention, one can infer from a query those KSRs that should not occur in a document/sentence alone, without other KSRs, since

they do not reflect the intent of the query. These KSRs are termed anti-KSRs. Anti-KSRs may also point to less relevancy of a document/sentence even if KSRs with positive weights were also found in the document/sentence. Adding semantics in this way adds very significant components to the feature vectors.

[51] According to an exemplary embodiment of the invention, as with keywords, KSRs can also be generalized with ontological relatives. Actions appearing in queries can be generalized to include synonyms and troponyms. In the same way, objects can match any of their hyponyms.

[52] According to an exemplary embodiment of the invention, feature vectors, enhanced with KSRs, can be used for classification (categorization) via various classification methods, including: *k*-nearest-neighbors, support vector machines (SVM), or naïve Bayesian classifiers. Many other text processing tasks can similarly benefit from extracted KSRs.

[53] According to another exemplary embodiment of the invention, the search method described can be approximated. A potential problem with the direct approach is that it may be prohibitive to index all possible KSRs in the corpus in advance, though it may be possible to semantically parse the sentences in the corpus. Thus, an alternative approach is to extract only those KSRs that appear in the query and only from potentially relevant documents retrieved by a standard search engine.

[54] An exemplary embodiment entails a necessary, but insufficient, condition for occurrence of a (positive) KSR, or syntactic relation, in a document, namely that the keywords (or expanded concepts) of a KSR appear in the same sentence or phrase. Relying on this

heuristic allows one to avoid parsing the document altogether or restrict parsing to a limited number of sentences in which the words of a positive or negative relation occur.

[55] An exemplary embodiment of the invention makes use of semantic parsing available for formal sentence-based analysis. Through semantic parsing of a query string and corpus sentences, KSRs are extracted. Query KSRs are matched with corpus KSRs to increase the relevancy of search results. The matching of KSRs can be performed by the search engine directly or in a post-processing operation performed using the results returned by the search engine in order to enhance the results. Semantic parsing of a sentence entails the parsing of a sentence through an understanding of concepts and meanings encapsulated in the sentence. There are various approaches to semantic parsing. Presently, the most successful semantic theory is based on events.

[56] The simplest relations are binary (dyadic), involving two components of a sentence. Components are entities such as Action, Agent, Recipient, and Instrument. Roles include: Spatial, Temporal, Process, Transfer, and Ambient. More complex relations, such as Agent-Action-Recipient, are broken up into pairs (Agent-Action and Recipient-Action, in this case).

[57] Binary relations are referred to as *semantic twigs* and to more complex relations as *semantic branches*. Semantic twigs or branches of a (multiword) query can serve as *key semantic relations*, or *KSRs*, for search purposes. Semantic parsing of corpus sentences also identifies such relations in documents. Matching query KSRs with corpus KSRs can significantly improve the relevance of retrieved documents.

[58] Consider the first sentence of *Genesis*: “In the beginning God created the heaven and the earth.” It comprises four twig relations:

1. (Agent: God; Action: create). This matches all phrases of the form: “God created [something]” or “[Something] was created by God”.

2. (Recipient: earth; Action: create). This matches all phrases of the form: “[Someone] created the earth”, “The earth was created [by someone]” or “earth creation”, “earth creator”, “Creating the earth”, etc.

3. (Recipient: heaven; Action: create). This matches phrases of the form: “[Someone] created the heavens”, or “The heavens were created [by someone]”.

4. (Temporal: beginning; Action: create). This matches occurrences of the predicate (verb) “create” modified by an adverbial phrase “in the beginning”, such as “In the beginning, [Someone] created [something]”, or “[Something] was created [by someone] in the beginning”, etc.

[59] In the man and dog example, there is a world of difference between a search consisting of the relations (Agent: man; Action: bite) and (Recipient: dog; Action: bite), and the alternative consisting of (Agent: dog; Action: bite) and (Recipient: man; Action: bite).

[60] Of course, binary relations still leave some room for error. A sentence such as, "The man ate lunch while his dog was eaten by a wolf" contains both semantic twigs, along with two partial matches: (Agent: wolf; Action: bite) and (Recipient: lunch; Action: bite). With ternary relations, this would be obviated, because the search would be looking for the semantic branch (Agent: man; Action: bite; Recipient: dog). Still, even with binary relations only, a document may be penalized when a query has mismatches involving the same action in the same sentence as the matching relations.

[61] The following examples demonstrate semantic parsing of queries and how this parsing and the resulting KSRs are used in the search process. In the following, all exemplary parsings are in accordance with the Theta-Grid System or extensions thereof. The Theta System is the central system of the systems of concepts. The Theta System provides an interface between systems of concepts and the computational system (syntax), and indirectly (via the syntactic representations), with the semantic inference systems. The Theta System may include: (1) coded concepts with formal features defining the θ-relations of verb-entries; (2) a set of arity operations on lexical entries, which may generate new entries or just new options of realization; and (3) marking procedures, which "prepare" a verb entry for syntactic derivations (e.g., assign an accusative feature to the verb in the relevant cases and determine merging properties of arguments, technically obtained by indices). The Theta System is further described by Tanya Reinhart in "The Theta System: an Overview," available at http://www.tau.ac.il/~reinhart/ling_dl/overview.doc, last visited October 26, 2007. However, the scope of this invention is not limited to utilization of the Theta-Grid System, and any other valid

event-based semantic framework can be utilized as well. The Theta-Grid System is provided by way of an example only.

[62] **FIG. 1** shows an event-based semantic parsing of the query, "pupils punish teachers," which is a VP. In this query, there is one verb, "punish" 100, which takes up to two arguments. The first argument is denoted as R1 and the second one is denoted as R2. In this example, R1 is "pupils" 110, while R2 is "teachers" 120.

[63] The query contains two relevant semantic relations, denoted as KSRs. In this example, both KSRs are two-keyword KSRs. The first KSR consists of the keyword "punish," which is a predicate, and the keyword "pupil," which is a first argument (R1, agent). The second KSR consists of the keyword "punish," which is a predicate, and the keyword "teacher," which is a second argument (R2, recipient).

[64] More particularly, a KSR is any subset of semantic tree nodes, all having a common ancestor. KSRs can have two or more tree nodes.

[65] In a semantic parsing tree of a sentence, the nodes contain each one word, and the nodes are connected by their corresponding semantic relation.

[66] KSRs represent various types of semantic relations, including nominalizations (e.g., code optimization); special morphological relations (e.g., parallel ◊ parallelism); expression and non-expression relations (e.g., operating system ◊ local system); predicate-argument relations in VPs; synonyms (e.g., running ◊ managing/jogging); hypernyms (e.g., programming language ◊ Fortran); spatial, temporal, causal, instrumental and intentional attributes; and parts of speech.

[67] **FIG. 2** shows a parsing tree having associated KSRs. In **FIG. 2**, the predicate 200 is at the root of the parsing tree. The predicate 200 has two arguments, R1 210 and R2 220. R1 in turn has two arguments, R1,1 230 and R1,2 240. R2 has one argument, R2,1 250. In the parsing tree of **FIG. 2**, the possible two-word KSRs are: predicate 200, R1 210; predicate 200, R2 220; R1 210, R2 220; predicate 200, R1,1 230; predicate 230, R1,2 240; predicate 200, R2,1 250; R1 210, R1,1 230; R1 210, R1,2 240; R1,1 230, R1,2 240; R2 220, R2,1 250; R1 210, R2,1 250; R1,1 230, R2,1 250; R1,2 240, R2,1 250; R1,1 230, R2 220; and R1,2 240, R2 220.

[68] Returning to the example query, "pupils punish teachers," an exemplary embodiment of the invention might return the following search results: "**74 would punish teachers** by allowing administrators to fire them for any reason...," "**Vallas punished teachers** and principals who chose to work with the neediest and lowest scoring children...," "**teachers were punished** by their aggravated **pupils** by...," and "tests **basically punish teachers**, students, administrators, and parents based upon outcomes over which they have little or no control..."

[69] **FIG. 3** shows a parsing tree for a second query phrase, "optimization of intermediate and machine code," which is an NP. The keyword "optimization" 300 is the predicate, the keyword "code" is the second argument R2 310 (the first argument R1 is the implied "agent"), the keyword "intermediate" is an attribute 320, and the keyword "machine" is an attribute 330.

[70] In this example, the most relevant semantic relation exists within the keyword-pair "optimization code." This phrase exhibits a significant semantic attribute-head relationship between keywords. Note that in this case, the FTS distance within the query is maximal. This

example demonstrates that often pairs or larger sets of meaningful words may be located in different parts of the sentence. Thus, caution must be exercised as the proximity assumption often used in existing keyword based search engines is not always correct.

