KeyQuery – A Front End for the Automatic Translation of Keywords into Structured Queries

Martin Erwig and Jianglin He

School of EECS, Oregon State University, Corvallis, OR 97331
{erwig,heji}@cs.orst.edu

Abstract: We demonstrate an approach to transform keyword queries automatically into queries that combine keywords appropriately by boolean operations, such as **and** and **or**. Our approach is based on an analysis of relationships between the keywords using a taxonomy. The transformed queries will be sent to a search engine, and the returned results will be presented to the user. We evaluate the effectiveness of our approach by comparing the precision of the results returned for the generated query with the precision of the result for the original query. Our experiments indicate that our approach can improve the precision of the results considerably.

1. Introduction

The most common way to find information on the Internet is to use a search engine and provide a list of keywords describing the sought information. A problem with keyword-based search is that it often returns too many irrelevant results. The use of boolean operators can improve the preciseness of queries considerably. However, there is strong empirical evidence that end users, who are by far the largest group of users of search engines, are not able to use Boolean operators correctly. For instance, Pane and Myers found that sometimes, when people say **and**, they actually mean **or** [11] (for example, "I am interested in blue *and* red cars" usually expresses the interest in cars that are red *or* blue.) This and/or confusion happens very frequently when keywords are used that are closely related in a concept hierarchy (for example, red and blue are both colors).

Based on this observation, we have designed and implemented a front end for search engines, called *KeyQuery*, which groups keywords based on their similarity and inserts the Boolean operator **or** between keywords in one group and connects different groups by **and**. Our approach is based on identifying the relations among the keywords used in a query by using a taxonomy. The transformed structured queries produce more relevant results, in particular, in the first couple of returned pages.

For example, a user wants to buy a Honda car and he likes red and blue colors. So he might input "red blue Honda" to search the Internet to find some information. The default relation between keywords for most search engines is **and**. So the web pages that contain all the three words "red", "blue", and "Honda" will be returned as a result. Other useful pages, which contain only "red" and "Honda", or only "blue" and

"Honda", but not all the three words, will not be in the results. The user might miss a lot of useful information. Aa another example, suppose a user wants to find the biographies of some classical musicians, and he types "biographies Mozart Debussy Beethoven Liszt Tchaikovsky" on the Internet, he may get only a few web pages because only a few web pages contain all the keywords. In fact, there are many web pages on the Internet that contain information about biographies of one or several of the musicians. All keywords except the first are related by the category "musician". Therefore, our goal is to add the boolean operator **or** between these keywords, and transform the whole query into "biography **and** (Mozart **or** Debussy **or** Beethoven **or** Liszt **or** Tchaikovsky)". This query returns more relevant information, in particular, on the first two pages of results.

The query interface is shown in Figure 1 on the left. A list of entered keywords will be transformed behind the scenes into a structured query, which will then be executed by a search engine capable of dealing with boolean operators (we are currently using Google [7]). The results will then be presented to the user in the same browser window. For example, Figure 1 shows on the right the result of the above musician/biography query.

