Database Selection for Longer Queries

Wensheng Wu[1], Clement Yu[2], and Weiyi Meng[3]

[1] University of Illinois at Urbana-Champaign, Urbana, IL 61801 wwu2@uiuc.edu
[2] University of Illinois at Chicago, Chicago, IL 60612 yu@cs.uic.edu
[3] SUNY at Binghamton, Binghamton, NY 13902 meng@cs.binghamton.edu

1 Introduction

Given the enormous amount of information now available on the Web, search engines have become the indispensable tools for people to find desired information from the Web. These search engines can be classified into two broad categories by the extent of their coverage. In the first category, we have the general-purpose search engines, such as Google and Yahoo, which have been attempting to index the *whole* Web and provide a search capability for *all* Web documents. Unfortunately, these *centralized* search engines suffer from several serious limitations such as the poor scalability and the difficulty in maintaining the freshness of their contents [1].

In the second category, we have the hundreds and thousands of search engines on confined domains which include the pervasive organization-level search engines and a large number of specialty search engines, such as CiteSeer (a technical report search engine) and mwsearch.com (a search engine for medical information), to name just a few [10]. Since each of these *local* search engines focuses on a much smaller portion of the Web, they are more scalable than the general-purpose search engines and it is much easier to keep their index data up to date. In this paper, we consider the *metasearch engine* approach, a highly scalable alternative to provide the search capability for the entire Web.

A metasearch engine is a system that supports unified access to multiple local search engines. It does not maintain its own index on Web documents but a sophisticated metasearch engine often maintains characteristic information about each underlying local search engine in order to provide better service. There are several serious challenges to implement an effective and efficient metasearch engine. Among the main challenges, the *database selection problem* is to identify, for a given user query, the local search engines that are likely to contain useful documents for the query. The *collection fusion problem* is to retrieve documents from selected databases and then merge these documents with the objective of listing more useful documents ahead of less useful ones.

A good metasearch engine should have the retrieval effectiveness close to that as if all documents were in a single database while minimizing the access cost.

In [16, 17] an approach was introduced to perform database selection and collection fusion (see Section 2 for more information). However, the approach works well only for typical Internet queries which are known to be short. It has been found that better search effectiveness can often be achieved when additional terms are added to the initial queries through query expansion or relevance feedback [4, 11, 14]. The resulting queries are usually much longer than the initial queries.

In this paper, we propose a new method (Section 3) to construct database representatives and to decide which databases to select for a given query when the query may be long. One critical difference from the earlier approaches is that the dependencies of words in documents are *captured* in the database representatives. Usually, the dependencies are limited to phrases, especially noun phrases. In spite of the restrictions, the problem seems difficult to conquer. One approach is to employ a probabilistic parser to identify phrases, as exemplified by [5]. Given the extreme varieties of vocabularies in the Web, accuracies may suffer significantly as words from specialized domains may not satisfy ordinary syntax rules. Another approach is to consider enumeration of all possible triplets or quadruplets of terms. Given the huge number of possible terms, this approach is unlikely to be feasible either.

In [14, 15], experimental results were given to demonstrate that it was possible to retrieve documents in distributed environments with essentially the same effectiveness as if all data were at one site. However, the results in [14] depend on a *training collection* which has similar coverage of subject matters and terms as the collection of databases to be searched. In the Internet environment where data are highly heterogeneous, it is unclear whether such a training collection can in fact be constructed. [15] relies on the properly clustering of the documents. In the Internet environment, it is not clear whether it is feasible to cluster large collections and to perform re-clustering for dynamic changes. Please see [7] for a more comprehensive review of other work in the metasearch engine and distributed information retrieval area.

2 A Framework for Database Selection and Collection Fusion

A query in this paper is simply a set of words submitted by a user. It is transformed into a vector of *terms* with *weights* [9], where a term is essentially a content word and the dimension of the vector is the number of all distinct terms. The weight of a term typically depends on two factors: tf (term frequency) and idf (inverse document frequency) [9]. A document is similarly transformed into a vector with weights. The similarity between a query and a document, denoted by $\text{sim}(q, d)$, can be measured by the dot product of their respective vectors. Often, the vectors are normalized by division with their

lengths such that the similarity values obtained are between 0 and 1. The corresponding weights in normalized vectors are called *normalized weights*. The similarity function with such a normalization is known as the Cosine function [9]. When the idf of each term is computed based on the global document frequency of the term (i.e., the number of documents containing the term across *all* databases), it is called global idf (or gidf) and the computed similarities are global similarities.