[71] An irrelevant pair of keywords in this example is “intermediate machine.” There is neither any attribute-head nor any other KSRs in the query between these keywords. Note that in this case, the FTS distance within the query is minimal; nonetheless, the pair is irrelevant.

[72] The keyword subsets identified in the query can be further classified as super-KSRs, which are subsets that exhibit the most meaningful semantic relations; intermediate-KSRs, which are subsets that exhibit semantic relations that are meaningful but less so than the super-KSRs; and anti-KSRs, which are subsets that have no meaningful semantic relations, and whose presence in the text may point to a meaning significantly different from the meaning expressed in the query.

[73] In general, KSRs exhibit the most meaningful semantic relations (i.e. semantically more “important/relevant”), are those KSRs which: (a) are closer to the root of the semantic parsing tree (i.e. the keywords constituting the KSR have a smaller, possibly weighted, distance to the root); and (b) have a smaller (possibly weighted) distance (on the semantic parsing tree) between the keywords constituting the KSR. Typically, a KSR's “importance” can be determined by some (optimized) function of (a) and (b).

[74] As an example, siblings in a parsing tree, such as “intermediate” 320 and “machine” 330 in FIG.4 may be considered anti-KSRs. In some embodiments, words and indirect attributes may also be considered anti-KSRs.

[75] Thus, in an exemplary embodiment, more importance is placed on the relevant KSRs, while anti-KSRs are treated as evidence against relevancy. This difference in importance can be handled by introducing a weighting scheme. In an exemplary embodiment of the invention, there is a weighting scheme from -2 to 5, whereby relevant KSRs receive a positive weight between 1 and 5, while anti-KSRs receive a negative weight between -1 and -2. Different weight ranges are also possible. The weights may be assigned manually or may be optimized based on the corpus to search and the type of queries involved.

[76] It is emphasized that anti-KSRs are also KSRs. A given KSR may become an anti-KSR in certain situations, while in other situations it may become a KSR with a positive weight. Consider, as an example, siblings in a parsing tree. In FIG. 4, the siblings “intermediate” 320 and “machine” 330 may be considered anti-KSRs. However, the siblings “optimization” 410 and “code” 420 form a KSR with a positive weight.

[77] There might be various methods for determining which KSR is actually an anti-KSR. One can have certain rules for labeling a KSR as an anti-KSR. For example, direct siblings connected with a coordination relation will always be anti-KSRs. Any method may be used for determining whether a KSR is an anti-KSR and in which situations.

[78] **FIG. 4** shows a parsing tree for the query, “Services of optimization for intermediate and machine code.” **FIG. 5** shows a table with all possible two-keyword KSRs in the query, together with their classification and weighting in an exemplary embodiment of the invention. Column 501 lists the first keyword in each two-keyword KSR, column 502 lists the second keyword in each two-keyword KSR, column 503 lists the score for the KSR, and column 504 lists the classification for the KSR. As shown in **FIGS. 4 and 5**, there are five keywords

400, 410, 420, 430, and 440, yielding ten distinctive pairs of two-keyword KSRs, shown in rows 500, 510, 520, 530, 540, 550, 560, 570, 580, and 590, in **FIG. 5**. (From five different items, the number of possible different pairs is ten.) Out of the ten pairs, there are five KSRs, 500, 510, 520, 530 and 540, and 5 anti-KSRs, 550, 560, 570, 580, and 590. There is only one super-KSR ("optimization services" 500), which receives a score (weight) of 5 (out of 5). There are four additional intermediate-KSRs, 510, 520, 530 and 540, each receiving a score of 3 or 4. There are also five anti-KSRs, 550, 560, 570, 580, and 590, each receiving a score of -2. It is worth mentioning that different super-KSRs may receive different weights within the same sentence, and the same applies for intermediate-KSRs and anti-KSRs.

[79] KSRs extracted from a query are then utilized in the FTS process. In general, FTS is carried out by matching a set of features extracted from the query to features found in the corpus. All other variables being constant, documents having a relatively high match of feature sets (in other words, having a relatively high number of features in the query that are identical or similar to features found in the documents) will be ranked higher and appear as more relevant than documents which have a relatively low match of feature sets.

[80] In the standard keyword search engines, keywords are the main features extracted from the query and documents. Other features can include phrases, synonyms, and the like. In accordance with an exemplary embodiment of the invention, KSRs are used as additional features to be incorporated as search features. Typically, feature vectors in which feature information is encoded are built for both the query and the corpus. These feature vectors are used in the matching process. The search engine attempts to find a full or partial match between the query feature vector and the feature vectors of the various documents in the corpus.

[81] Thus, both keywords and KSRs may be used as search features. Keywords are used only in cases where no KSRs can be found in the query. KSRs may be used either by incorporation as search features in a standalone system or by utilization in a post-processor for re-ranking the results of a standard FTS engine.

[82] Turning first to the use of KSRs as search features in a standalone system, in the simplest formulation of standard keyword-based FTS, queries and corpus documents are represented as vectors containing the number of occurrences of each word in some corpus-dependent lexicon. Generally, rather than words themselves, occurrences of base forms of words are counted (for example, the base form of both "walking" and "walked" is "walk"). For single-word features, it is standard to take its frequency in the corpus as a whole into account, by using *tf-idf* (term frequency/inverse document frequency) or one of its many variants. Then, a measure, such as the discrete cosine, is applied to find the most relevant documents.

[83] By the same token, multiword sequences (*n*-grams) or meaningful phrases can be used as features. Very frequent words ("stop-words") and very infrequent words are usually omitted from consideration, so as not to introduce "noise". Besides word occurrences, features can include synonyms ("heaven," "Eden," "paradise," "nirvana," "promised land," and "Shangri-la" are synonyms in WordNet), or related words derived from ontologies.

[84] According to an exemplary embodiment of the invention, the number of occurrences of query KSRs is added to the feature vectors used to retrieve relevant documents. Adding semantics in this way will add very significant components to the feature vectors.

[85] Thus, both keywords and KSRs are present in query and corpus feature vectors. In the search process, documents are then ranked based on weighted occurrences of keywords

and/or KSRs. If KSRs were found within the query, only KSRs are used in the search process, otherwise only keywords are used in the search process, similar to other keyword-based search engines. There are many possible weighting schemes that may be utilized. Various weighting schemes may be more adequate in some situations than others. The weighting schemes can be optimized to yield favorable search results for given corpuses and query types.

[86] In addition to KSRs, anti-KSRs may be utilized to further enhance the search process. Thus, in the search weighting scheme, anti-KSRs may induce negative weights so as to give less relevance to the documents containing them. Utilizing KSRs in this way adds very significant components to the feature vectors. Anti-KSRs are mainly relevant in sentences not including any KSRs with positive weights.

[87] Similar to keywords, KSRs may also be generalized with ontological relatives. Actions appearing in queries may be generalized to include synonyms and troponyms (e.g., march is a troponym of walk). In the same way, objects may match any of their hyponyms, which are words or phrases whose semantic range is included within that of another word (e.g., scarlet, vermilion, carmine, and crimson are all hyponyms of red, their hypernym).

[88] Feature vectors, enhanced with KSRs, may be used for classification (categorization) via various classification methods, including: *k*-nearest-neighbors, SVM, or naïve Bayesian classifiers. Many other text processing tasks can similarly benefit from extracted KSRs.

[89] Prior to the search, the corpus undergoes an indexing process in which keywords and KSRs are extracted and encoded in the index. The index maps the space of keywords and KSRs to the documents containing them. Thus, once an index is built, it is easier and faster to

locate all documents containing all or a subset of a given feature vector. There are many possible ways to construct indexing schemes. Various indexing schemes may be more adequate in some situations rather than others.

[90] In an exemplary embodiment of the invention, the indexing scheme employs a corpus-dependent lexicon that stores base forms of words rather than words themselves. Various indexing schemes may be employed for KSRs. In the following, an example indexing scheme for KSRs is presented.

[91] Each word in the lexicon points to an entry in another table. This entry in turn contains the list of all sentences (represented as some ordinal within the corpus) in the corpus which contain this word. Another structure holds the semantic tree structure of each sentence. Now, assume a two-word KSR contains Word-A and Word-B. The search for this KSR in the corpus proceeds as follows. First, all sentences containing both Word-A and Word-B are found (as the intersection of all sentences that contain Word-A and all sentences that contain Word-B). Now, for each sentence that contains both Word-A and Word-B, the parsing tree of the sentence is consulted to see whether Word-A and Word-B have the same relation within the semantic tree as within the KSR searched for.

[92] Using the indexing scheme for keywords and KSRs, queries and corpus documents may be represented as vectors containing the number of occurrences of each word and KSR. Then, the index may be utilized for a faster retrieval of the relevant documents.