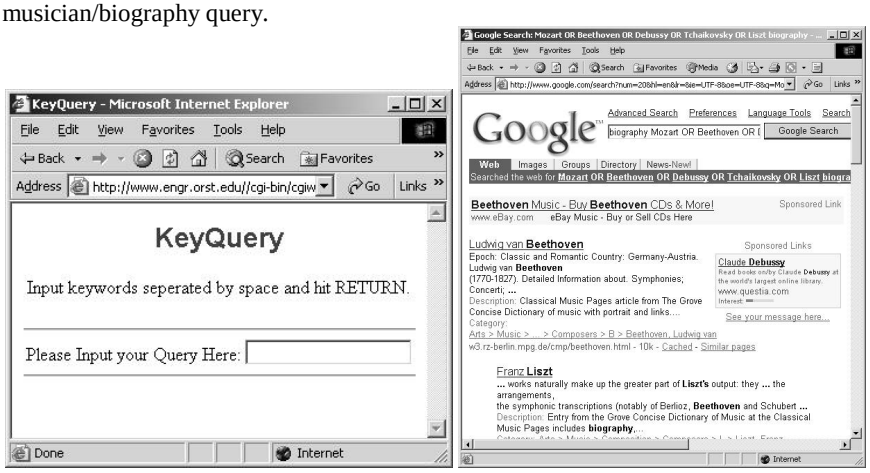
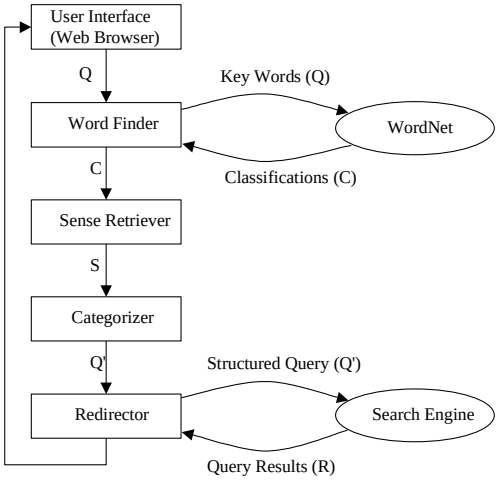


Fig. 1. User interface and example query results

The rest of this paper is structured as follows. In Section 2 we discuss the general approach and give an overview over the query system. In Section 3 we introduce our similarity measures for keywords and describe the concept of a distance matrix, which is used by the algorithm to form groups of similar keywords, introduced in Section 4. In Section 5 we evaluate our approach by comparing search results for keyword queries with the results for the transformed queries. We draw some conclusions and give remarks on future work in Section 6.

2. System Overview

Our system KeyQuery can be considered as query preprocessing front end for web browsers. We require access to the taxonomy WordNet [14], either installed locally or through the Internet. The query processing is performed in several steps. Step 1: In a dedicated user interface, users can input a query that consists of one or more keywords. We call this query Q. Step 2: The user interface sends the query Q to the "Word Finder", which sends all the keywords to WordNet and obtains a classification file C for each keyword. Step 3: The "Sense retriever" extracts all the senses S for each keyword from the classification files obtained from WordNet. Step 4: The "Categorizer" calculates similarity between every pair of keywords and categorizes them based on these similarities by using a threshold-based algorithm. A new query Q', combined with boolean operators **and** and **or,** will be generated based on the categorized keywords. Step 5: The "Redirector" sends Q' to a search engine that supports Boolean operators. It obtains the results R from the search engine and sends the results back to the user interface. The system architecture is summarized in Figure 2.



Q	Original keyword query
C	Classification files downloaded from WordNet for each keyword
S	The senses of all the keywords
Q'	Transformed query
R	The results returned by the search engine

Fig. 2. System structure

To categorize a list of keywords into different groups, we use WordNet's lexical database as the taxonomy in a semantic similarity measurement task. WordNet was

developed by the Cognitive Science Laboratory at Princeton University and has been used in many different projects [2, 8, 13]. WordNet essentially provides information regarding four relationships [9]: 1. Synonymy, 2. Antonymy, 3. Meronymy, and 4. Hyponymy. Our system currently exploits only the hponymy/hypernymy relationship, which basically represents an IS-A relationship. For example, "sedan" is a kind of "car".

In the IS-A hierarchy provided by WordNet, each word has one or several *senses.* A *sense* is one meaning of a word. As an example, we show sense 3 of the word "red" in Figure 3. We can see that the structure of a word sense is a DAG.

