A Methodology to Retrieve Text Documents from Multiple Databases*

Clement Yu[1], King-Lup Liu[2], Weiyi Meng[3], Zonghuan Wu[3], Naphtali Rishe[4]
[1] Dept. of EECS, University of Illinois at Chicago, Chicago, IL 60607
yu@eecs.uic.edu
[2] School of Computer Science, Telecommunications and Information Systems
DePaul University, Chicago, IL 60604
[3] Dept. of Computer Science, SUNY – Binghamton, Binghamton, NY 13902
[4] School of Computer Science, Florida International University, Miami, FL 33199

November 1, 2000

Abstract

In this paper, we present a methodology for finding the n most similar documents across multiple text databases for any given query and for any positive integer n. This methodology consists of two steps. First, the contents of databases are indicated approximately by database representatives. Databases are ranked using their representatives in a certain order with respect to the given query. We provide a necessary and sufficient condition to rank the databases optimally. In order to satisfy this necessary and sufficient condition, we provide three estimation methods. One estimation method is intended for short queries; the other two are for all queries. Second, we provide an algorithm, *OptDocRetrv*, to retrieve documents from the databases according to their rank and in a particular way. We show that if the databases containing the n most similar documents for a given query are ranked ahead of other databases, our methodology will guarantee the retrieval of the n most similar documents for the query. We also show that all our estimation methods together with our retrieval algorithm guarantee optimal retrieval for single-term queries. Experimental results are given to compare the retrieval effectiveness of the three estimation methods which are used in conjunction with *OptDocRetrv*.

When the number of databases is large, it may not be practical to compare the query against all database representatives. In that situation, database representatives are pre-arranged in a hierarchy and a best-search algorithm is presented to search the hierarchy. It is shown that the effectiveness of the best-search algorithm is the same as that of evaluating the user query against all database representatives.
Keywords: Distributed information retrieval, resource discovery, database selection.

1 Introduction

The Internet has become a vast information resource in recent years. To help ordinary users find desired data in this environment, many *search engines* have been created. Each search engine has a *text database*

*Portions of this paper appeared in the following papers [36, 49, 50].

that is defined by the set of documents that can be searched by the search engine. Usually, an inverted file index for all documents in the database is created and stored in the search engine. For each *term* which can represent a significant word or a combination of several (usually adjacent) significant words, this index can identify the documents that contain the term quickly.

Frequently, the information needed by a user is stored in the databases of multiple search engines. As an example, consider the case when a user wants to find research papers in some subject area. It is likely that the desired papers are scattered in a number of publishers' and/or universities' databases. Substantial effort would be needed for the user to search each database and identify useful papers from the retrieved papers. A solution to this problem is to implement a *metasearch engine* on top of many local search engines. A metasearch engine is a system that supports unified access to multiple existing search engines. It does not maintain its own index on documents. However, a sophisticated metasearch engine may maintain information about the contents of its underlying search engines to provide better service. When a metasearch engine receives a user query, it first passes the query to the appropriate local search engines, and then collects (sometimes, reorganizes) the results from its local search engines. With such a metasearch engine, only one query is needed from the above user to invoke multiple search engines.

Building a metasearch engine is also an effective way to increase the search coverage of the Web. As more and more data are put on the Web at faster paces, the coverage of the Web by individual search engines has been steadily decreasing. A recent study [31] indicates that the number of publicly indexable web pages has increased from 320 million in Dec. 1997 to 800 million in Feb. 1999 while the coverage of the Web by the largest search engine has decreased from 30% to no more than 16%. By combining the coverages of multiple search engines, a metasearch engine can have a much larger coverage of the Web than that of any individual underlying search engines.

A closer examination of the metasearch approach reveals the following problems.

1. If the number of local search engines in a metasearch engine is large, then it is likely that for a given query, only a small percentage of all search engines may contain sufficiently useful documents to the query. In order to avoid or reduce the possibility of invoking useless search engines for a query, we should first identify those search engines that are most likely to provide useful results to the query and then pass the query to only the identified search engines. Examples of systems that employ this approach include gGlOSS [17], SavvySearch [21], D-WISE [53], CORI Net [6], and ProFusion [9, 14]. The problem of identifying potentially useful databases to search is known as the *database selection problem*.

2. If a user only wants the n most similar documents across all local databases, for some positive integer n, then the n documents to be retrieved from the identified databases need to be carefully specified and retrieved. This is the *document selection problem*.

In this paper, we study both the database selection problem and the document selection problem. First, we identify a necessary and sufficient condition for a collection of databases to be ranked optimally with respect to a given query (essentially, the databases containing one or more of the n most similar documents are ranked higher than other databases, but a precise definition will be given later). This is a key step in the database selection problem. Second, we provide estimation algorithms to attempt to rank the databases optimally for each query. One of the estimation algorithms has an exponential complexity in the number of distinct terms in the query. As a result, it can be used for short queries only. The other estimation algorithms have linear complexities and can be used for all queries. Third, we provide an algorithm to retrieve documents from ranked databases. We show that if the databases are ranked optimally, then the n most similar documents will be retrieved by our algorithm. We also show that all our estimation methods when used in conjunction with our algorithm to retrieve documents from local databases will be able to optimally retrieve the n most similar documents for single-term queries. In the Internet, it is known that single-term queries are submitted frequently [1, 18, 23]. Fourth, experimental results are shown to compare the performance of our algorithms. On the one hand, significant improvements of all of our algorithms over the high-correlation method in a well-known system gGlOSS [17, 18] are provided. On the other hand, for short queries with no more than 6 words per query, one of our algorithms retrieves, on the average, from 98% to 99% of the n most similar documents while searching additional 7.5% to 13.7% databases and transmitting additional 10.9% to 24.4% documents, when n varies from 5, 10, 20 to 30. This shows that our methodology can essentially achieve the same retrieval effectiveness as if all data were stored in one database. For other queries, each containing 7 or more words, one of our algorithms retrieves, on the average, from 90% to 98% of the n most similar documents while searching additional 6.8% to 12% databases and transmitting additional 38% to 53.5% documents, when n varies from 5, 10, 20 to 30. Fifth, we provide an algorithm for the database selection problem which is suitable for numerous databases. Specifically, representatives which indicate approximate contents of databases are arranged in a hierarchy and a search algorithm is given to compare the given query against the hierarchy. It is shown that not all database representatives need to be compared against the query, but the effectiveness of the search algorithm is the same as that of comparing the user query against all database representatives.

The rest of the paper is organized as follows. In Section 2, the database selection problem and the document selection problem are discussed in more detail in the context of this paper. A summary of related work is given in Section 3. In Section 4, our methodology is presented and a necessary and sufficient condition to rank databases optimally is provided. In Section 5, estimation methods to satisfy the necessary and sufficient condition are described. In Section 6, an algorithm to retrieve documents from local databases and its desirable properties are given. In Section 7, experimental results are reported. In Section 8, an algorithm to compare a user query against a hierarchy of database representatives is given. We conclude the paper in Section 9.

2 The Database Selection Problem and the Collection Fusion Problem

We assume that the vector space model [38] is used to represent documents and queries. In this model, each document (or each query) is simply a set of words. It is transformed into a vector of *terms* with weights [38]. The weight of a term usually depends on the number of occurrences of the term in the document (relative to the total number of occurrences of all terms in the document) [38, 52]. It may also depend on the number of documents having the term relative to the total number of documents in the database (i.e, the *inverse document frequency* weight [43, 39]). A query is similarly transformed into a vector of terms with weights. The similarity between a query $q = (q_1, ..., q_k)$ and a document $d = (d_1, ..., d_k)$ can be measured by $\sum_{i=1}^{k} q_i * d_i$ which is the dot product of the two vectors. Often, the dot product is divided by the product of the *norms* of the two vectors, where the norm of a vector $(x_1, x_2, ..., x_k)$ is $\sqrt{\sum_{i=1}^{k} x_i^2}$. This is to normalize the similarity between 0 and 1. The similarity function with such a normalization is known as the *Cosine* function [38, 52]. Other similarity functions, see for example [42], are also possible.