[93] **FIG. 6A** shows the operations performed by the system according to an exemplary embodiment of the invention:

[94] (1) In operation 600, prior to search, the system processes the corpus, extracts all keywords and KSRs from the corpus' documents, and creates an index of the features;

[95] (2) In operation 610, the system parses the query text and extracts keywords, KSRs, and anti-KSRs from the query;

[96] (3) In operation 620, the system utilizes the index to locate those documents in the corpus that contain the keywords and KSRs; and

[97] (4) In operation 630, the system ranks the documents in order of relevancy. The ranking algorithm takes into account the KSRs and anti-KSRs found in each file, together with their corresponding weights. In general, the more relevant KSRs a file contains, the higher its rank shall be. (Ranking algorithms are described below with more particularity.) Thus, the semantically relevant results (files) are pushed higher in the keyword search ranking ladder, and the semantically irrelevant results (files) are pushed lower in the keyword search ranking ladder.

[98] Turning now to the utilization in a post-processor for re-ranking the results of a standard FTS engine, in this scheme, the system according to an exemplary embodiment of the invention works as a post-processor to existing keyword FTS engines, in order to enhance the search results. Several advantages may be realized by this approach. Under this approach, it is not necessary to store and maintain the corpus. When dealing with large corpuses, very large amounts of storage are necessary, which translate into high costs and significant ongoing maintenance. Additionally, no a priori building of an index is necessary. When dealing with large corpuses, creating an index may be very time and space consuming and in turn adds to the cost.

[99] **FIG. 6B** shows the operations performed by the search algorithm according to an exemplary embodiment of the invention:

[100] (1) In operation 640, keywords are extracted from the query by the keyword search engine;

[101] (2) In operation 650, the keyword search engine recalls/retrieves all files that include the query's keywords and ranks the files by a certain FTS logic/algorithm. This may be any keyword search engine with reasonable performance;

[102] (3) In operation 660, the system parses the query text and identifies KSRs;

[103] (4) In operation 670, the system parses the keyword search output (either all of the output or a subset thereof) and searches for matching KSRs; and

[104] (5) In operation 680, the system re-ranks the keyword search results. The re-ranking algorithm takes into account the KSRs and anti-KSRs found in each file, together with their corresponding weights. In general, the more relevant KSRs a file contains, the higher its rank shall be. (Re-ranking algorithms are described below with more particularity.) Thus, the semantically relevant results (files) are pushed higher in the keyword search ranking ladder, and the semantically irrelevant results (files) are pushed lower in the keyword search ranking ladder.

[105] **FIG. 7A** is a flow chart depicting the search algorithm's flow according to an exemplary embodiment of the invention employing a post-processor. A query is entered at step 7000. Once a query is entered at operation 7000, the query is enhanced with ontology at operation 7010. Once the query is enhanced with ontology at operation 7010, keywords and logical operators are extracted from the query text by a third party search engine at operation 7020. Next, files including the keywords are retrieved and ranked by the third party search

engine at operation 7030. Then, the query and query result text files are preprocessed to obtain discrete sentences at operation 7040.

[106] Alternatively, after the query is entered at operation 7000, the flow may jump directly to the pre-processing step 7040. Alternatively, the query is entered at operation 7000, the flow may jump directly to operation 7020, extracting keywords and logical operators from the query text. After operation 7010, enhancing the query with ontology, the flow may alternatively jump to the pre-processing operation 7040.

[107] After the pre-processing step 7040, the query and query results (either the entire documents or document snippets) are parsed at operation 7050. Next, KSRs are extracted from the query and query results at operation 7060. After the KSRs are extracted, the query results are re-ranked using KSR-matching in operation 7070. After the re-ranking of the query results, the re-ranking information may optionally be integrated into the third party search engine query results, either by the third party search engine or by the post-processor, at operation 7080.

[108] **FIG. 7B** is a schematic diagram depicting an exemplary embodiment of the invention employing a post-processor. The system server 7140 communicates through the Internet or Intranet (or any other communication method) 7120 with a user's computer/terminal 7100. The system server 7140 also communicates with a third party search engine API 7130, which communicates through the Internet/Intranet 7120 with a third party search engine 7110. The system server 7140 also communicates with an ontology database 7150, a language dictionary 7160, and a score rule-base 7170.

[109] **FIG. 7C** is a schematic diagram depicting an exemplary embodiment of the post-processor 7220. The post processor 7220 communicates through the Internet/Intranet 7210 with a third party search engine 7200 via the third party search engine communication module 7240.

[110] The third party search engine communication module 7240 communicates with the text pre-processor module 7230, which communicates with the English semantic parser module 7250. The English semantic parser module 7250 communicates with the ontology database module 7255, the query enhancer module 7265, and the relation extractor module, 7260. The ontology database module 7255 communicates with the query enhancer module 7265, and the query enhancer module 7265 communicates with the relation extractor module 7260.

[111] The relation extractor module 7260 communicates with the relevance ranker module 7270, which communicates with the score rule-base module 7280 and the post search integrator module 7290. The post search integrator module 7290 communicates with the third party search engine communication module 7240.

[112] In one exemplary embodiment, the keyword FTS engine retrieves all the documents that match the query (search results). Typically this number is very large, so it is not cost-effective to re-rank all the search results. An approximation is thus to process only the top *N* documents (e.g., the top 1000).

[113] A disadvantage of this scheme in the exemplary embodiment is that some accuracy is lost. The top *N* documents which are processed may not be the *N* most relevant documents that would have been retrieved utilizing the KSR matching. Thus, many good KSR matches are potentially lost. However, if *N* is large enough, the results are typically still

satisfactory. In any event, adding the KSR re-ranking phase typically significantly enhances the search results of the keyword FTS engine.

[114] Turning now to the re-ranking algorithm, this algorithm takes as input the top *N* results of the keyword search engine and re-ranks them taking into account KSRs as well as the search engine ranking.

[115] An exemplary embodiment of the invention may, for each result, retrieve the whole document. Then, keywords and KSRs are extracted from the document and are used in the matching process. Another exemplary embodiment of the invention may, for each result, retrieve only the snippet (that is, the text fragments that the search engines retrieves together with the result link, which typically contain keywords found in the query and in the document) of the document. Then, keywords and KSRs are extracted from the snippet and are used in the matching process.

[116] Still another exemplary embodiment of the invention may employ a combination of the above, first using snippets, extracting keywords as well as KSRs from the snippets. An initial re-ranking of the results is performed and the top *K* results (where $K < N$) are selected. Then, for these top *K* results, the entire documents are retrieved and keywords as well as KSRs are extracted from the documents. The extracted keywords and KSRs are used in the final matching process.

[117] Once keywords and KSRs are extracted from the documents or snippets, a matching algorithm determines a re-ranking of the results. The matching may be performed through various methods and by use of various similarity metrics. In general, matching of feature vectors found in the query as well as the results text is performed.

[118] The matching algorithm determines a similarity measure of each result to the query. Typically, the higher the similarity measure, the better the match. However, one can also take into account the original ranking by the keyword-based search engine. In the later ranking algorithm, other considerations affect ranking in addition to the similarity measure (e.g., how recently the document was updated, how popular the document is as employed by Google's® PageRank™ algorithm, payment for advancing the result up the ranking ladder, etc.).

[119] Various approaches may be used to preserve the information used in the original ranking by the keyword-based search engine. These approaches include taking into account the keyword search engine's considerations (smaller alterations), optimization processes, and re-ranking the first top *N*, then continuing in the background to enhance search results by re-ranking the next top *N*s.

[120] In an exemplary embodiment, the following KSRs may be used: noun-phrase predicate-argument relations; nominalizations (e.g., "memory management aspects"); special morphological relations (e.g., "parallel" vs. "parallelism"); and expression/non-expression relations (e.g., "operating system" is a collocation; "local system" is not). This basic set of KSRs, used in a postprocessor according to an exemplary embodiment of the invention to re-rank the results of the Lucene search engine, when run on the CACM collection, gave a mean average (non-interpolated) precision of 0.35, representing an improvement of 15.5% over and above the current record for this benchmark, and an 11-point interpolated precision of 0.37, representing an improvement of 14%.

[121] Experiments were also performed with the National Institute of Standards and Technology *Text Retrieval Conference (TREC) 8* benchmark. Search precision on the top ten

results were compared with and without key binary semantic relations. Here, too, using post-processing to reorder the documents retrieved by the leading search engine in this category increased precision (on preliminary tests of 50 queries) by 5%.

[122] Exemplary embodiments of the invention can be used with many different FTS engines, including the following prominent FTS segments:

[123] Open Internet search, where queries are submitted to be searched across the Internet domain. Examples for systems in this segment are the corresponding systems from Google®, Yahoo®, and MSN®;

[124] Corporate search, where queries are submitted to be searched within the enterprises databases and files. Examples of systems in this segment are the corresponding systems from Autonomy-Verity, Oracle, and Google®;

[125] Closed database content providers that have databases containing useful information for sale and allow clients to search for the desired information in these databases. Examples for systems in this segment are the corresponding systems from Westlaw (Thomson), LexisNexis (Reed-Elsevier) and Wolters Kluwer;

[126] Web site search, where a given web site is searched. This can also be extended to e-commerce web sites, where a search can be for full text or items sold on the given web site. Examples for systems in this segment are the corresponding systems from Google® and Celebros; and

[127] Desktop search, where the end-user's computer system is searched, including files, emails, and other items. Examples for systems in this segment are the corresponding systems from Google® and Microsoft.