We now review a framework for database selection and collection fusion which was first introduced in [16, 17]. Suppose a user is interested in retrieving the m most similar documents for a query q from N databases $D_1, D_2, \ldots, D_N$, where m is any positive integer. This framework can be summarized into one definition on optimal database ranking, a necessary and sufficient condition for ranking databases optimally and an algorithm for integrated database selection and collection fusion based on ranked databases.

Definition 1. *A set of N databases is said to be optimally ranked in the order $[D_1, D_2, ..., D_N]$ with respect to a given query q if for every positive integer m, there exists a k such that $D_1, D_2, ..., D_k$ contain the m most similar documents and each D_i, $1 \leq i \leq k$, contains at least one of the m most similar documents.*

Note that the ordering of the databases with respect to a query and a given similarity function may be different from their ordering with respect to the same query but with a different similarity function. However, the following proposition holds for any similarity functions.

Proposition 1. *[16] Databases $D_1, D_2, ..., D_N$ are optimally ranked in the order $[D_1, D_2, ..., D_N]$ with respect to a given query q if and only if $msim(q, D_1) > msim(q, D_2) > ... > msim(q, D_N)$, where $msim(q, D_i)$ is the global similarity of the most similar document in database D_i with the query q.* □

After the databases have been ordered, a merging algorithm, known as OptDocRetrv, was developed to perform database selection and collection fusion [16, 17]. This algorithm is sketched as follows. Suppose the first s databases have been selected (s is 2 initially). Each of these selected search engines returns the actual global similarity of the most similar document to the metasearch engine which computes the minimum, denoted *min_sim*, of these s values. Each of the s search engines then returns to the metasearch engine those documents whose global similarities are greater than or equal to *min_sim*. Note that at most m documents from each search engine need to be returned to the metasearch engine. If m or more documents have been returned from the s search engines, then they are sorted in descending order of similarity and the first m documents are returned to the user. Otherwise, the next database in the determined order will be selected and the above process is repeated until at least a total m documents are returned to the user.

3 Ranking Databases with Reduced Document Vectors

In this section, we propose our new method for database selection based on the framework described in Section 3. A key step is to estimate the global similarity of the most similar document in each database for any given query. We present a new type of database representative, which we believe is more capable of handling longer queries in estimating the similarity of the most similar document than existing approaches.

Consider a term t_i and a local database D_j. Let $\text{mnw}_{i,j}$ and $\text{anw}_{i,j}$ be the *maximum normalized weight* and the *average normalized weight* of t_i over all documents in D_j, respectively. Suppose a query q is represented by vector $(q_1 * \text{gidf}_1, \ldots, q_n * \text{gidf}_n)$, where q_i and gidf_i are the tf and the global idf of term t_i in query q, respectively. Then the global similarity of the most similar document of database D_j with respect to q can be estimated by [16]:

$$\max_{1 \le i \le n} \{q_i * \text{gidf}_i * \text{mnw}_{i,j} + \sum_{k=1, k \ne i}^{n} q_k * \text{gidf}_k * \text{anw}_{k,j}\}/|q| \qquad (1)$$

The intuition for having this estimate can be described as follows. The most similar document in a database is likely to have the maximum normalized weight on one of the query terms, say term t_i. This yields the first half of the above expression within the braces. For each of the other query terms, the document takes the average normalized value. This yields the second half. Then, the maximum is taken over all i, since the most similar document may have the maximum normalized weight on any one of the n query terms.

The reason that Formula (1) can be inaccurate is explained as follows. Consider a query having term t and a set of other terms T. Formula (1) would use the maximum normalized weight of term t and the average normalized weights of the terms in T. It was observed in [6] that the average normalized weight of a term is usually one to two orders of magnitude smaller than the maximum normalized weight, because the average is computed over all documents in the database, including those that do not contain the term. For queries with 2 or 3 terms, the inaccuracy is tolerable as the maximum normalized weight is usually much larger than other weights. However, for queries with more terms, the problem becomes more serious.

3.1 A New Type of Representatives

One can observe that, in the above discussion, if the document d (having the maximum normalized weight of a query term) itself were kept in the database representative, then the precise similarity between d and the query would be calculated, and as a result, there would be no need to estimate this similarity. (Note, however, this document may not necessarily be the most similar document to the query in that database.) Based on this observation, we propose the following representative for a database.