Given a global similarity function $sim()$, and a query q, the n most similar documents are those documents which have the n largest $sim(q, d)$ values, where d is a document. The database selection problem is to identify, for the query, the databases that contain the n most similar documents. In order that appropriate databases can be identified, the metasearch engine maintains a *representative* for each database. The representative of a database indicates approximately the contents of the database. When a query q is submitted, q is compared against the database representatives. (The case where there are too many database representatives to be compared against the user query will be discussed in Section 8.) Based on the comparisons between q and the database representatives, the databases are ranked in the order $D_1, D_2, ..., D_m$ such that D_i is searched before D_{i+1}, $1 \leq i \leq m - 1$, where m is the number of databases. In Section 4, we will give a necessary and sufficient condition for databases to be optimally ranked, i.e., it is sufficient to search the first k databases, for some k less than or equal to m, in order to retrieve the n most similar documents and each of the first k databases contains at least one of the desired documents. The necessary and sufficient condition is simple: rank database D_i ahead of database D_j if the global similarity of the most similar document in D_i is greater than that of the most similar document in D_j. Based on this result, we provide algorithms to estimate the global similarity of the most similar document in each database.

Consider the problem of retrieving documents from local databases. In the Internet environment, local search engines are likely to be autonomous and may rank locally retrieved documents using different local similarity functions. Since local similarities across multiple databases may be incomparable, the n most similar documents across all local databases to a given query are determined by similarities computed using a *global similarity function*. It is likely that local similarity functions are different from the global similarity function. Even if the same similarity function is used, the weight of a term which may depend on the number of documents having the term may change from the global database (i.e., the imaginary database unioned from all local databases) to a local database. As a result, the similarity of a document with respect to a local database may differ from the similarity of the same document with respect to the global database. This situation causes a problem if our goal is to retrieve documents whose global similarities are greater than a certain threshold. Two solutions were proposed in [35] to tackle this problem. The first solution is to transform the threshold T_0 for the global database (i.e., the global threshold) to a local threshold T_i for each local database D_i so that all documents in D_i having global similarities $\geq T_0$ are contained in the set of documents in D_i having local similarities $\geq T_i$. This ensures that the former set of documents is retrieved. Furthermore, the latter set is the smallest possible, indicating the tightness of the local threshold T_i. The second solution is that the metasearch engine modifies the user query before submitting it to a local search engine such that the local similarity of a document in that local database with the modified query is the same as the global similarity of that document with the original user query. In either solution, the actual global similarities of documents from local databases can be determined. Assuming that the databases have been ranked in the order $D_1, D_2, ..., D_m$, we need to determine an integer k such that the first k databases are searched but the remaining databases are not searched. Furthermore, when a database is searched, we want to minimize the number of documents to be transmitted from the database to a software component called *result merger* (which is part of the metasearch engine). This software component merges the transmitted documents from the searched databases to form a list of n documents to be returned to the user. One simple strategy is to transmit n documents having the largest global similarities from each of the first k databases to the result merger which merges them into a list in descending order of the global similarities and then take the top n documents. This ensures that if the databases are optimally chosen (i.e., databases containing one or more of the n most similar documents are ranked ahead of other databases), then all the n most similar documents will be retrieved. However, using this method, a total number of $k * n$ documents will be transmitted. We provide an algorithm to determine the cut-off k and to determine the documents to transmit from each of these k databases so that usually fewer than $k * n$ documents are transmitted.

3 Related Work

Non-trivial existing solutions to the database selection problem can be roughly classified into the following three categories. *Qualitative Approaches* predict the quality of each database with respect to a given query based on certain ranking formula. Often, the quality measures or ranking scores are difficult to understand. Some of these approaches use very rough database representatives such as a few words or a few paragraphs ((e.g., WAIS [24], ALIWEB [27], NetSerf [7] and Search Broker [34]) while others use very detailed representatives in the sense that one or more pieces of statistical information are kept for each term in the database (e.g., gGlOSS [17], CORI Net [6] and its later extension [46], and D-WISE [53]). *Quantitative Approaches* predict the usefulness of each database based on measures that are much easier to understand than the measures used in qualitative approaches. In other words, the measures used in quantitative approaches reflect the usefulness of a database with respect to a given query more directly and explicitly in comparison to qualitative approaches. Usually, one or more pieces of statistical information need to be kept in order to enable the database usefulness prediction in quantitative approaches. One measure used by quanlitative approaches is the "the number of documents in a database whose similarity with a given query is above a threshold". Using this measure, the approaches in [51, 32] represent each document d as a binary vector such that a 0 or 1 at the ith position indicates the absence or presence of term t_i in d. A substantial amount of information will be lost when documents are represented by binary vectors. The estimation methods given by us are very different from those given by other researchers, including those in [18]. In this paper, one of our results will show that all of our estimation algorithms when used in conjunction with our algorithm to retrieve documents from local databases yield higher retrieval effectiveness than the high-correlation method in gGlOSS in a distributed database environment using documents and queries that are used to evaluate gGlOSS. *Learning-based Approaches* make use of past retrieval experiences with respect to a component search engine to predict the usefulness of the search engine. The retrieval experiences could be obtained through the use of training queries before the database selection algorithm is put to use (e.g., MRDD [45]) and/or through the real user queries while database selection is in active use (e.g., SavvySearch [8] and ProFusion [9]). The obtained experiences against a database will be saved as the representative of the database. The theoretical approaches taken by [2, 12] are very different from ours. No experimental results are reported in [12]. Recent experimental results reported in [3] show that if the number of documents retrieved is larger than or comparable to the number of databases, then good retrieval effectiveness can be achieved; otherwise, there is substantial deterioration in performance. In this paper, we show good experimental results in both situations, although our data collection is relatively small. It was shown in [46] that excellent retrieval effectiveness can be achieved using *query expansion*. However, the queries used in [46] are much longer than typical Internet queries. In addition, the proposed approach depends on the existence of a training collection which has similar coverage of subject matters to the collections of documents to be searched. Whether such a training collection can be constructed in the Internet environment which contains very heterogeneous documents remains to be seen.

Our necessary and sufficient condition to rank databases optimally is different from others. One of the statistics we use in each database representative is the maximum (normalized) weight of each term. We believe that this statistics, which is absent in others' works in this area, is critical in accurate determination of which databases to search. Many previous works utilize the average (normalized) weight. In our opinion, it is not possible to estimate accurately quantities such as the similarity of the most similar document and the expected number of documents which are most similar to a query in a database without using the maximum (normalized) weight of each term. The database representatives required by one of our estimation algorithms use the same amount of storage space as those required by gGlOSS.

Non-trivial existing solutions to the *document selection problem* can also be classified into four categories. The *user determination approach* lets the user determine how many documents should be retrieved from each local search engine. MetaCrawler and SavvySearch use this approach. In the approach used in [46], for each query, the 10 best databases are selected; then 30 documents are retrieved from each selected database to form a ranked list of 300 documents; finally a desired number of documents are selected from this list. More sophisticated *weighted allocation approaches* retrieve proportionally more documents from local search engines whose databases have higher ranking scores. CORI Net, ProFusion and D-WISE employ such approaches. *Learning-based approaches* determine the number of documents to retrieve from a local database based on past retrieval experiences with the database. Several learning-based algorithms in [44, 45] make use of a set of *training queries*. Weighted allocation and learning-based approaches are heuristic in nature and they do not guarantee that all globally most similar documents will be retrieved from each local search engine. The *guaranteed retrieval approach* aims at guaranteeing such a property. The algorithm in [19] while guaranteeing that all potentially useful documents are retrieved may unnecessarily retrieve many non-similar documents. The approach in [35] is also a guaranteed retrieval approach but has a second goal of minimizing the number of non-similar documents retrieved. The guaranteed retrieval approach has important applications in medical and legal domains as doctors or lawyers often want to find all or nearly all past cases most similar to their present cases.