[128] Exemplary embodiments of the invention may also be used in various other fields, including:

[129] Contextual advertisement, where in an exemplary embodiment, advertisements are matched to submitted queries, so as to increase the likelihood of advertisement to be relevant to the end-user. Examples for systems in this domain are the corresponding systems from Google®, Yahoo®, MSN® and Quigo. In an exemplary embodiment, text on a web page visited by a user is treated as query text, and a collection of advertisements that may be displayed is treated as the search corpus. A search is then performed, similar to previously described embodiments, to locate the ads that best match the text or web page context. Another usage is for matching ads for search engines. When a query is submitted to a search engine, the search query is treated as query text, and a collection of advertisements that may be displayed is treated as the search corpus. A search is then performed, similar to previously described embodiments, to locate the ads that best match the text or web page context;

[130] Operating system (OS) help and troubleshooting, where in an exemplary embodiment, users submit questions and other queries to be answered by the OS help/troubleshooting system. The OS help system contains documents with relevant information for the help function. In an exemplary embodiment, the user help query is treated as query text, and a collection of OS help system documents is treated as the search corpus. A search is then performed, similar to previously described embodiments, to locate the help contents that best match the user query;

[131] Helpdesks, where in an exemplary embodiment, a request submitted by a user (e.g., for support, problem troubleshooting, etc.) is treated as query text, and a collection of helpdesk documents and/or service options is treated as the search corpus;

[132] Blocking of irrelevant advertisement (e.g., ad anti-virus), where in an exemplary embodiment, for a given text content (e.g., web page, web site, search query text, enterprise query, etc.) the system identifies the advertisements that are not relevant to the given content in order to block them. In an exemplary embodiment, the given text content is treated as query text, and a collection of advertisements that may be displayed is treated as the search corpus. A search is then performed, similar to previously described embodiments, to locate the ads that best match the text or web page context. Next, all other ads in the ads collection (namely, the ads that do not match) are blocked. Another usage is for blocking of ads coming from a list of web sites. In an exemplary embodiment, the given text content is treated as query text, and a collection of web sites is treated as the search corpus. A search is then performed, similar to previously described embodiments, to locate the web sites that best match the text context. Next, all other web sites in the list in the web site collection (namely, the web sites that do not match) are blocked;

[133] Full-text matching, where in an exemplary embodiment, matching documents from a collection of documents to a given document are found (this is document to document matching);

[134] Speech recognition, where in an exemplary embodiment, speech is translated into text, and the translated text is then treated as query text for search against a corpus. In another exemplary embodiment, KSRs may be used to aid a speech recognition

system in determining unintelligible or unrecognizable words. In this embodiment, sets of KSRs are determined for each of multiple versions of the sentence, each version using a different possibility for an unintelligible or unrecognizable word. Then, the KSR collections are compared with KSR collections for sentences in a very large search corpus comprised of a broad variety of documents, in order to determine which KSR collection is most probable based on the frequencies in the corpus. The possibility for the unintelligible or unrecognizable word used in the sentence with the most probable KSR collection is then assumed to be the correct word;

[135] Document summarization, where in an exemplary embodiment, documents are summarized by their KSRs. There may be a hierarchy of summarizations. In the first level, the document is summarized by the KSRs with the highest importance/weight (e.g., Super KSRs). In the lower levels, KSRs of less importance/weight are added to the summarization; and

[136] "What did you mean?" services, where in an exemplary embodiment, the user who submits a query is suggested a list of other similar relevant queries to choose from (e.g., to further enhance the search, to broaden the scope of search, etc.). In an exemplary embodiment, the query submitted by the user is parsed to extract KSRs. Then, for some or all KSRs, other KSRs are derived (e.g., by ontology, by synonyms, etc.). Then, the newly derived KSRs are placed back into the query instead of the original KSR to yield a derived query. Now, the list of thus newly created derived queries is presented to the user to choose from, and may be potentially (weighted) ordered by the importance/weight of the KSRs they contain. As an example, see the example for the query "Steve Irwin killed by a stingray." in **FIG. 8** relating to

"sentence based semantic services." In this example, the user may be presented with other derived queries (e.g., "People killed by a stingray," "People killed by fish," "Steve Irwin," etc.).

[137] An exemplary embodiment of the invention is a sentence based semantic services (hereinafter "S3") application. In this application, for example, imagine while reading an online newspaper a reader encounters a sentence that catches the reader's eye: "Steve Irwin killed by a stingray." Immediately, the reader wishes to read more about cases where people were killed by stingrays.

[138] Clicking the right mouse button while the cursor is located somewhere on the above sentence, the reader is shown a pop-up window or a pop-up overlay. The user is presented with a display similar to that shown in **FIG. 8**. This display provides semantically relevant information. In contrast, a conventional keyword search engine might provide the user with a less semantically relevant display, similar to that shown in **FIG. 9**.

[139] Such an application may be useful for closed database content provides, with priority given to those that also generate revenues from advertising, and preferably focused advertisements. Examples of such content providers include LexisNexis and the Wall Street Journal.

[140] An application similar to this exemplary embodiment of the invention benefits the content provider by increasing archive revenues. That is, end users will purchase more paid articles because more end users will access the archive search because of the attractive and convenient graphical user interface (hereinafter "GUI") as well as the free service, and more users will find articles of interest because those articles are ranked higher in the abstract list.

Often, this better relevance ranking will make the difference between a user finding and purchasing the article and quitting without having found and purchased a desired article.

[141] The content provider is also benefited by increasing advertising revenues. That is, the advertising price and space may be increased because KSR matching between the users' query and the advertisers' slogans creates the ultimate focused/contextual advertising campaign where the advertiser's ad is matched with all relevant user queries and only relevant queries, and because the S3 application pop-up GUI itself provides additional advertisement space.

[142] An application similar to this exemplary embodiment of the invention also benefits the end user through enhanced search results for archive, in-site, and open Internet search; free translation services; and a friendly GUI (e.g., there is no need to type or cut and paste query text, immediate access to a search box is provided anywhere in the content provider's site, etc.).

[143] The invention is not limited to the exemplary embodiments described above. That is, the above and other features of the invention including various novel method operations and a system of the various modules have been particularly described with reference to the accompanying drawings and pointed out in the claims. It will be understood that the particular process and construction of parts embodying the invention is shown by way of illustration only and not as a limitation of the invention. The principles and features of this invention may be employed in varied and numerous embodiments without departing from the spirit and scope of the invention as defined by the appended claims.

WHAT IS CLAIMED IS:

1. A method for using key semantic relations in full text search, the method comprising:

performing a keyword search using a search query;

identifying at least one key semantic relation in the search query;

identifying at least one key semantic relation in at least one document or document snippet in output of the keyword search that matches a key semantic relation identified in the search query; and

re-ranking the keyword search output by using the at least one identified matching key semantic relation in the at least one document or document snippet.

2. The method of claim 1, wherein the re-ranking of the keyword search output by using the at least one identified matching key semantic relation in the at least one document or document snippet comprises determining a weight for each of the at least one document or document snippet.

3. The method of claim 2, wherein the re-ranking of the keyword search output by using the at least one identified matching key semantic relation in the at least one document or document snippet further comprises re-ranking the keyword search output in order of weight.

4. The method of claim 3, wherein in the re-ranking of the keyword search output in order of weight, a document or document snippet with largest weight is ranked first.

5. The method of claim 4, wherein the determining of the weight for each of the at least one document or document snippets comprises increasing by some function a document or document snippet's weight for each identified matching key semantic relation in the document or document snippet.

6. The method of claim 5, wherein in the increasing by some function the document or document snippet's weight, a larger increase is made for each identified matching super key semantic relation and a smaller increase is made for each identified matching intermediate key semantic relation.

7. The method of claim 5, wherein the determining of the weight for each of the at least one document or document snippets further comprises decreasing by some function the document or document snippet's weight for each identified matching anti key semantic relation.

8. The method of claim 1, wherein the key semantic relations are generalized with ontological relatives.

9. A method for using key semantic relations in full text search, the method comprising:

identifying at least one keyword and at least one key semantic relation in a search corpus;

creating an index of features using the identified at least one keyword from the search corpus and the identified at least one key semantic relation from the search corpus;

identifying at least one keyword and at least one key semantic relation in a search query; and

performing a search using the index of features and the identified at least one keyword from the search query and the identified at least one key semantic relation from the search query.