Let $T(D)$ denote the set of terms that appear in database D. For each term t in $T(D)$, let $d(t)$ be the set of documents in D that have the *maximum normalized weight* on t across all documents in D. Although it is possible that $d(t)$ could contain more than one document, in practice $d(t)$ usually contains just a single document. For ease of discussion, we assume that $d(t)$ contains exactly one document and $d(t)$ denotes this document. Since $d(t)$ is a document, it usually contains many terms, many of which are somewhat unrelated to t. We consider only terms that are likely to form phrases with t. It is well known that components of a phrase usually occur within 3 words apart. Thus, any term t_1 in which all occurrences of t_1 in d are more than 3 words apart from any occurrence of t in d is removed. This forms a reduced document $d'(t)$. Now the new representative of database D can be defined as

$$Rep(D) = \{(t, d'(t)) | t \in T(D)\} \tag{2}$$

In other words, for each distinct term in the database, the representative contains the term as well as the reduced document that has the maximum normalized weight of the term.

3.2 Estimation of the Similarity of the Most Similar Document

Based on this database representative, the similarity of the most similar document can be estimated as follows. Suppose q is a query with k terms $(t_1, ..., t_k)$. For each term t in the query, if t appears in the database, then document $d'(t)$ can be identified in the database representative and the similarity between q and $d'(t)$ can be computed. If t does not appear in the database, then t is not used in the estimation process for this database. The similarity of the most similar document in the database can be estimated to be the largest similarity between q and $d(t_i)$, $1 \leq i \leq k$, that is:

$$msim(q, D) = \max_{1 \leq i \leq k \wedge t_i \in T(D)} \{sim(q, d'(t_i))\} \tag{3}$$

Note that $msim(q, D)$ as computed by the above formula may not be 100% accurate. This is because (a) the reduced vectors $d'(t)$ are used instead of the original document vectors $d(t)$ and (b) Formula (3) considers only those documents that have the maximum normalized weight for at least one query term. It is possible that a document in D does not have the maximum normalized weight for any query term but its similarity with the query is higher than that produced by Formula (3). In this case, the similarity produced by Formula (3) won't be the same as $msim(D, q)$. Our experimental results (see Section 4) indicate that Formula (3) is reasonably accurate in obtaining the similarity of the most similar document in a database.

3.3 Integration of New Representatives

To achieve storage and computational efficiencies, instead of keeping a separate representative for each database, a single integrated representative for

all databases is constructed [13]. Suppose the metasearch engine is designed to search no more than r search engines for any given query (a small r, say 30, is likely to be sufficient for most users if relevant search engines can be selected. In the Internet environment, most users examine a small number of documents, say no more than 30 documents per query.). If this is the case, it will be sufficient to store, for each term, the r reduced document vectors from different databases, that have the overall r largest maximum normalized weights and the corresponding database identifiers in the representative.

3.4 Capturing Additional 2-Term Phrases

When Formula (3) is applied to our integrated representative to estimate the similarity of the most similar document for each of the r most promising databases, there are potential inaccuracies. These inaccuracies can be due to (1) not all meaningful phrases are captured by our representative, i.e., certain phrases are not in any of our reduced document vectors; (2) although a phrase is in our representative, not all of the r highest maximum normalized weights for that term are kept in our representative.

To provide partial remedy to these situations, we provide the following process to identify additional *2-term phrases*, where the two terms occur in adjacent locations in a query. Let two such terms be t_i and t_j. For a database D there are two documents, one having the maximum normalized weight on t_i denoted by $S = (s_1, \ldots, s_i, \ldots, s_j, \ldots)$ and another document having the maximum normalized weight on t_j denoted by $R = (r_1, \ldots, r_i, \ldots, r_j, \ldots)$. For a document $d = (d_1, \ldots, d_i, \ldots, d_j, \ldots, d_m)$ in a database D, if $\max_{d \in D}\{gidf_i * d_i + gidf_j * d_j\} > \max\{gidf_i * s_i + gidf_j * s_j,\ gidf_i * r_i + gidf_j * r_j\}$, then the co-occurrences of t_i and t_j in d is very high (in fact higher than their co-occurrences in S and in R) and as a consequence, t_i and t_j will be considered as a *2-term phrase*. We then compute, for each such *2-term phrase*, its maximum normalized weight for each database. Then, only the top r maximum normalized weights together with the corresponding database IDs where these maximum normalized weights are achieved are stored. Thus, our integrated representative consists of (1) the top r reduced document vectors for each individual term, together with their database IDs and (2) the top r maximum normalized weights for each *2-term phrase* (which is constructed as described above) together with their database IDs.