The document retrieval algorithm we propose in this paper has the property that when it is used together with any of our database selection methods, all the n most similar documents for any single-term query will be retrieved. Experimental results will show that with our approach between 98% to 99% of the n most similar documents are retrieved for queries, each containing up to 6 words, while the additional databases to be searched and the additional documents to be transmitted vary from 7.5% to 13.7% and from 11.2% to 24.4%, respectively. Since Internet queries are typically very short (an average query has about 2.2 terms [25]), the vast majority of Internet queries will have no more than 6 words per query. For queries with 7 or more words, experimental results will show that with our approach between 90% to 98% of the n most similar documents are retrieved, while additional 6.8% to 12% databases are searched and additional 38% to 53.5% documents are transmitted.

4 A Methodology for Database Selection and Collection Fusion

The methodology that we propose to retrieve the n most similar documents across multiple databases for a given query consists of the following two steps:

1. Rank the databases so that a database with a higher rank will be searched before a database with a lower rank.

2. Search the databases according to their rank in a certain manner to retrieve the n documents.

We first present a necessary and sufficient condition to rank databases optimally. Then, in Section 6, we provide an algorithm for step 2, i.e., to retrieve documents from the ranked databases. In this paper, we assume that all local databases in a metasearch engine are distinct. Identical databases (search engines) may exist due to the creation of mirror sites.

4.1 A Necessary and Sufficient Condition for a Set of Databases to Be Ranked Optimally

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

Intuitively, the ordering is optimal because whenever the n most similar documents to the query are desired, it is sufficient to examine the first k databases. Note that the ordering of the databases depends on the query q. For ease of presentation, we shall assume that all similarities of the documents with the query are distinct so that the set of the n most similar documents to the query is unique.

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

Example 1 Consider the databases D_1, D_2 and D_3. Suppose the global similarities of the most similar documents to a given query in the databases D_1, D_2 and D_3 are 0.5, 0.75 and 0.6, respectively. Then, the databases should be ranked in the order $[D_2, D_3, D_1]$ for the query. ■

The above proposition says that in order to rank databases optimally, it is sufficient to examine only the most similar document in each database: find their highest global similarities and then order them in descending order. This proposition cannot be used as is, however, because we cannot afford to search each database and obtain the global similarity of its most similar document. Instead, for each database, we shall apply one of the three methods (to be described in the next section) to estimate the required similarity.

Proof of Proposition 1:
Sufficiency: Suppose $msim(q, D_1) > msim(q, D_2) > ... > msim(q, D_m)$. We need to show that $[D_1, D_2, ..., D_m]$ is an optimal order. We establish by induction that for any given n, there exists a k such that $D_1, D_2, ..., D_k$ are the only databases containing the n most similar documents with each of them containing at least one such document.

For $n = 1$, since $msim(q, D_1) > msim(q, D_2) > ... > msim(q, D_m)$, database D_1 contains the overall most similar document. Thus, $k = 1$ is found for $n = 1$.

For $n = i$, suppose that $D_1, D_2, ..., D_s$ contain the i most similar documents with each of them containing at least one of the i most similar documents. When $n = i + 1$, consider the $(i + 1)$-th most similar document. It appears either in one of the databases $D_1, ..., D_s$ or in one of the remaining databases. In the former case, $D_1, ..., D_s$ contain all of the $i + 1$ most similar databases and $k = s$. In the latter case, the $(i + 1)$-th most similar document must appear in D_{s+1} because $msim(q, D_{s+1}) > msim(q, D_{s+2}) > ... > msim(q, D_m)$. Thus, for the latter case, $k = s + 1$.

Necessity: Suppose the optimal rank order of the databases is $[D_1, D_2, ..., D_m]$. We now show that $msim(q, D_1) > msim(q, D_2) > ... > msim(q, D_m)$. When $n = 1$, the most similar document is in database D_1. Thus, $msim(q, D_1) > msim(q, D_j)$, $2 \leq j \leq m$. Let n be increased to i_1 so that the most similar $i_1 - 1$ documents appear in database D_1 and the i_1-th most similar document appears in another database D. This i_1-th most similar document must be the most similar document in D and because $D_1, D_2, ..., D_m$ are optimally ranked, the database D must be D_2. Thus, $msim(q, D_2) > msim(q, D_j)$, $3 \leq j \leq m$. Let n be increased from i_1 to i_2 so that the i_1-th to $(i_2 - 1)$-th most similar documents appear in database D_1 or database D_2 and the i_2-th most similar document appears in another database D$'$. Again by the optimal rank ordering of $[D_1, D_2, ..., D_m]$, database D$'$ must be D_3 and hence, $msim(q, D_3) > msim(q, D_j)$, $4 \leq j \leq m$. By repeatedly increasing n in the manner described above, we obtain $msim(q, D_4) > msim(q, D_j)$, $5 \leq j \leq m$, ..., $msim(q, D_{m-1}) > msim(q, D_m)$. By combining all these derived inequalities, we obtain $msim(q, D_1) > msim(q, D_2) > ... > msim(q, D_m)$. ■

We have the following observations about Proposition 1.

Observation 1: The necessary and sufficient condition is independent of the similarity function. In fact, if there is a relevance function which assigns degrees of relevance to documents, the same result will also be applicable. The condition can be applied to all types of databases, including image, audio and video databases.

Observation 2: The necessary and sufficient condition is rather surprising, because one normally ranks a database, say D_i, containing many of the n most similar documents, ahead of another database, say D_j containing fewer most similar documents. But if D_j contains the document with the largest similarity among all databases, then our result will rank database D_j highest. This is to ensure that if the user wants the document with the largest similarity, D_j will be searched before any other databases. Note that the condition $msim(q, D_1) > msim(q, D_2) > ... > msim(q, D_m)$ is independent of n. On the other hand, if the user wants only most but not necessarily all of the n most similar documents, then the above database D_i may be ranked higher than database D_j.

Observation 3: If not all similarities of the documents with the query are distinct, Proposition 1 remains essentially true (need to change all $>$ to $\geq$) but the optimal order may no longer be unique. In this case, if $msim(q, D_1) \geq msim(q, D_2) \geq ... \geq msim(q, D_m)$, then for every positive integer n, there exists a k such that $D_1, D_2, ..., D_k$ contain one set of n documents that have the highest similarities with q among all documents and each D_i, $1 \leq i \leq k$, contains at least one document in the set. It is possible that a document not in the set has the same similarity as some documents in the set.

5 Estimation Methods

In this section, we present three methods to estimate the similarity of the most similar document to a query q in any given database D. To enable the estimation, we need a representative for database D. The representative indicates an approximate content of the database. In addition, a representative for the "global database" which logically contains all documents but does not exist physically is created. The representative for the global database contains for each term, the global inverse document frequency weight of the term. The representative for each individual database varies from one estimation method to another and will be described with its associated estimation method. The first method, the *subrange-combined-term method*, is given in [36, 33]. Although it is reasonably accurate, its time complexity is exponential in the number of terms of the query. As a result, the method is applicable to short queries only in practice. As pointed out earlier, most Internet queries belong to this type. For queries of arbitrary lengths, we provide two new methods, *fast-similarity method* and *fast-combined-term method*, to estimate the similarity of the most similar document. The complexities of these algorithms are linear in the number of query terms. Thus, they can be executed efficiently.

5.1 Subrange-Combined-Term Method

We first review the *subrange-based method* [36] for estimating the usefulness of a database with respect to a given query. We then sketch the *subrange-combined-term method* [33] which is an extension of the *subrange-based method* by incorporating one type of term dependencies between adjacent terms.

5.1.1 The Subrange-Based Estimation Method

Suppose database D has m distinct terms. Each document d in this database can be represented as a vector $d = (d_1, \ldots, d_m)$, where d_i is the weight (or significance) of the ith term due to its term frequency (i.e. the number of occurrences) in the document, $1 \le i \le m$. Each query is also represented by a vector. Consider query $q = (u_1, u_2, ..., u_m)$, where u_j is the weight of t_j in the query, $1 \le j \le m$. The weight u_j is due to the term frequency of the term as well as its document frequency (the number of documents having the term). Usually, the higher the document frequency, the smaller the weight is assigned. In this paper, we use the product of the term frequency weight and the inverse document frequency weight as given in [45]. The global similarity between q and d can be defined as the dot product of their respective vectors, namely $sim(q, d) = u_1 * d_1 + \cdots + u_m * d_m$. Similarities are often normalized between 0 and 1. One common normalized similarity function is the $Cosine$ function [38]. This is easily implemented by dividing each d_i by the norm of the document and each u_j by the norm of the query. Thus, it is sufficient to consider the dot product similarity function. For ease of reading, we sometimes use unnormalized term weights in our discussion.