10. A computer program product for enabling a computer to use key semantic relations in full text search, the computer program product comprising:

software instructions for enabling the computer to perform predetermined operations; and

a computer readable medium bearing the software instructions;

the predetermined operations including:

performing a keyword search using a search query;

identifying at least one key semantic relation in the search query;

identifying at least one key semantic relation in at least one document or document snippet in output of the keyword search that matches a key semantic relation identified in the search query; and re-ranking the keyword search output by using the at least one identified matching key semantic relation in the at least one document or document snippet

11. A computer program product for enabling a computer to use key semantic relations in full text search, the computer program product comprising:

software instructions for enabling the computer to perform predetermined operations; and

a computer readable medium bearing the software instructions;

the predetermined operations including:

identifying at least one keyword and at least one key semantic relation in a search corpus;

creating an index of features using the identified at least one keyword from the search corpus and the identified at least one key semantic relation from the search corpus;

identifying at least one keyword and at least one key semantic relation in a search query; and

performing a search using the index of features and the identified at least one keyword from the search query and the identified at least one key semantic relation from the search query.

12. A computer system adapted to use key semantic relations in full text search, the computer system comprising:

a processor; and

a memory including software instructions that cause the computer system to perform:

a keyword search using a search query;

an identification of at least one key semantic relation in the search query;

an identification of at least one key semantic relation in at least one document or document snippet in output of the keyword search that matches a key semantic relation identified in the search query; and

a re-ranking of the keyword search output by using the at least one identified matching key semantic relation in the at least one document or document snippet.

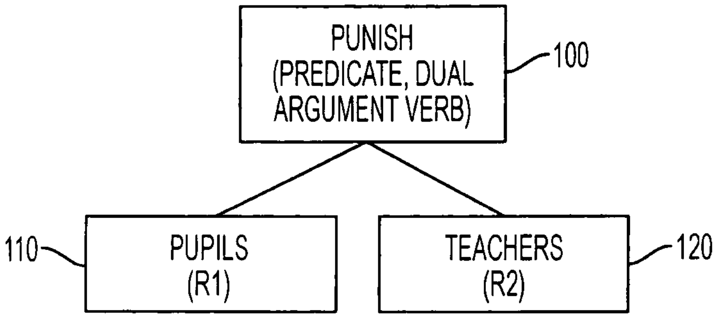

PUNISH
(PREDICATE, DUAL
ARGUMENT VERB)
100
110
PUPILS
(R1)
TEACHERS
(R2)
120

FIG. 1

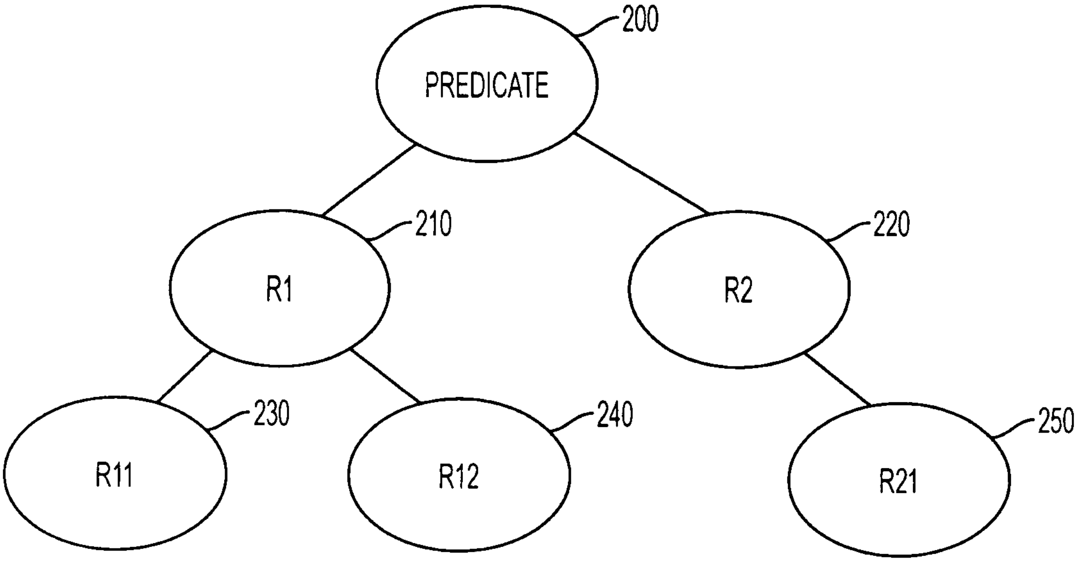


FIG. 2

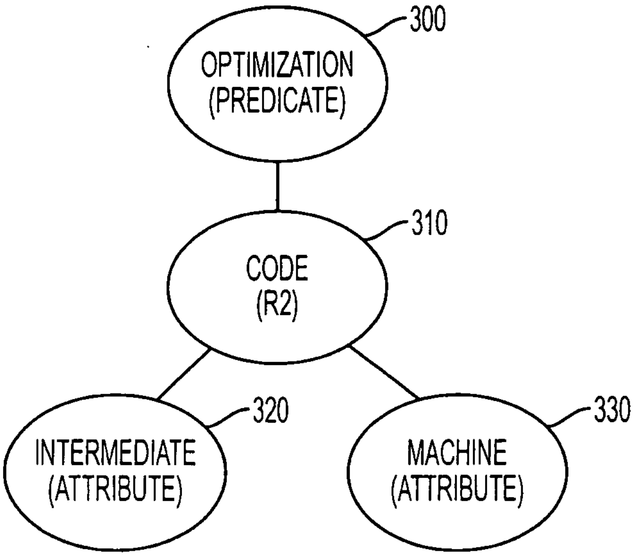

300
OPTIMIZATION
(PREDICATE)
310
CODE
(R2)
320
INTERMEDIATE
(ATTRIBUTE)
330
MACHINE
(ATTRIBUTE)

FIG. 3

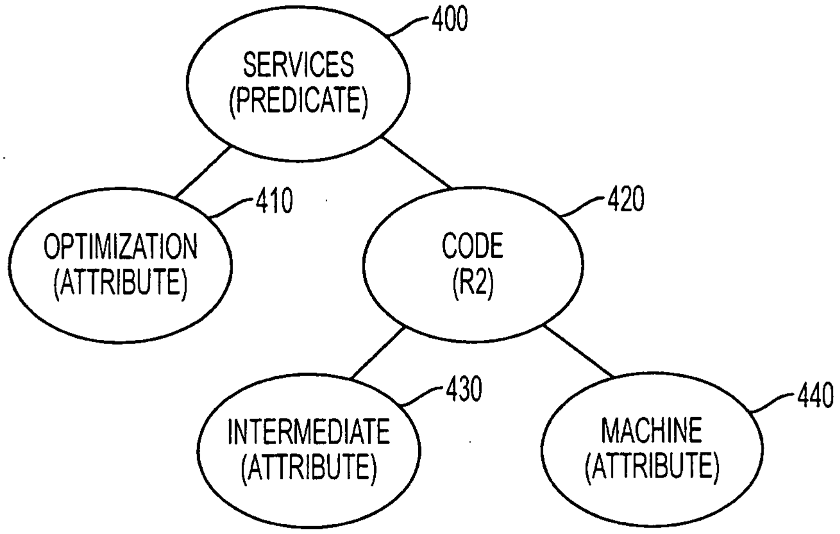
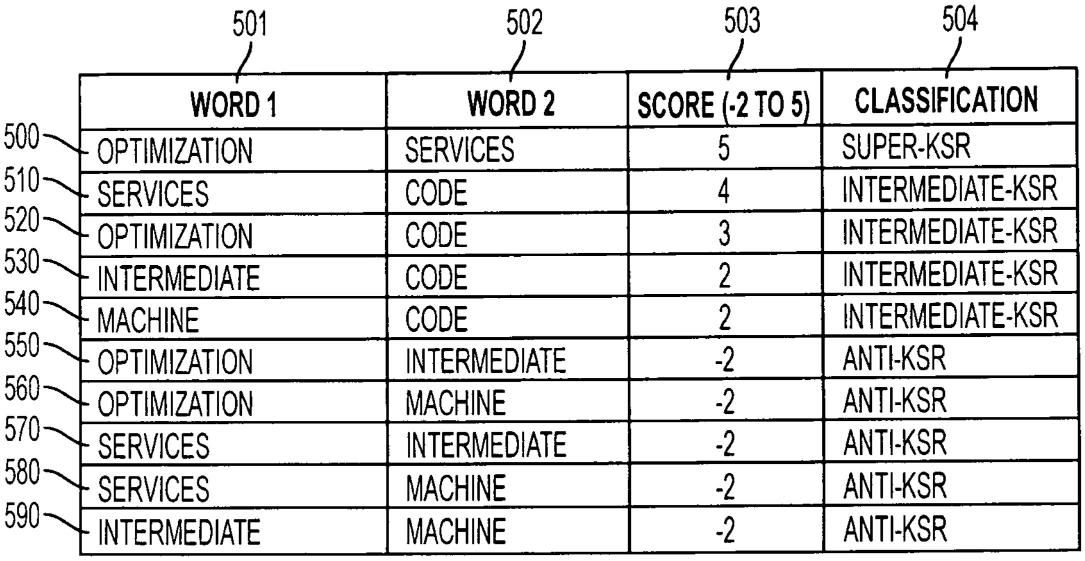