Intuitively, a database selection method is effective if the most desired documents are contained in a relatively small number of databases selected by this method. In Section 4, we will conduct experiments to evaluate the effectiveness of our method based on more rigorous measures. The proposition below shows that for any single-term query (which constitutes about 30% of all Internet queries [2]), the local databases selected by the integrated representative are guaranteed to contain the m most similar documents in all databases with respect to the query when $m \leq r$.

Proposition 2. *For any single-term query, if the number of documents desired by the user, m, is less than or equal to r — the number of documents stored in the integrated representative for the query term, then all the m most similar documents to the query are contained in the r local databases containing the r stored documents for the query term.* □

4 Experimental Results

In this section, we report some experimental results. 221 databases are used in our experiments. These databases are obtained from five document collections on TREC disks 4 & 5 created by NIST (National Institute of Standards and Technology of the US). The five collections are CR (Congressional Record of the 103rd Congress), FR (Federal Register 1994), FT (articles in Financial Times from 1992 to 1994), FBIS (articles via Foreign Broadcast Information Service) and LAT (randomly selected articles from 1989 & 1990 in Los Angeles Times). These collections are partitioned into 221 databases of various sizes ranging from 222 documents (about 2 MB) to 7,760 documents (about 20 MB). A total of about 555,000 documents (about 2 GB in size) are in these databases. There are slightly over 1 million distinct terms in these databases.

Two sets of experiments are conducted. In the first set of experiments, the Cosine similarity function is used and test queries are the 400 queries in the first 8 TREC conferences. A typical TREC query consists of three portions (i.e., *topic*, *description* and *narrative*). We used the topic portion of each query in the first set of experiments. The average length of a query is slightly over 4. In contrast, the average length of real Internet queries is only about 2.2 [3].

In this experiment, we use the following two measures to evaluate the performance of a method to search for the m most similar documents in a set of databases: (1) **cor_iden_doc**: The percentage of correctly identified documents, that is, the ratio of the number of documents retrieved among the m most similar documents over m. (2) **db_effort**: The database search effort is the ratio of the number of databases searched by the algorithm over the number of databases which contain one or more of the m most similar documents. The ratio can be both higher or lower than 1. Note that the first measure indicates effectiveness (quality) of retrieval while the second measure reflects efficiency of retrieval.

For a given set of queries, the measures reported in this paper are averaged over all queries in the set that contain at least one real term. The results obtained from the previous approach [13] and the new integrated representative approach using the Cosine function are given in Table 1 where the *add. phrases* columns show the results with new representatives *and* additional 2-term phrases being captured as described in Section 3.4.

From the table, we can observe that the new integrated database representative gives much higher retrieval effectiveness on longer queries (as mentioned

m	cor_iden_doc (%)			db_effort (%)		
	previous	new	add. phrases	previous	new	add. phrases
5	64.8	87.6	92.4	112.1	120.4	121.2
10	68.8	87.1	92.8	100.3	108.1	109.9
20	72.7	87.0	92.9	92.3	100.9	103.0
30	76.6	87.9	93.7	89.9	97.7	101.1

Table 1. Comparisons of previous and new integrated representatives using Cosine similarity function

m	prec_ratio		db_effort	
	previous	new	previous	new
5	61.8%	98.3%	138.9%	134.3%
10	60.8%	94.9%	107.9%	117.5%
20	65.3%	93.6%	90.2%	105.6%
30	65.1%	95.2%	79.6%	100.8%

Table 2. Comparisons of previous and new representatives using Okapi's similarity function on *long* queries

above, the average length of TREC queries is much larger than that of a typical Internet query). The percentage of correctly identified documents range from 64.8% to 76.6% when the previous integrated database representative is used; the corresponding figures for the new database representative (containing the reduced document vectors) are from 87.0% to 87.9%. When additional 2-term phrases are captured, the accuracy of retrieving the m most similar documents increases consistently over all m's, ranging from 4.8% to 5.9%. In both types of new representatives, the efficiencies of accessing databases vary from 97.7% to 121.2%. In other words, in order to retrieve most of the most similar documents, only 21.2% of additional databases need to be accessed.