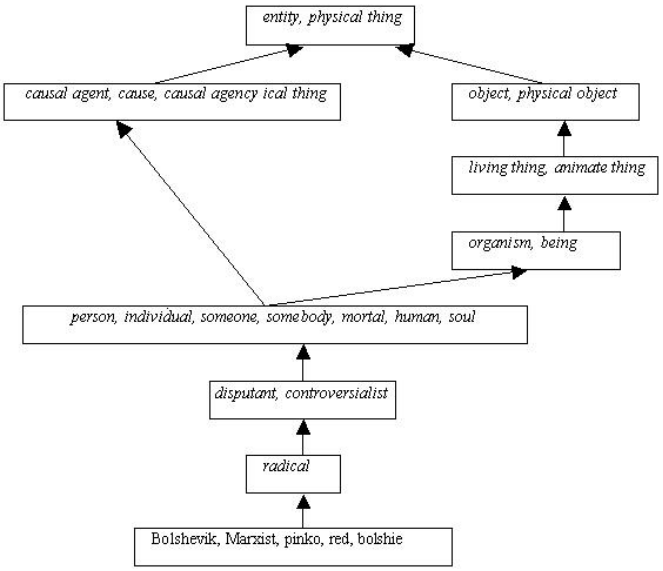


Fig. 3. Structure of a word sense

3. Similarity

The groups that are derived from a list of keywords depend on the similarity between those words. The degree of similarity of two words A and B is determined by their *conceptual distance*, which we define as follows.

Definition 1 (*Conceptual Distance*): If there is not any common ancestor for word A and word B, then $D_{AB} = D_{max}$, where D_{max} is the maximal value of conceptual distance; otherwise, if there is a common ancestor at level i of sense m in word A's ancestors and level j of sense n in word B's ancestors, then we define:

$$D_{AB} = C * (m + n - 2) + i + j$$

It is easy to verify that if the distance between A and B is D_{AB}, the distance between B and A is also D_{AB}. For every pair of common ancestors, we apply this formula and get the conceptual distances D_1, D_2, D_3, …, D_p. Let D_{AB} = min (D_1, D_2, D_3, …, D_p). C is a constant, called "sense amplifier", used to enlarge the conceptual distance of two words linearly with the sense levels. We use this sense amplifier to penalize similarity in later senses since WordNet orders senses by their frequencies of usage.

The presented formula presents a compromise between simplicity and expressiveness; our experiments have shown that it yields good results. In particular, we have found that C = 4 is a good choice. Let us explain the use of C by the following example. Consider the query "red blue black coat". We can find that level 1 of sense 1 of the word "red" has a common ancestor "chromatic color" with level 1 of sense 1 of the word "blue". So by definition, their conceptual distance is 2. However, the conceptual distance between the word "red" and the word "black" is 4 since the closest ancestor of "black" is "achromatic color". Both, "red" and "black", are expected to be in the same category "color" by most people. By our definition, the conceptual distance between "blue" and the word "coat" is 8. If we do not introduce the sense amplifier and consider all senses have the same role and only count the level difference, the distance between "Blue" and "coat" is also 2. However, "blue" is not similar to "coat" because most people will think that "blue" is a kind of "color" like "red" while "coat" is a kind of "clothing". Thus we need a sense amplifier to enlarge the distance if two words have no common ancestor in their first senses.

WordNet lists as the first sense the most often used meaning for a word. WordNet ranks senses of a word by the frequency of usage of that sense. To determine if two words are similar, we also need a threshold value. We call this value *group distance* and write Δ for it. Only if the conceptual distance between two words is less than Δ, they should be in the same group and we say that they are "similar"; otherwise, they are called "different". It is very important to choose an appropriate group distance. If the group distance is too small, then a lot of similar words will not be correctly grouped. If the group distance is too large, then more words will be similar to each other than normally considered by users. The group distance also depends on the value of the sense amplifier, which must, in particular, allow similarity between words on second-level senses. The key point is that we must find a balance that can fairly treat the weight of senses as well as the group distance. For C = 4 we have found that Δ = 7 yields good results.

We can calculate the conceptual distance between each pair of keywords by using the definition of conceptual distance. Given a list of keywords K_1, K_2, …, K_n, we can put the distances into a so-called "distance matrix". (Since the conceptual distance is symmetric, we need only half of the matrix.)