400
SERVICES (PREDICATE)
410
OPTIMIZATION (ATTRIBUTE)
420
CODE (R2)
430
INTERMEDIATE (ATTRIBUTE)
440
MACHINE (ATTRIBUTE)

FIG. 4


501
502
503
504
500
510
520
530
540
550
560
570
580
590

WORD 1	WORD 2	SCORE (-2 TO 5)	CLASSIFICATION
OPTIMIZATION	SERVICES	5	SUPER-KSR
SERVICES	CODE	4	INTERMEDIATE-KSR
OPTIMIZATION	CODE	3	INTERMEDIATE-KSR
INTERMEDIATE	CODE	2	INTERMEDIATE-KSR
MACHINE	CODE	2	INTERMEDIATE-KSR
OPTIMIZATION	INTERMEDIATE	-2	ANTI-KSR
OPTIMIZATION	MACHINE	-2	ANTI-KSR
SERVICES	INTERMEDIATE	-2	ANTI-KSR
SERVICES	MACHINE	-2	ANTI-KSR
INTERMEDIATE	MACHINE	-2	ANTI-KSR

FIG. 5

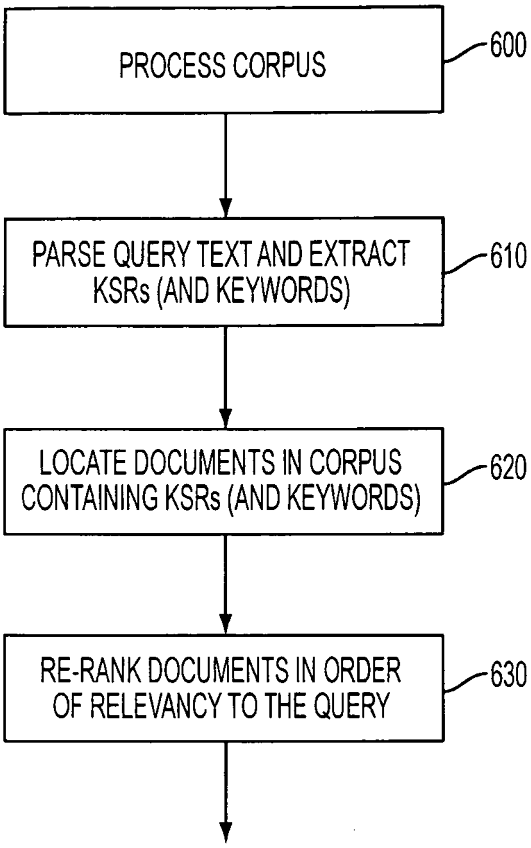

PROCESS CORPUS
600
PARSE QUERY TEXT AND EXTRACT KSRs (AND KEYWORDS)
610
LOCATE DOCUMENTS IN CORPUS CONTAINING KSRs (AND KEYWORDS)
620
RE-RANK DOCUMENTS IN ORDER OF RELEVANCY TO THE QUERY
630

FIG. 6A

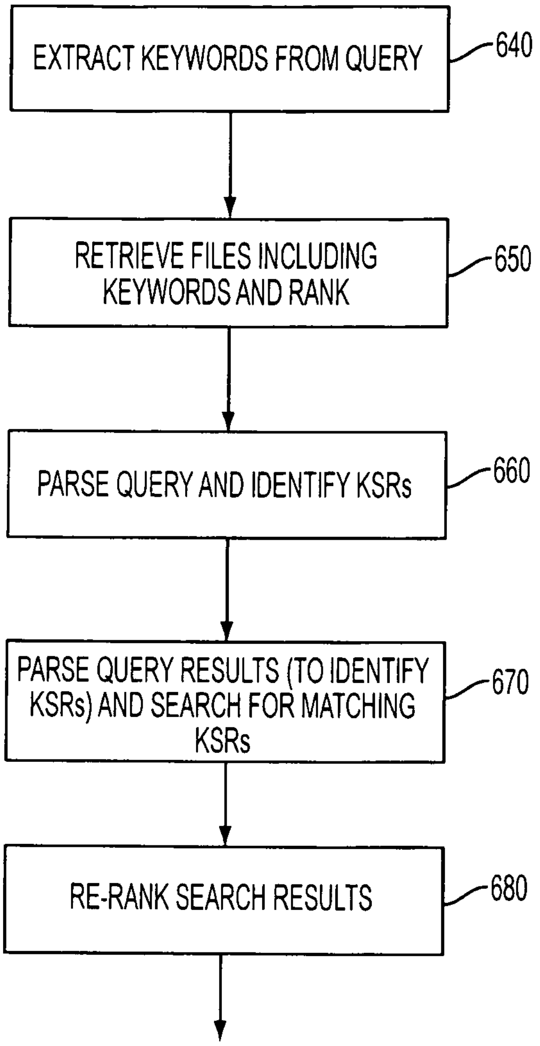

EXTRACT KEYWORDS FROM QUERY
640
RETRIEVE FILES INCLUDING KEYWORDS AND RANK
650
PARSE QUERY AND IDENTIFY KSRs
660
PARSE QUERY RESULTS (TO IDENTIFY KSRs) AND SEARCH FOR MATCHING KSRs
670
RE-RANK SEARCH RESULTS
680