The representative of a database D with m distinct terms can be given by m tuples $\{(p_i, w_i, mnw_i, sd_i) \mid i = 1, \ldots, m\}$, where p_i is the probability that term t_i appears in a document in D, w_i is the average of the (normalized) non-zero weights of t_i in the set of documents containing t_i, mnw_i is the maximum normalized weight of t_i and sd_i is the standard deviation of the non-zero normalized weights of t_i. For example, if the normalized weights of t_i on 10 documents are (0, 0, 0, 0, 0.2, 0.2, 0.4, 0.4, 0.6, 0.6), then $p_i = 0.6$, as 6 out of the 10 documents have the term, $w_i = 0.4$, as the average is over all documents having the term, mnw_i

$= 0.6$, as the maximum normalized weight is 0.6 and $sd_i = 0.16$. For a given query $q = (u_1, u_2, ..., u_m)$, the database representative is used to estimate the similarity of the most similar document in D. Without loss of generality, we assume that only the first r u_i's are non-zero, $0 < r \leq m$. Therefore, q becomes $(u_1, u_2, ..., u_r)$ and $sim(q, d)$ becomes $u_1 * d_1 + \cdots + u_r * d_r$.

The distribution of the similarities of the query with the documents due to term t_i can be represented by the following polynomial [36]:

$$p_{i1} * X^{wm_{i1} * u_i} + p_{i2} * X^{wm_{i2} * u_i} + \cdots + p_{il} * X^{wm_{il} * u_i} + (1 - p_i) \tag{1}$$

where $p_{i1} + p_{i2} + ... + p_{il} = p_i$ is the probability that a document has the term. The range of positive weights of t_i is partitioned into l subranges of weights such that the probability that a weight of t_i occurs in the j-th subrange is p_{ij} and the median of the j-th subrange is wm_{ij}, $j = 1, ..., l$. The variable X is utilized such that its coefficient represents a probability and its exponent represents an increase to the similarity value. In the above example, we can have three subranges of weights whose medians are 0.2, 0.4 and 0.6, respectively, and each of their associated probabilities is 0.2. The resulting polynomial is:

$$0.2X^{0.6*u_i} + 0.2X^{0.4*u_i} + 0.2X^{0.2*u_i} + 0.4 \tag{2}$$

This polynomial says that 20% of the documents have similarities $0.6 * u_i$, 20% of the documents have similarities $0.4 * u_i$, 20% of the documents have similarities of $0.2 * u_i$ and 40% of the documents have similarities $= 0$ due to the term t_i alone. Under the assumption that non-zero normalized weights of each term satisfy the normal distribution, the standard deviation sd_i together with p_i and w_i permit the generation of the median wm_{ij} and its associated probability p_{ij} so that they need not be stored. For each term in the user query, there is a polynomial as indicated above. When all these polynomials are multiplied together and terms with the same exponents of X are merged, we obtain

$$a_1 * X^{b_1} + a_2 * X^{b_2} + \cdots + a_c * X^{b_c} \tag{3}$$

We assume that the terms in (3) are listed in descending order of the exponents, i.e. $b_1 > b_2 > ... > b_c$. It was shown in [36] that if the terms are independent, then the coefficient of X^s in function (3) is the probability that a document in database D has similarity s with q. Thus, $N * a_i$ is the expected number of documents that have similarity b_i with query q, where N is the number of documents in database D. As an example, suppose a query has two terms, each with a weight of 1 such that the polynomial of the first term is given by (2) with $u_i = 1$ and that of the second term is given by

$$0.3X^{0.2} + 0.1X^{0.1} + 0.6 \tag{4}$$

Then, the multiplication of the two polynomials in (2) and (4) yields

$$0.06X^{0.8} + 0.02X^{0.7} + 0.18X^{0.6} + 0.02X^{0.5} + ... \tag{5}$$

From this polynomial, we interpret that $0.06N$ is the expected number of documents in the database having similarity 0.8 with the query, $0.02N$ documents in the database have similarity 0.7 with the query, etc.

To estimate the similarity of the most similar document in a database, we simply scan the terms in expression (3) in descending order of exponents until $\sum_{i=1}^{s} a_i * N$ is approximately one for some s. The exponent, b_s, is the estimated similarity of the most similar document in the database as the expected number of documents in the database with similarity greater than or equal to b_s is approximately 1. More technical information about the estimation process can be found in [36].

The complexity of the estimation method is exponential in the number of terms of the query for the following reason. Suppose for each term in the query, there are p coefficients in the polynomial to represent the term. Multiplying two of these polynomials together takes $O(p^2)$ time. Thus, the multiplication of t such polynomials takes $O(p^t)$ time, where t is the number of terms in the query. Fast Fourier Transform is unlikely to speed up the process in practice.

The space requirement for this method is that for each term, four quantities are kept (the average normalized non-zero weight, the maximum normalized weight, the probability, the standard deviation of the normalized weights). With the average normalized non-zero weight and the standard deviation, it is possible to approximate the distribution of the normalized weights of the term, as required in the polynomial for the term. It was pointed out [36] that the space requirement for this type of representatives is about 1.5% to 3.0% of the space required to store the documents in the database.

5.1.2 The Subrange-Combined-Term Estimation Method

The assumption that terms are independently distributed in the previous solution is not entirely realistic. For example, the two terms "computer" and "algorithm" may appear together more frequently in documents in a database than that expected if the two terms were independently distributed in the database. The *subrange-combined-term method* is designed to remedy the term independence assumption through the incorporation of one type of dependencies between two adjacent terms.

Consider the distributions of terms t_i and t_j in a database of documents. Within the set of documents having both terms, there is a document having the largest sum of the normalized term weight of t_i and the normalized term weight of t_j. Let the largest sum be called the *maximum normalized weight* of the combined term and be denoted by mnw_{ij}. If terms t_i and t_j are combined into a single term, then the probability that a document in the database has the maximum normalized weight of the combined term, mnw_{ij}, can be assumed to be $1/N$, where N is the number of documents in the database, because it is unlikely that another document in the database has the same maximum normalized weight under the combined term.

The criteria that the two terms t_i and t_j should be combined into a single term t_{ij} are

- the estimated probability that a document in the database has at least the normalized sum of term weights mnw_{ij} for the two terms t_i and t_j under the term independence assumption is very different from $1/N$; and
- the maximum normalized weight of the combined term is higher than the maximum normalized weight of each of the two individual terms.

Since our aim is to estimate the similarities of the most similar documents, the second condition is to ensure that if the combined term is used, it will not lead to smaller similarities. The first condition is implemented by computing the difference in absolute value between $1/N$ and the estimated probability and then comparing to a pre-set threshold. If the difference exceeds the threshold, then the two terms should be combined.

If the two terms are combined, then we apply the *subrange-based method* for the combined term. The rest of the estimation process remains unchanged.

The space requirement for this method is higher than that for the subrange-based method because for each combined-term pair, we also need to keep the four quantities. However, two terms are combined only when they appear in adjacent locations in a previous query, they appear within reasonable proximity condition (say no more than 3 words apart) in a reasonable number of documents, and they satisfy the above two criteria to be combined.

5.2 Fast-Similarity Method

Consider a query q having k terms $t_1, t_2, ..., t_k$ with corresponding weights $q_1, q_2, ..., q_k$. These weights incorporate both the term frequency and the inverse document frequency information and have been normalized between 0 and 1. Let d be a document in a database D having the maximum normalized weight, mnw_i, of one of the query terms, say term t_i. We want to compute the expected similarity of d with the query q, assuming that the other query terms are distributed independently in the set of documents. Let anw_j be the average normalized weight of term t_j in all documents in database D, including documents not having the term. (Note that this average normalized weight is different from the average normalized non-zero weight used in the *subrange-combined-term method*, as the latter includes documents having the term only.) Then, the expected similarity of d with q is $q_i * mnw_i + \sum_{j=1,j\neq i}^{k} q_j * anw_j$. The expected similarity of the most similar document in database D is at least that of d. This argument can be repeated for other documents in database D, each having the maximum normalized term weight of one of the k query terms. Thus, the expected similarity of the most similar document in D $\geq \max_i \{q_i * mnw_i + \sum_{j=1,j\neq i}^{k} q_j * anw_j\}$. We assume that the most similar document has one of the query terms having the maximum normalized weight. Thus, the inequality is replaced by an equality. Note that for this estimation method, we only need two pieces of information, mnw_i and anw_i, for the ith term in the representative.