Distance	**K_1**	**K_2**	**…**	**K_n**
K_1	0	D_{12}	…	D_{1n}
K_2		0	…	D_{2n}
…			0	…
K_n				0

4. Keyword Categorization

In grouping similar words together, we have to resolve the following ambiguity. Consider the keyword query "yellow orange apple" and assume $D_{max} = 99$, $\Delta = 7$, and $C = 4$. The conceptual distance between "yellow" and "orange" is 6, so "yellow" and "orange" are similar and *could* be in the same group. Meanwhile, the conceptual distance between "orange" and "apple" is 3, so "orange" and "apple" are also similar and also *could* be in the same group. The problem is, "orange" could be in the same group with "yellow" and also could be in the same group with "apple". But "yellow" and "apple" could not be in the same group since their conceptual distance is 99 and greater than the group distance. To solve this conflict, we employ the following rule:

Priority Grouping Rule: Words with smaller conceptual distance have a higher priority to be grouped.

This priority grouping rule is realized in the following algorithm by sorting the keyword pairs that are to be grouped by their conceptual distances and processing them in order of increasing distance. We write $S \oplus X$ for $S \cup \{ X \}$.

Categorization Algorithm:

Input: A list of keywords $K = \{K_1, K_2, \ldots, K_n\}$
Output: A set of groups $G = \{ g_1, g_2, \ldots, g_s \}$
Method: First, we determine the distance matrix D for K, with entries $D_{12}, D_{13}, \ldots, D_{1n}, D_{23}, \ldots, D_{(n-1)n}$. Let L be the sorted list that contains the $(n^2 - n) / 2$ elements (the upper right half of the matrix) in D.
If $n = 1$, there is only one element, so we get only one group, i.e., let $G = \{\{K_1\}\}$.
If $n > 1$, we proceed as follows:

Let $G = \varnothing$.
For every element in L do
If $D_{ij} < \Delta$ then S
Consider the following cases:
Case 1: $\forall\, g \in G, K_i, K_j \notin g$. Then let $G = G \oplus \{ K_i, K_j \}$
Case 2: $\exists\, g_m \in G, K_i \in g_m$ and $\forall\, g \in G, K_j \notin g$.
If $\forall\, K_a \in g_m, D_{aj} < \Delta$ then let $G = G - \{ g_m \} \oplus (g_m \oplus K_j)$
else let $G = G \oplus \{K_j\}$
Case 3: Symmetric to Case 2 with K_i, K_j swapped.
Case 4: $\exists\, g_m \in G, K_j \in g_m$ and $\exists\, g_r \in G, K_i \in g_r$ and $g_m \neq g_r$.
If $\forall\, K_a \in g_m, D_{ai} < \Delta$ and $\forall\, K_b \in g_r, D_{rj} < \Delta$ then
let $G = (G - \{g_m, g_r\}) \oplus (g_m \cup g_r)$
If $D_{ij} \geq \Delta$ then
If $\forall\, g \in G, K_i \notin g$ then let $G = G \oplus \{ K_i \}$
If $\forall\, g \in G, K_j \notin g$ then let $G = G \oplus \{ K_j \}$
Sort all the groups in G by the smallest indices of their keywords in ascending order.
Return G. □

Here is an example for case 2 when $D_{ij} < \Delta$. Consider the query "yellow orange apple". According this algorithm, "orange" and "apple" will be grouped first since they have the smallest conceptual distance 3. Then we find that the conceptual distance between "orange" and "yellow" is smaller than Δ. But because the conceptual distance between "yellow" and "apple" is 99 and is greater than Δ, the word "yellow" will be placed in another group. Cases 3 and 4 are very similar to case 2.

So far, we can categorize a list of keywords into different groups based on the distance matrix. We also sort these groups by the smallest indices of their keywords in ascending order since in some search engines, such as Google and Yahoo, the order of the keywords will affect the results. We try to preserve the original order of the keywords. Finally, the groups of keywords will be transformed into a query by inserting **or** between all keywords in a group and **and** between all groups.

5. Evaluation

The two most common measures of retrieval effectiveness are *recall* and *precision.* Recall measures the percentage of relevant documents in a collection that are actually found by a retrieval system—i.e., the ratio of the number of relevant retrieved documents over the total number of relevant documents contained in the collection. Precision measures the percentage of retrieved documents that are judged to be relevant to the original request—i.e., the ratio of the number of relevant retrieved documents over the total number of retrieved documents. Normally there is a trade-off between recall and precision.