FIG. 6B

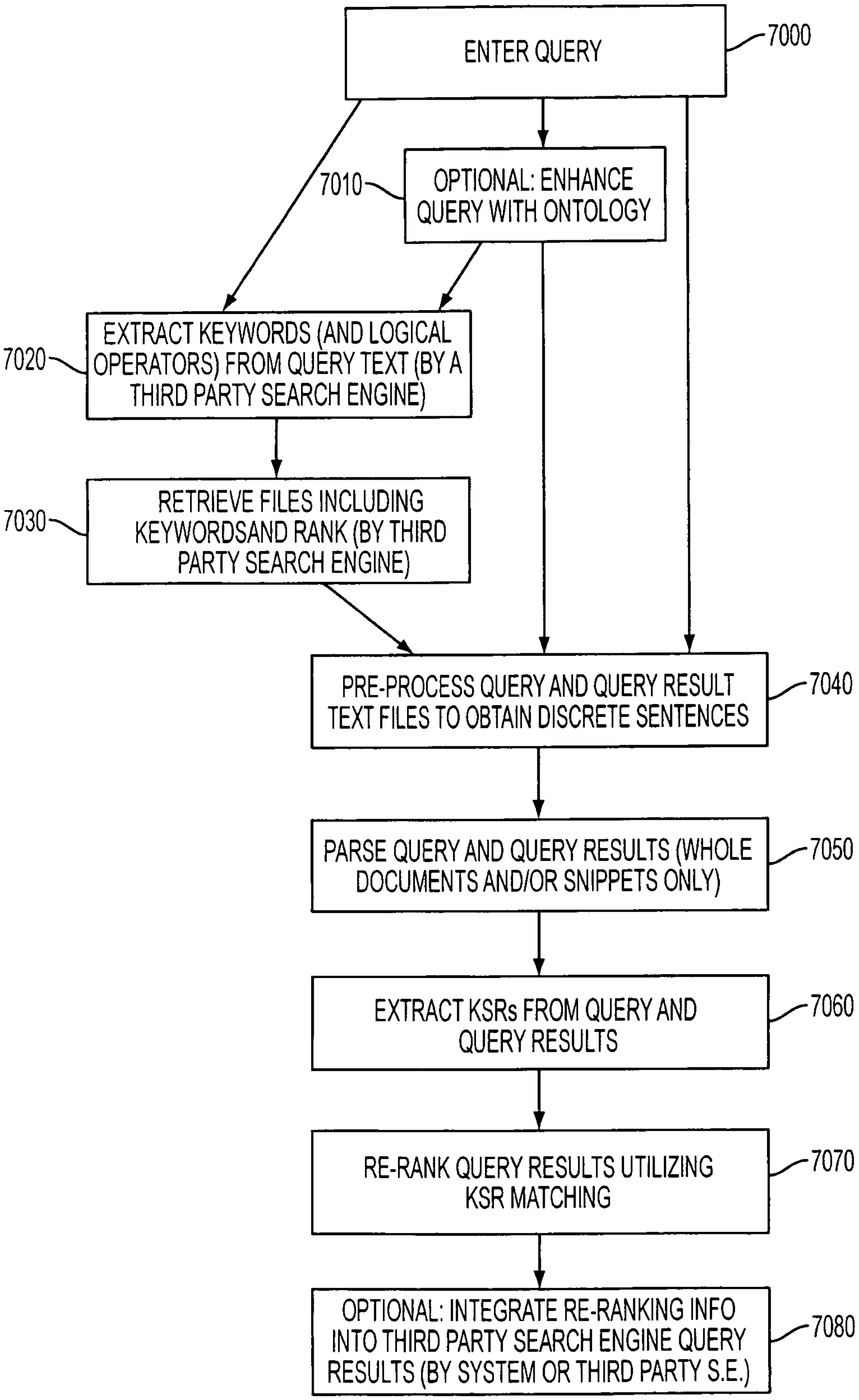

ENTER QUERY
7000
7010
OPTIONAL: ENHANCE QUERY WITH ONTOLOGY
7020
EXTRACT KEYWORDS (AND LOGICAL OPERATORS) FROM QUERY TEXT (BY A THIRD PARTY SEARCH ENGINE)
7030
RETRIEVE FILES INCLUDING KEYWORDSAND RANK (BY THIRD PARTY SEARCH ENGINE)
PRE-PROCESS QUERY AND QUERY RESULT TEXT FILES TO OBTAIN DISCRETE SENTENCES
7040
PARSE QUERY AND QUERY RESULTS (WHOLE DOCUMENTS AND/OR SNIPPETS ONLY)
7050
EXTRACT KSRs FROM QUERY AND QUERY RESULTS
7060
RE-RANK QUERY RESULTS UTILIZING KSR MATCHING
7070
OPTIONAL: INTEGRATE RE-RANKING INFO INTO THIRD PARTY SEARCH ENGINE QUERY RESULTS (BY SYSTEM OR THIRD PARTY S.E.)
7080

FIG. 7A

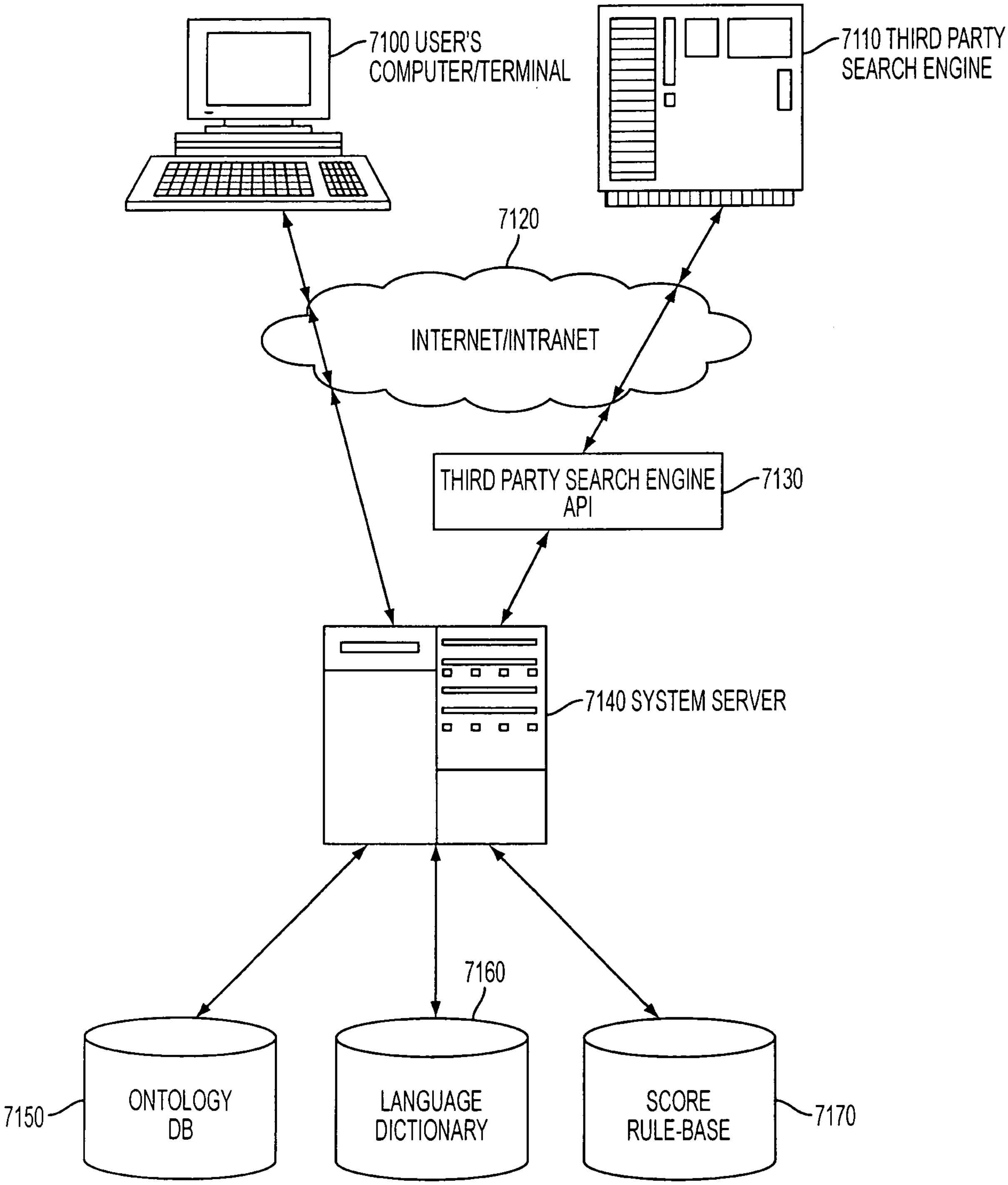

7100 USER'S COMPUTER/TERMINAL
7110 THIRD PARTY SEARCH ENGINE
7120
INTERNET/INTRANET
THIRD PARTY SEARCH ENGINE API
7130
7140 SYSTEM SERVER
7160
7150
ONTOLOGY DB
LANGUAGE DICTIONARY
SCORE RULE-BASE
7170