The computational complexity for this method is linear in the number of terms for each database. First, we compute $\sum_{j=1}^{k} q_j * anw_j$. This takes $O(k)$ time. From this temporary result, we generate $q_i * mnw_i + \sum_{j=1,j\neq i}^{k} q_j * anw_j$, $1 \leq i \leq k$, by subtracting $q_i * anw_i$ and adding $q_i * mnw_i$, for each i between 1 and k. This takes $O(k)$ time, as there are k such expressions. Finally, the maximum of the k values is taken. Thus, altogether $O(k)$ is spent.

5.3 Fast-Combined-Term Method

This estimation method is essentially the same as that of the *fast-similarity method*, except that adjacent query terms may be combined together as in the *subrange-combined-term method*. The criterion for combining two terms is simpler. Two adjacent query terms may be combined together if the *maximum weighted normalized weight* of the combined term (i.e., the two terms t_i and t_j are treated as a single term) is higher than its expected value. The maximum weighted normalized weight of the combined term is defined as follows. Consider a document $d = (d_1, ..., d_i, ..., d_j, ..., d_m)$ containing both terms t_i and t_j. The *weighted normalized weight* of the combined term is $idf_i * d_i + idf_j * d_j$, where idf_i and idf_j are the global inverse document frequency weights of terms t_i and t_j, respectively. The maximum weighted normalized weight of the combined term is the maximum value over all documents. Consider the documents having either the ith term or the jth term with the maximum normalized weight. If these two terms are independent, then the expected similarity between a query having one occurrence of each of the two terms and one such document is estimated to be $ES = \max\{idf_i * mnw_i + idf_j * anw_j, idf_i * anw_i + idf_j * mnw_j\}$.

The subtraction of the latter expression, ES, from the maximum weighted normalized weight is called the *deviation* of the pair of terms (t_i, t_j) from independence. If the deviation is greater than 0, then the pair is a potential pair of combined terms.

Consider a query containing terms t_i, followed by t_j and then t_k. Suppose the deviations are greater than 0 by the term pairs (t_i, t_j) and (t_j, t_k). To decide which term pair to use for the query, we compare the deviation from independence for the term pair (t_i, t_j) with that for the pair (t_j, t_k). The term pair with the larger deviation from independence will be chosen.

6 Retrieval from Local Databases

The methods from Section 5 estimate the similarity of the most similar document in each database. Using the necessary and sufficient condition given in Section 4, the databases are ranked based on the estimated similarities of their most similar documents. Suppose the order of the databases is $[D_1, D_2, ..., D_m]$. We now give an algorithm which determines a cut-off k such that the top k ranked databases will be searched and the remaining databases will not be searched. The algorithm also determines which documents need to be transmitted from the selected databases to the result merger to form a list of documents in descending order of global similarity. This algorithm has the following desirable properties: (1) If databases, each of which contains at least one of the n most similar documents are ranked higher than databases which do not contain any of the n most similar documents, then the algorithm will retrieve all the n desired documents. Furthermore, the number of databases searched is at most one more than the number of databases containing the desired documents. (2) Any of the three estimation methods from Section 5 when used in conjunction with this retrieval algorithm will retrieve all the n desired documents for single term queries.

We now present the algorithm, *OptDocRetrv*, to attempt to retrieve the n most similar documents with respect to a given query from these databases. The basic idea is that we retrieve documents from the databases in the order $[D_1, D_2, \ldots, D_m]$ until n most similar documents contained in the selected databases are obtained. (The obtained n most similar documents in the selected databases may or may not be the n most similar documents from all databases.) Consider the top s databases $D_1, D_2, \ldots, D_s$. From each of these databases we obtain the actual global similarity of its most similar document. Let the minimum of these s similarities be *m-asim*. Next, from these s databases we transmit to the result merger all documents whose actual global similarities are greater than or equal to *m-asim*. If n or more documents have been obtained, these documents are sorted in descending order of similarity and the first n documents are returned to the user. Otherwise, the next database in the given order, namely D_{s+1}, will be considered and its most similar document will be transmitted. The actual similarity of this document is then compared against *m-asim*. The minimum of these two similarities will be used as a new threshold to transmit all documents from these $s+1$ databases whose actual global similarities are greater than or equal to this threshold.

Algorithm OptDocRetrv

1. transmit the most similar document from database D_1 to the result merger and let $asim_1$ be its actual

global similarity;

2. $m\text{-}asim := asim_1$;

3. $j := 1$; /* D_j is the database whose most similar document has similarity equal to the current $m\text{-}asim$. */

4. $i := 2$;

   while (the number of documents obtained so far is less than n) {

   (a) transmit the most similar document from database D_i to the result merger and let $asim_i$ be its actual global similarity;

   (b) if ($m\text{-}asim \geq asim_i$), then {

      i. transmit documents from databases $D_1, \ldots, D_{i-1}$ to the result merger such that each of the retrieved documents has actual global similarity $\geq asim_i$;

      ii. $m\text{-}asim := asim_i$ and $j := i$;

      }

      else { transmit from database D_i to the result merger all documents having actual global similarities $\geq m\text{-}asim$;

      }

   (c) $i := i + 1$;

   }

5. At the result merger, sort all the transmitted documents in descending order of their actual global similarities and return the top n documents to the user.

Note: In Step 4 (b), documents which had been previously transmitted need not be transmitted again. In addition, the number of documents to be retrieved from any single chosen database should be no more than n, as the user is interested in n documents only.

Example 2 Suppose databases are ranked in the order $D_1, D_2, D_3, D_4, \ldots$ based on the *estimated* global similarity of the most similar document in each database. For the purpose of illustration, suppose also that in each of the first four databases, the documents are arranged in descending order of their *actual* global similarities as shown below.

Database D_1: d_1 0.53, d_2 0.48, d_3 0.39, ...

Database D_2: d_{10} 0.47, d_{21} 0.43, d_{52} 0.42, ...

Database D_3: d_{23} 0.54, d_{42} 0.49, d_{62} 0.38, ...

Database D_4: d_{33} 0.40, ...

Suppose the user wants to retrieve 4 most similar documents. From databases D_1 and D_2, the result merger receives documents d_1 and d_{10} with similarities 0.53 and 0.47, respectively. The minimum of these two similarities, m-asim, is 0.47. From database D_1, documents with similarities $\geq$ m-asim are retrieved. This brings in document d_2. Since less than 4 documents are retrieved, database D_3 will be accessed. The retrieval of document d_{23} with similarity 0.54 results in m-asim remaining unchanged. From database D_3, documents with similarities $\geq$ m-asim are retrieved. After document d_{42} is retrieved, the 5 documents d_1, d_2, d_{10}, d_{23} and d_{42} are sorted in descending order of similarity and the top 4 documents are returned to the user. Note that in this example, although the databases are not ranked in descending order of the actual global similarities of the most similar documents, the $n = 4$ most similar documents are retrieved, as databases D_1, D_2 and D_3 which contain the 4 most similar documents, are ranked ahead of other databases. ■

This algorithm has the following properties.

Proposition 2 For a given query, let S be the set of the n most similar documents. Suppose S is unique (note that in general S may not be unique due to identical similarities of different documents) and databases $D_1, D_2, \ldots, D_k$ collectively contain the documents in S and each of these k databases contains at least one of the n documents in S. If a database selection method ranks the databases $D_1, D_2, \ldots, D_k$ ahead of other databases, then the n most similar documents to the query will be retrieved by algorithm *OptDocRetrv* and furthermore, at most $k + 1$ databases will be examined.