It is very common that a user gets millions of results returned from a keyword search. However, most users only go over the first one or two pages. Therefore, precision in the first two pages is crucial to search engines. The ranking of results by search engines has an influence on the precision in the first pages. In particular, ranking pages by their popularity yields much better results than by using just a count of number of contained keywords (cf. Google's PageRank™ [16] techonology).

For our system we can observe that if no two keywords are similar in a query, KeyQuery will **and** all the keywords, so the results will be the same as for the original query because the boolean operator **and** is the default relation between keywords for most search engines. From this it follows that in these cases the recall and precisions of KeyQuery and the other search engines are the same since KeyQuery does not change the original query. But if some keywords are similar to each other, KeyQuery will insert the relation **or** between those keywords. The transformed query will be different from the original query. Since we add the **or** operator between keywords, the infamous trade-off between precision and recall will hit: if the overall recall is increased, then the overall precision will be decreased as we retrieve more results than the original ones.

If we redirect our transformed query to Google, what will happen? Let us consider a query "A1 A2 B1". We assume that A1 and A2 are similar words and are grouped together by our algorithm. B1 is in a different group. The original results from Google

are ‘ranked web pages that contain “A1 A2 B1” by their popularity. The new results are ‘ranked web pages that contain “A1 B1” or “A2 B1” by their popularity’. Do we improve the precision for the first two pages?

We explain the effect of our approach for this example in Figure 5. Assume that a user looks only at the first five results. Table 1 holds the results of original query. We assume that R1, R2, R3, R4 and R5 are the ranked results and 2100, 1300, 900, 500 and 300 are their PageRank values, respectively. Now we reorganize the query into two queries: “A1 B1” and “A2 B1”. Table 2 and 3 hold envisioned results of the queries “A1 B1”and “A2 B1”, respectively. Table 4 holds the results of the transformed query “(A1 **or** A2) **and** B1” that we get from our KeyQuery system. All the results in these tables are ranked by their corresponding PageRank values. Some of the original query results are in the results for the reorganized queries. For example, R1 = L2 = K4. The results in table 4 are those results with highest PageRank values chosen from table 2 and table 3. We can observe that the constraint of three keywords prevents some of the possibly more relevant results, i.e., the results with larger PageRank values, from appearing in the initial part of the result list for the original query. From these tables, we can conclude that the first five results for the transformed query are possibly more relevant than those for the original query.

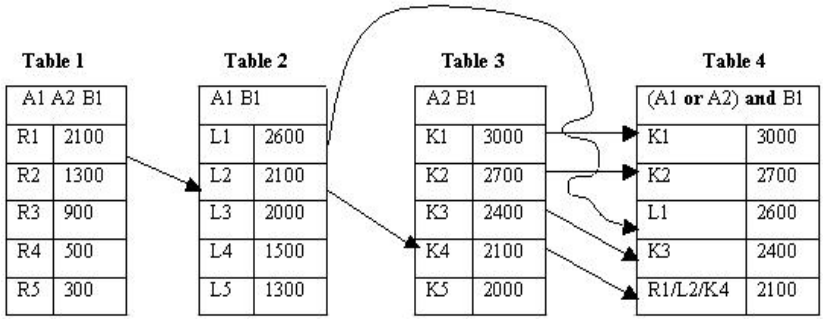


Fig. 5. Abstract example

As another example, consider again the query “biography Mozart Debussy Beethoven Liszt Tchaikovsky” (This is query 1). Since Google supports boolean operators, we have used Google as our back end search engine, and we have checked the relevance of the first 20 returned links. A group of students from different departments have judged the search engine results for relevance. We have averaged all the precision rates. We have obtained an average of 9.2 relevant links with the original query, which means the precision is 46%. With the transformed query, the precision is 96%. As another example, consider the query “admission engineering MIT Stanford”. With the original query, the precision is only 26%. However, the transformed query obtained by our system yields a precision of 98%. Other queries used in our evaluation are:

3. California Hawaii Vacation package
4. Seattle Chinese Japanese Mexican restaurant

5. Red blue Honda coupe
6. Diamond ruby emerald wedding ring
7. Smoked pork beef recipe
8. How to plant apple pear orange tree
9. Cat dog training program
10. BMW convertible sedan dealer Portland

The results are shown in Figure 6.