FIG. 7B

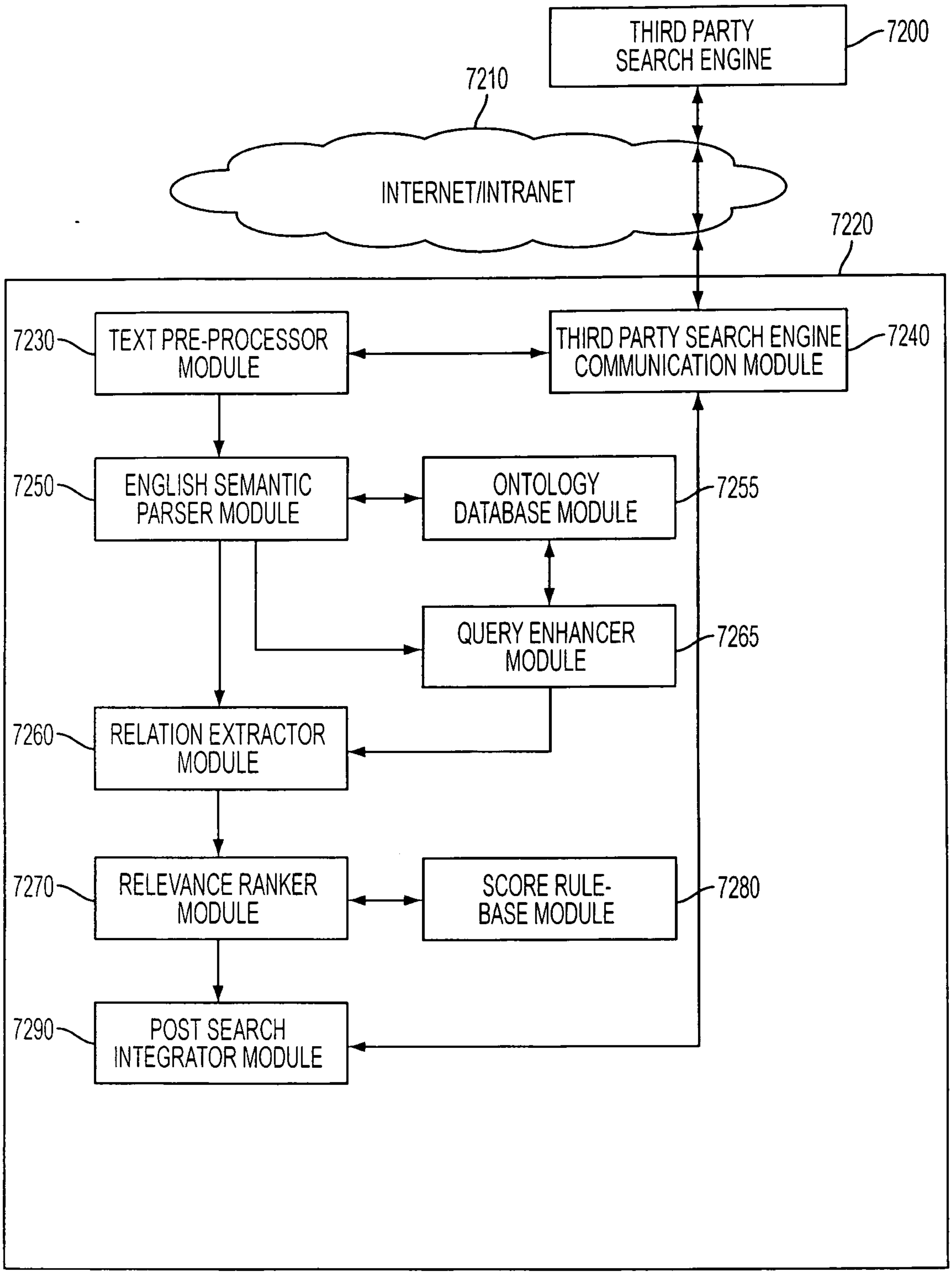
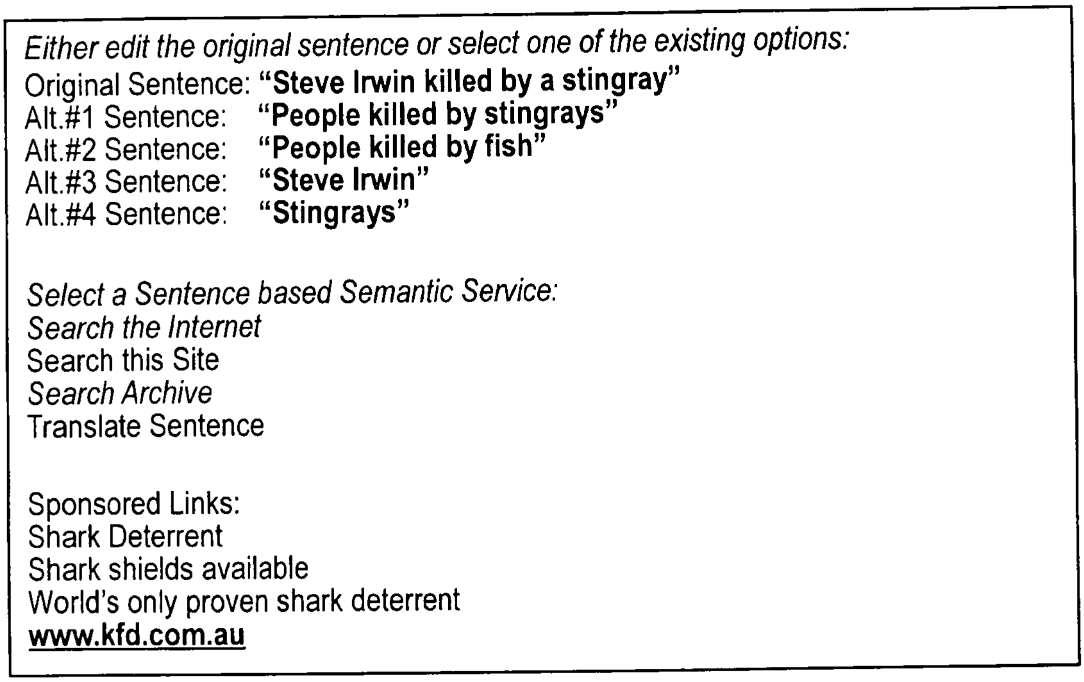

THIRD PARTY SEARCH ENGINE
7200
7210
INTERNET/INTRANET
7220
7230
TEXT PRE-PROCESSOR MODULE
THIRD PARTY SEARCH ENGINE COMMUNICATION MODULE
7240
7250
ENGLISH SEMANTIC PARSER MODULE
ONTOLOGY DATABASE MODULE
7255
QUERY ENHANCER MODULE
7265
7260
RELATION EXTRACTOR MODULE
7270
RELEVANCE RANKER MODULE
SCORE RULE-BASE MODULE
7280
7290
POST SEARCH INTEGRATOR MODULE

FIG. 7C

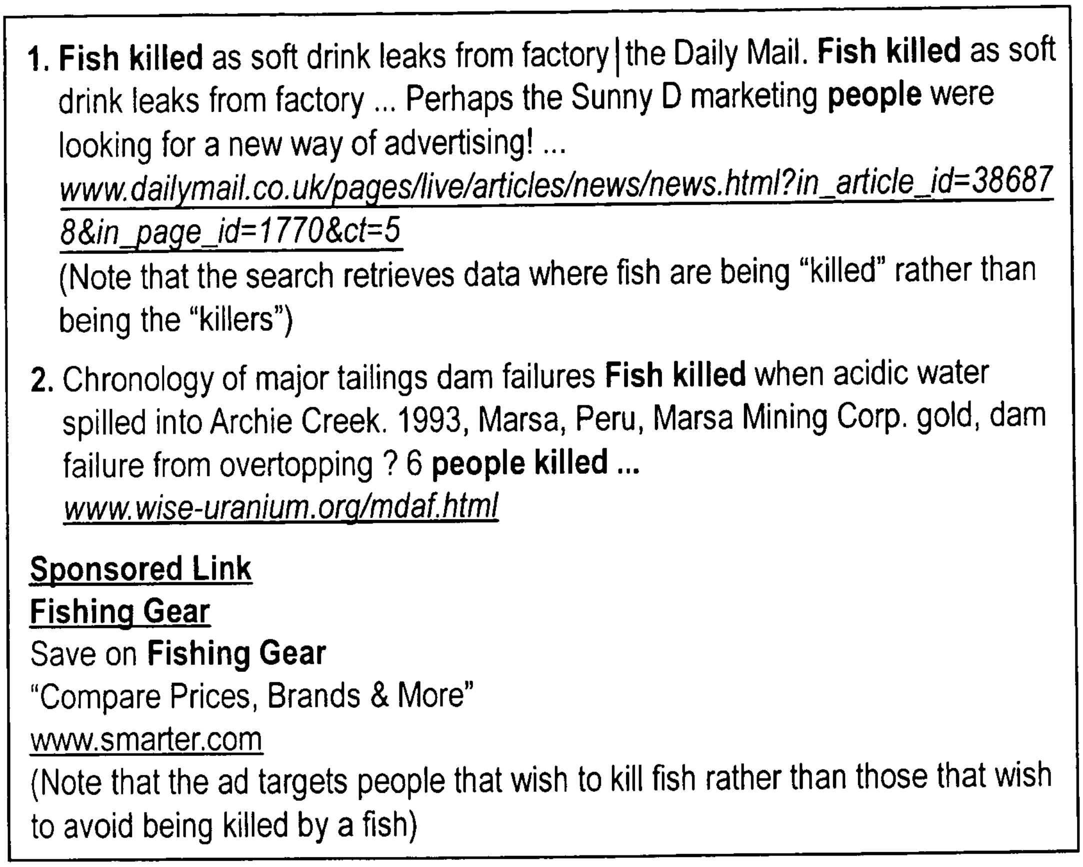


Either edit the original sentence or select one of the existing options:
Original Sentence: **“Steve Irwin killed by a stingray”**
Alt.#1 Sentence: **“People killed by stingrays”**
Alt.#2 Sentence: **“People killed by fish”**
Alt.#3 Sentence: **“Steve Irwin”**
Alt.#4 Sentence: **“Stingrays”**

Select a Sentence based Semantic Service:
Search the Internet
Search this Site
Search Archive
Translate Sentence

Sponsored Links:
Shark Deterrent
Shark shields available
World’s only proven shark deterrent
www.kfd.com.au

FIG. 8



1. **Fish killed** as soft drink leaks from factory | the Daily Mail. **Fish killed** as soft drink leaks from factory ... Perhaps the Sunny D marketing **people** were looking for a new way of advertising! ...
   www.dailymail.co.uk/pages/live/articles/news/news.html?in_article_id=386878&in_page_id=1770&ct=5
   (Note that the search retrieves data where fish are being "killed" rather than being the "killers")
2. Chronology of major tailings dam failures **Fish killed** when acidic water spilled into Archie Creek. 1993, Marsa, Peru, Marsa Mining Corp. gold, dam failure from overtopping ? 6 **people killed ...**
   www.wise-uranium.org/mdaf.html

Sponsored Link
Fishing Gear
Save on **Fishing Gear**
"Compare Prices, Brands & More"
www.smarter.com
(Note that the ad targets people that wish to kill fish rather than those that wish to avoid being killed by a fish)

FIG. 9
RELATED ART