Proof: Let $m\text{-}asim = \min\{asim_1, ..., asim_k\}$, where $asim_i$ is the actual global similarity of the most similar document in database D_i, $i = 1, ..., k$. Let the document having this similarity be from database D_j, i.e., $asim_j = m\text{-}asim$, $1 \leq j \leq k$. Let the actual global similarity of the n-th most similar document be *minsim*. Then, $m\text{-}asim \geq minsim$. Based on algorithm *OptDocRetrv*, when database D_j is examined, all documents from databases $D_1, D_2, \ldots, D_{j-1}$ whose actual global similarities are greater than or equal to *m-asim* will be retrieved. Clearly, these documents are in S. When databases $D_{j+1}, \ldots, D_k$ are examined, documents from these databases having similarities $\geq$ *m-asim* will be retrieved. Again, these documents are in S. Clearly, if $m\text{-}asim = minsim$, then all documents in S would have been retrieved based on the definition of *minsim*. If $minsim < m\text{-}asim$, then consider the next database to be examined, say D_{k+1}. Let $asim_{k+1}$ be the actual global similarity of the most similar document in D_{k+1}. Notice that $asim_{k+1} < minsim$ since the n most similar documents are in databases $D_1, \ldots, D_k$ and S is unique. When D_{k+1} is examined, all documents in $D_1, D_2, \ldots, D_k$ having actual global similarity greater than $asim_{k+1}$ will be retrieved. This includes all documents in S. By step 5 of algorithm *OptDocRetrv*, S will be returned. From the above discussion, the top k or $k + 1$ databases will be examined depending on whether $m\text{-}asim = minsim$ is true. ■

Note that if the set of documents S in Proposition 2 is not unique, then the algorithm *OptDocRetrv* guarantees only that the documents in one set of the n most similar documents be retrieved if the databases containing this set of documents are ranked higher than other databases.

Proposition 2 says that for any database selection method, if it ranks the databases correctly with respect to a given query and if the ranking is used in such a way as in algorithm *OptDocRetrv*, then all the n most similar documents with respect to the query can be retrieved. Even if the first k databases are not ranked optimally (in descending order of the similarities of the most similar documents) but they are ahead of the other databases which do not contain any of the n most similar documents, then all the desired documents will be retrieved. Also, according to Proposition 2, if databases $D_1, D_2, ..., D_k$ which contain the n most similar documents are ranked ahead of other databases, the algorithm *OptDocRetrv* examines at most $k + 1$ databases. Thus, the minimum number of databases plus at most one additional database will be examined by the algorithm. Our experimental results will show that on the average no more than 13.7% additional (beyond the ideal number of databases containing the n most similar documents) databases will be examined.

Proposition 3 For each single-term query, if the set of the n most similar documents S is unique, then all documents in S will be retrieved correctly by the *subrange-combined-term method*, the *fast-similarity method* and the *fast-combined-term method* when used in conjunction with the document retrieval strategy given in algorithm *OptDocRetrv*.

Proof: For each of the three estimation methods, there is a maximum normalized weight associated with each term in each database. For a query containing a single term, say t, the maximum normalized weight of the term for database D is precisely the actual global similarity of the most similar document in database D with respect to the query. (For a single term query, the weight of the term in the query after normalization is 1, irrespective of the inverse document weight of the term.) As a result, databases will be ranked in descending order of the similarity of the most similar document in each database. If databases $D_1, \ldots, D_k$ contain the n most similar documents, then the maximum actual global similarity of any document in any other database will be smaller than the similarity of the most similar document within any of the databases $D_1, \ldots, D_k$. Thus, databases $D_1, D_2, \ldots, D_k$ will be ranked higher than other databases. By Proposition 2, the n most similar documents will be retrieved. ■

Several observations can be made about the above results.

Observation 1: An important assumption implicit in the algorithm *OptDocRetrv* is that the most similar document within each database with respect to the global similarity can be retrieved. There are a number of ways to implement this. One way is to retrieve a number of documents from the chosen database using its local similarity function (see [35]) and then re-compute the actual global similarities of these documents to determine the most similar document for this database. These documents will be saved in cache to be used in later steps of the algorithm. Another way is to modify the query so that the local similarity of the modified query is the same as the actual global similarity of the original query. The following example illustrates this situation.

Example 3 Let the original query $q = (q_1, q_2)$ and a document $d = (d_1, d_2)$. Let the global similarity function $g(q,d) = a_1 * q_1 * d_1 + a_2 * q_2 * d_2$, where a_1 and a_2 are two parameter values used by the *global database*. Let the local similarity function $l(q,d) = b_1 * q_1 * d_1 + b_2 * q_2 * d_2$, where b_1 and b_2 are two parameter values determined by a local database. When the modified query is $q' = (q_1 * a_1/b_1, q_2 * a_2/b_2)$, it can be easily verified that $l(q', d) = g(q, d)$. In information retrieval, it is common to use the *inverse document frequency* weight. The document frequency of a term t in the *global database* (i.e., the number of documents containing the term in the global database) is usually different from its document frequency in a local database. This variation is modeled by the parameters a's and b's in this example. The same approach applies if the standard *Cosine* function is used together with the inverse document frequency weight. ■

Observation 2: In algorithm *OptDocRetrv*, we examine one database at a time. An alternative is as follows. Apply an estimation method such as the *subrange-combined-term method* to identify the databases whose sum of expected numbers of the most similar documents is approximately n. (It is not difficult to see how this can be done using polynomial (3).) Let the number of such databases be m. Then, retrieve from the $(m-1)$ most highly ranked databases by (a) finding from each such database the actual global similarity of the most similarity document; (b) compute the minimum of these $(m-1)$ similarities (let it be *m-asim*); and (c) transmit all documents from these $(m-1)$ databases whose actual global similarities are greater than or equal to *m-asim*. If n or more documents are obtained, then terminate; otherwise, another database is examined. In this way, there will be parallel accesses to the databases and the efficiency of the algorithm may be improved. By either applying the algorithm as is or its modified version as indicated above, a few of the most similar documents will be produced quickly and they can be immediately sent to the user so that the waiting time to see some output can be significantly reduced.

Observation 3: In algorithm *OptDocRetrv*, it is possible that some documents are retrieved from a database, followed by additional retrieval of documents from the same database for a single query. In practice, the n most similar documents from a selected database should be cached such that if documents are to be retrieved in a number of iterations, the documents should be taken out from the cache instead of invoking the search engine of the database.

Observation 4. Algorithm *OptDocRetrv* does not guarantee that the n documents returned are unique if there are overlaps among local databases. However, the algorithm can be easily modified to ensure the retrieval of n distinct documents. The only change that needs to be made is to change the while loop condition in step 4 to "the number of distinct documents obtained so far is less than n".

7 Experimental Results

In this section, we report some experimental results. 15 databases are used in our experiments. These databases are formed from articles posted to 52 different newsgroups in the Internet. These articles were collected at Stanford University [17]. Each newsgroup that contains more than 500 articles forms a separate database. Smaller newsgroups are merged to produce larger databases. The table below shows the number of documents in each database.

database	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#docs	761	1014	705	682	661	622	526	555	629	588	558	526	607	648	564

There are altogether 6,597 queries submitted by real users. Both the data and the queries were used in the gGlOSS project [17]. From these 6,597 queries, we obtain two subsets of queries. The first subset consists of the first 1,000 queries, each having no more than 6 words. They will be referred later as *short queries*. The second subset consists all queries having 7 or more words. There are 363 *long queries*.

In [36], the *subrange-based estimation method* was compared with the *high-correlation method* [17, 18] in terms of the estimation accuracy for individual databases only (i.e., not in terms of retrieving from multiple databases) using the first set of queries identified above (i.e., *short queries*). Here, we compare the performance of the following estimation methods in retrieving the n most similar documents for $n = 5$, 10, 20 and 30 from the 15 databases given above.

1. The *high-correlation method*. Although there are two estimation methods in gGlOSS, our earlier results indicate that the *high-correlation method* is more accurate. Therefore, we choose it for comparison.

The *high-correlation method* does not provide any detail on how a cutoff in database selection is chosen nor which documents are picked from each chosen database. In order to have a fair comparison, all estimation methods will use the *OptDocRetrv* algorithm to retrieve documents from the databases, after the databases have been ranked.