In the second set of experiments, we utilize the human relevance assessments available from TREC to evaluate the effectiveness of our database selection algorithm and to further evaluate our methods on the Okapi's similarity function [8] which is known to be among the best performers in TREC competitions. The Okapi's formula is given as follows.

$$\sum_{t \in q} (\log \frac{P - n + .5}{n + .5}) * (\frac{(k_1 + 1)tf}{k_1((1-b) + b\frac{dl}{avgdl}) + tf}) * (\frac{(k_3 + 1)qtf}{k_3 + qtf}) \quad (4)$$

The first term in the summation can be regarded as $gidf$ where P is the total number of documents in the global database (the global database logically contains data from all local databases but physically it does not exist), and n is the number of documents in the global database which contain term t. The second and the third term correspond to the document term weight and the query term weight respectively. Note that tf and qtf are the document and

query term frequency of term t, respectively, dl is the length of the document and $avgdl$ is the average length of documents. Note also that k_1, b and k_3 are 3 empirical constants and the summation is taken over all terms in query q.

The test queries are the 50 queries from TREC-6 ad hoc tasks [12]. Both the title and description portions of the queries are used, the average number of terms per query is about 12 which is much longer than that in the first set of experiments, and none of the new queries has less than 5 terms. In this set of experiments, we also gauge the performance of our method based on the ratio of the average precision of the m documents retrieved by the metasearch engine to the average precision of the m documents retrieved as if all documents were placed in a single database. This new measure is denoted as **prec_ratio**.

The result is shown in Table 2. The new representative utilizes both 2-term and 3-term phrases in the queries. From the table it can be observed that the new representative permits a highly effective retrieval of documents from selected databases on longer queries with *prec_ratio* improved from 60.8-65.3% with the previous representative to 93.6-98.3% with the new representative. This indicates that the same effectiveness can be achieved as that of the retrieval from a centralized database with new representatives.

5 Concluding Remarks

Being able to accurately estimate the similarity of the most similar document in a database for a given query is critical in order to rank databases optimally. Previous solutions were designed to estimate the desired similarity for short queries. In this paper, we proposed a new type of database representative that is more suitable to handle longer queries. The information we use to construct this type of database representative is drastically different from previously proposed approaches. The main difference is that the new representative is better at capturing the dependencies of words in documents. As a result, it can produce more accurate estimates for longer queries. Our experimental results using the TREC collection indicated that the new approach can yield high retrieval effectiveness while maintaining high scalability.

References

1. D. Hawking and P. Thistlewaite. Methods for Information Server Selection. ACM TOIS, 17(1), January 1999.
2. B. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real Life Information Retrieval: A Study of User Queries on the Web. ACM SIGIR Forum, 32:1, 1998.
3. S. Kirsch. The Future of Internet Search: Infoseek's Experiences Searching the Internet. ACM SIGIR Forum, 32(2):3-7, 1998.

4. K. Kwok and M. Chan. Improving Two-Stage Ad-Hoc Retrieval for Short Queries. ACM SIGIR, 1998.
5. E. Lima and J. Pedersen. Phrases Recognition and Expansion for Short, Precision-based Queries on a Query Log. SIGIR, 1999.
6. W. Meng, Z. Wu, C. Yu and Z. Li. A Highly Scalable and Effective Method for Metasearch. ACM TOIS, 2001.
7. W. Meng, C. Yu, and K. Liu. Building Effective and Efficient Metasearch Engines. ACM Computing Surveys, 34(1):48-84, March 2002.
8. S. Robertson, S. Walker and M. Beaulieu. Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive track. TREC-7, 1998.
9. G. Salton and M. McGill. Introduction to Modern Information Retrieval. New York: McCraw-Hill, 1983.
10. http://www.SearchEngineWatch.com.
11. A. Sugiura and O. Etzioni. Query Routing for Web Search Engines: Architecture and Experiments. WWW9 Conference, 2000.
12. E. Voorhees and D. Harman. Overview of the Sixth Text REtrieval Conference. TREC-6, 1997.
13. Z. Wu, W. Meng, C. Yu, and Z. Li. Towards a Highly-Scalable and Effective Metasearch Engine. WWW10, Hong Kong, May 2001.
14. J. Xu and J. Callan. Effective Retrieval with Distributed Collections. SIGIR, 1998.
15. J. Xu and B. Croft. Cluster-based Language Models for Distributed Retrieval. ACM SIGIR, 1999.
16. C. Yu, K. Liu, W. Meng, Z. Wu, and N. Rishe. A Methodology for Retrieving Text Documents from Multiple Databases. IEEE TKDE, 14(6):1347-1361, 2002.
17. C. Yu, W. Meng, K. Liu, W. Wu, and N. Rishe. Efficient and Effective Metasearch for a Large Number of Text Databases. CIKM'99, 1999.
18. C. Yu, W. Meng, W. Wu, K. Liu. Efficient and Effective Metasearch for Text Databases Incorporating Linkages among Documents. SIGMOD'01, May 2001.