	Precision (%)		
Query#	Original Query	Transformed Query	Improvement
1	46	96	109
2	26	98	277
3	87	95	9
4	69	94	36
5	60	93	55
6	78	97	24
7	85	100	18
8	50	57	14
9	63	69	10
10	35	95	171
Avg. Precision (%)	60	89	72

Fig. 6. Comparison table

6. Conclusions and Future Work

We have developed the system KeyQuery that can improve the precision of search results for keyword queries considerably. The approach taken was to use a taxonomy to automatically insert boolean operators **and** and **or** between keywords. Since the generated queries are in boolean form, we can apply our approach to any search engine or database that support boolean operators.

One of the next steps is to perform a systematic study of the precision improvements that can be achieved by our system. Moreover, the system can be improved in several ways. One route for future work is to include adjectives and adverbs from WordNet.

Another area of future work is to construct a taxonomy besides WordNet. The reason is that WordNet does not include many common words, like brand names, such as Honda. However, since manually maintaining such taxonomies is almost impossible, we currently consider a strategy of generating and updating these taxonomies by web queries themselves.

References:

[1] AltaVista, http://www.altavista.com.

[2] E. Agirre and G. Rigau, "Word Sense Disambiguation Using Conceptual Density", In Proceedings of the 16th International Conference on Computational Linguistics (Coling '96), Copenhagen, Denmark, 1996.

[3] N.J. Belkin, R.N. Oddy, and H.M. Brooks, "ASK for information retrieval: Part 1. Background and theory", Journal of Documentation, 38, pp. 61–71, 1982.

[4] C. Silverstein, M. Henzinger, H. Marais, and M. Moricz, "Analysis of a very large web search engine query log", SIGIR Forum, 33 (1): 6-12, 1999. Previously available as Digital System Research Center TR 1998-014 at http: //www.research.digital.com/SRC

[5] D.D. Lewis and K. Sparck Jones, "Natural Language Processing for Information Retrieval", Comm. ACM, Vol. 39, No. 1, Jan. 1996, pp. 92–101.

[6] M. Erwig. and M.M. Burnett, "Adding Apples and Oranges", 4th Int. Symp. on Practical Aspects of Declarative Languages, LNCS 2257, 173-191, 2002.

[7] Google, http://www.google.com.

[8] J.J. Jiang and D.W. Conrath, "Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy", in Proceedings of ROCLING X (1997) International Conference on Research in Computational Linguistics, Taiwan, 1997.

[9] G. Miller, (1990) "Five papers on WordNet", Special Issue of International Journal of Lexicography 3(4).

[10] N. Guarino, C. Masolo, and G. Vetere, "OntoSeek: Content-Based Access to the Web", IEEE Intelligent Systems, 14(3), 70--80, (May 1999).

[11] J.F. Pane and B.A. Myers, "Tabular and Textual Methods for Selecting Objects from a Group", VL 2000: IEEE International Symposium on Visual Languages. IEEE Computer Society, September 10-13 2000, pp. 157-164. Seattle, WA.

[12] R. Prieto-Diaz, "Implementing Faceted Classification for Software Reuse", Comm. ACM, Vol. 34, No. 5, May 1991, pp. 88–97.

[13] P. Resnik, "Using Information Content to Evaluate Semantic Similarity in a Taxonomy", Proceedings of the 14th International Joint Conference on Artificial Intelligence, Vol. 1, 448-453, Montreal, August 1995.

[14] WordNet, http://www.cogsci.princeton.edu/~wn/

[15] N. Kurtonina and M. de Rijke, "Classifying description logics", In Proceedings DL'97, 1997.

[16] PageRank, http://www.google.com/technology/PageRank.html