2. The *subrange-combined-term method*. We will use this method for the set of *short queries*, as its exponential complexity is likely to make it not practical for long queries.

3. The *fast-similarity method*.

4. The *fast-combined-term method*.

Both the *fast-similarity method* and the *fast-combined-term method* run in linear time. Thus, they will be used to run for both long and short queries.

The performance measures of an algorithm to search for the n most similar documents in a set of databases are given as follows.

1. The percentage of correctly identified documents, that is, the ratio of the number of documents retrieved among the n most similar documents over n. This percentage is denoted **cor_iden_doc**.

2. 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 n most similar documents. This ratio is denoted **db_effort**. The ratio is usually more than 1.

3. The document transmission effort is the ratio of the number of documents transmitted by the algorithm over n. This ratio is denoted **doc_effort**.

A good retrieval algorithm should aim at achieving *cor_iden_doc* close to (but below) 1, *db_effort* slightly above 1 and *doc_effort* close to but above 1.

In these experiments, the *Cosine* similarity function is utilized. Both the *term frequencies* and the *inverse document frequencies* (**tf-idf**) (the log of the number of documents divided by the number of documents containing a term is the inverse document frequency weight of the term) are utilized [45]. The degree of improvement of method A over method B in terms of retrieving the desired documents is given by (the percentage of desired documents retrieved by method A - the percentage of desired documents retrieved by method B)/(the percentage of desired documents retrieved by method B). In each of the experiments, the numbers of most similar documents for each query are 5, 10, 20 and 30.

A summary of the experimental results is given as follows.

1. From Table 1, it can be seen that for short queries, the best performer is the *fast-combined-term method*. This is followed by the *subrange-combined-term method*, the *fast-similarity method* and then the *high-correlation method*. The same performance ordering of the methods holds true for long queries (see Table 2), except that subrange-combined-term method is not applied to long queries. For both short and long queries, as the number of most similar documents to be retrieved increases, the percentage of correctly identified documents also increases. This is true for each method. Although the fast-combined-term method is better than the subrange-combined-term method in identifying the most

similar documents and in efficiency, the latter method is capable of identifying a set of databases whose sum of the estimated numbers of most similar documents is approximately equal to the desired number of most similar documents. As a result, the latter method can access all these databases in parallel, while the former method may need to determine the desired databases sequentially.

Estimation Method	n	cor_iden_doc	db_effort	doc_effort
High-Correlation	5	69.73%	98.47%	162.4%
High-Correlation	10	75.64%	89.60%	159.2%
High-Correlation	20	82.87%	87.70%	141.1%
High-Correlation	30	87.68%	88.71%	127.4%
Subrange-Combined-Term	5	96.54%	112.8%	125.6%
Subrange-Combined-Term	10	98.01%	109.4%	115.5%
Subrange-Combined-Term	20	99.04%	107.5%	114.0%
Subrange-Combined-Term	30	99.34%	106.6%	112.0%
Fast-Similarity	5	90.67%	112.5%	125.7%
Fast-Similarity	10	93.66%	109.1%	115.9%
Fast-Similarity	20	95.55%	107.6%	112.1%
Fast-Similarity	30	97.10%	106.8%	113.0%
Fast-Combined-Term	5	98.41%	113.7%	124.4%
Fast-Combined-Term	10	99.29%	110.7%	115.2%
Fast-Combined-Term	20	99.58%	108.6%	110.9%
Fast-Combined-Term	30	99.70%	107.5%	111.2%

Table 1: Performances of the Four Methods Using tf-idf for Short Queries

2. It can be seen from Table 1 that for short queries, the *subrange-combined-term method*, the *fast similarity method* and the *fast-combined-term method* achieve close to optimal results. The *fast-combined-term method* retrieves from 98.4% to 99.7% of the most similar documents. This essentially says that it achieves close to the same performance as if all documents were at one site and in one database. Furthermore, the number of databases searched is on the average only at most 13.7% beyond the number of databases containing all desired documents. The number of documents transmitted is on the average at most 24.4% beyond the desired number of documents.

3. Overall, the *fast-similarity method* achieves rather impressive results, given its simplicity and efficiency of computation. It is only slightly worse than the *subrange-combined-term method* for short queries (see Table 1) and is significantly better than the *high-correlation method* for both short and long queries (see Tables 1 to 2). Furthermore, the space required to store the database representatives in order to use the *fast-similarity method* is comparable to that required to store the database representatives to use the *high-correlation method*.

4. If tf term weighting scheme is utilized, very similar results are obtained. Due to the similarity of the two results, only *tf-idf* results are given. For each estimation method, the *tf-idf* term weighting scheme is either comparable or gives better performance than the tf term weighting scheme. The reason is that *idf* weights make the distributions of the similarities of documents skewer. This makes it easier to detect the desired databases.

Estimation Method	n	cor_iden_doc	db_effort	doc_effort
High-Correlation	5	61.44%	99.31%	209.50%
High-Correlation	10	67.64%	87.23%	177.83%
High-Correlation	20	75.82%	83.29%	163.61%
High-Correlation	30	81.82%	86.65%	170.06%
Fast-Similarity	5	75.72%	110.1%	158.8%
Fast-Similarity	10	82.27%	104.9%	151.1%
Fast-Similarity	20	88.31%	104.2%	143.0%
Fast-Similarity	30	91.30%	104.1%	141.5%
Fast-Combined-Term	5	90.22%	112.0%	153.5%
Fast-Combined-Term	10	93.58%	107.4%	148.7%
Fast-Combined-Term	20	97.09%	106.2%	140.4%
Fast-Combined-Term	30	98.54%	106.8%	138.0%

Table 2: Performances of the Three Methods Using tf-idf for Long Queries

5. For quite a few entries in the tables (see for example the first 4 rows in Table 1), we observe that the average number of databases searched can be less than the number of databases containing all the most similar documents. The reason is as follows. When a non-desired database is searched, its most similar document, say d, with similarity, say s, is transmitted to the result merger and documents from previously examined databases having similarities $\geq s$ are also transmitted to the result merger. Since d is not a desired document, its similarity s can be rather low and as a result, it is possible to find as many as n documents in previously examined databases with similarities $\geq s$. This causes the retrieval algorithm to terminate without searching other databases. As pointed out earlier, if all the desired databases are ranked ahead of all other databases, at most 1 additional database will be searched. That not too many databases will be searched is a property of the *OptDocRetrv* algorithm.

8 Searching a General Hierarchy of Database Representatives

In the previous sections, we assumed a two-level architecture for database representatives, where the bottom level contains database representatives for individual databases and the root node is the representative for the "global database". The latter representative contains the global inverse document frequency weight for each term. This architecture is suitable for a moderate number of search engines (or databases — we shall use search engine or database interchangeably since logically each search engine retrieves documents from a logical database), say 100 databases. However, when the number of databases is very large, say thousands or tens of thousands, then there will be efficiency problems. Estimating the similarity of the most similar document for each database could be time consuming for a large number of databases. For this reason, we introduce a general hierarchy of database representatives and a search algorithm for such a hierarchy so that the number of estimations can be significantly reduced.

We first define this hierarchy of database representatives. As in the two-level architecture, the lowest level contains all representatives of individual databases. Individual databases can be logically grouped into *superdatabases*. For example, if superdatabase S_1 contains databases D_1, D_2 and D_3, then all documents in these databases are logically contained in S_1. Physically, superdatabases do not exist. The next level of the

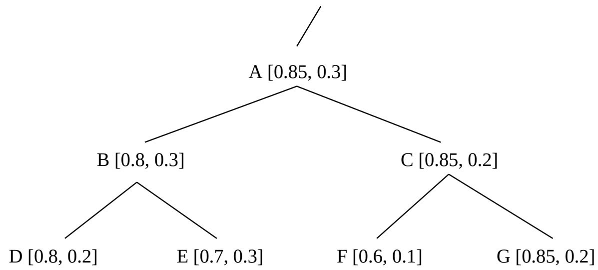

Figure 1: Illustrating the Computation of Super-Representative for One Term

hierarchy contains representatives of superdatabases formed from local databases directly. The representative of a superdatabase will be called a *super-representative*. Each super-representative is constructed from the representatives which are one level below them and is kept physically. The super-representative, RS, which is constructed from a set, R, of representatives, consists of two quantities for each term which appears in any of the representatives in R. The two quantities are the *maximum normalized weight* and the *maximum averaged normalized weight* of the term. Let the maximum normalized weights of term t_i in the j-th representative in R be mnw_{ij}. Then, the maximum normalized weight of term t_i in the super-representative RS is $\max_{all\ j\ in\ R}\{mnw_{ij}\}$. Similarly, the maximum average normalized weight of term t_i in the super-representative RS is obtained by taking the maximum of the corresponding quantities in the component representatives. If the number of super-representatives at a given level is still too large, then they can be grouped into higher level super-representatives by repeating the above process which ends when the number of super-representatives below the root node is sufficiently small. The root node representative contains the same information as that in the root node of the two-level hierarchy case discussed in Section 5, i.e., the global inverse document frequency weight of each term which appears in any local database is kept.

Figure 1 illustrates the process of deriving the quantities for a single term in a hierarchy. The first quantity associated with each node is the maximum normalized weight and the second quantity is the maximum average normalized weight.

8.1 An Algorithm for Searching the Hierarchy

Let RS be the super-representative for superdatabase S that logically contains local databases $D_1, \ldots, D_r$. Let idf_i be the global inverse document frequency weight of the ith term. The similarity of the most similar document in S with the query $q = (q_1 * idf_1, \ldots, q_k * idf_k)$, can be estimated using RS and the root representative as follows. Let the estimate be

$$est\_msim(q, S) = \max_{1 \le i \le k} \left\{ mnw'_i * idf_i * q_i + \sum_{\substack{j=1 \\ j \ne i}}^{k} maw_j * idf_j * q_j \right\} \tag{6}$$

where $mnw'_i = \max\{mnw_i(1), \ldots, mnw_i(r)\}$, $mnw_i(e)$ is the maximum normalized weight of term t_i in database D_e, $1 \le e \le r$; $maw_j = \max\{anw_j(1), \ldots, anw_j(r)\}$, $anw_j(e)$ is the average normalized weight of term t_j in the database D_e, $1 \le e \le r$. (In principle, the weights in the query should be divided by the norm of the query. However, since we are comparing the similarity of the most similar document in one superdatabase against that of another, the norm of the query is in common and can be ignored.) Note that

mnw_i' and maw_j are kept in the representative of the superdatabase S. For every database D_e, we estimate the similarity of the most similar document in D_e by

$$est\_msim(q, D_e) = \max_{1 \le i \le k} \left\{ mnw_i(e) * idf_i * q_i + \sum_{\substack{j=1 \\ j \ne i}}^{k} anw_j(e) * idf_j * q_j \right\} \tag{7}$$

Since $mnw_i' \ge mnw_i(e)$ and $maw_j \ge anw_j(e)$, by comparing equation (6) with equation (7), we have $est\_msim(q, S) \ge est\_mism(q, D_e)$. This result is summarized as follows.

Lemma 1 *Let S be a superdatabase containing databases $D_1, \ldots, D_r$. For a given query q, let $est\_mism(q, S)$ and $est\_mism(q, D_e)$ be the estimated similarity of the most similar document in S and in D_e, respectively. Then, $est\_msim(q, S) \ge est\_msim(q, D_e)$.*

Suppose superdatabase S and other superdatabases are contained in a superdatabase T. Recall that the representative of T is formed from the representatives of the children of T in exactly the same way that the representative of S is formed from the representatives of $D_1, \ldots, D_r$. Thus, the estimated similarity of the most similar document in T, denoted by $est\_msim(q, T)$, satisfies $est\_msim(q, T) \ge est\_msim(q, S)$.

Consider a path P containing $A_1, A_2, \ldots, A_s$ in the hierarchy, where each A_i represents the representative of a superdatabase and A_{i+1} is a child of A_i. With a slight abuse of notation, we also use A_i to denote a database since there is a 1–1 correspondence between a database and its representative. Based on our discussion, we have $est\_msim(q, A_i) \ge est\_msim(q, A_{i+1})$ for any given query. By transitivity of inequality, we have the following lemma.

Lemma 2 *In the hierarchy of representatives, if A_i is an ancestor of A_j, then $est\_msim(q, A_i) \ge est\_msim(q, A_j)$ for any given query q.*

This property will allow us to devise a *best-first* search strategy which ranks databases exactly the same as if the similarity of the most similar document in each database D_i were computed and then the databases were ranked in descending order of $est\_msim(q, D_i)$. This best-first strategy avoids the computation of $est\_msim(q, D_j)$ for most databases D_j.

We now provide an algorithm to search this hierarchy of representatives. The main idea of the best-first search algorithm is as follows. For a given query, we estimate the similarity of the most similar document in each child of the root. These child nodes are arranged to form a list in descending order of estimated similarity. The representative which yields the largest estimated similarity is selected. If it is the representative of a local database, then the corresponding search engine is invoked and documents are retrieved in the way as described in Section 6. If it is a super-representative, i.e., an intermediate node in the hierarchy, then the similarity of the most similar document in each of its child representatives is estimated. These child representative nodes are arranged in descending order of similarity and merged with the current list of nodes in descending order of similarity to form a list of representative nodes in the same order. In this list, the estimated similarities can be due to the representatives of local databases or from the non-leaf super-representatives. In either case, we always take the largest value. If the largest value is from the representative of a local database, then the corresponding search engine is invoked and documents are retrieved according to Section 6; otherwise, the best-first search process is executed on the node with the largest estimated similarity value. The details are as follows.

Best_First_Search(n, q, Root) /* n is the number of documents to be retrieved;
q is the query; *Root* is the root node of the hierarchy */

1. Initialization: *min-sim := 1*; /* the minimum of the similarities of the retrieved documents from previously searched databases is initially set to be 1 (the highest possible similarity) */

2. The similarity of the most similar document in each child of the root node, *Root*, is estimated. These child nodes are arranged in a list L in descending order of the estimated similarities.

3. The first node, say N, is removed from L.
   If it is the representative of a local database D, then
   Step (3.1): {
   (a) local database D is searched;
   (b) the most similar document and its similarity with q, *csim*, are returned to the result-merger;
   (c) If *csim* > *min-sim* then {
      i. send from database D all documents with similarity $\geq$ *min-sim* to the result-merger;
      ii. if n or more documents have been received by the result merger, then take the n most similar documents and stop;

   }
   else {for each local database D' which has been searched, do
      i. send all documents from D' which have similarities $\geq$ *csim* (but have not been transmitted) to the result-merger;
      ii. *min-sim := csim*;
      iii. if n or more documents have been received by the result-merger, then take the n most similar documents and stop;

   } } /* end of step (3.1) */

   else { Step (3.2):
   (a) the similarity of the most similar document in each child of N is estimated;
   (b) these child nodes are arranged in a list L_1 in descending order of estimated similarity;
   (c) L and L_1 are merged to form L in descending order of estimated similarity;

   }

4. repeat Step (3);

The *result merger* collects the transmitted documents from the searched databases to form the list of n documents to present to the user.

8.2 Effectiveness of Algorithm Best_First_Search(n, q, Root)

In Algorithm $Best\_First\_Search(n, q, Root)$, the result-merger gathers the documents retrieved from various local databases until n or more documents have been received. Step (3.1) is exactly the same process for deciding which documents from the selected databases will form the n documents to present to the user as that described in Section 6. In the two-level hierarchy, the representatives (the leaf-nodes) are arranged in a list in descending order of estimated similarity and then Step (3.1) is executed to those databases in the ordered list. In Algorithm $Best\_First\_Search(n, q, Root)$, the leaf-nodes and the intermediate nodes are interleaved in the list of nodes in descending order of estimated similarity. Whenever a leaf-node, i.e., the representative of a local database is reached, Step (3.1) is executed to determine the documents to retrieve from the local database. Whenever a non-leaf node is encountered, it is replaced by its children. If the leaf-nodes executed by Step (3.1) are exactly in the same order as if databases were arranged in descending order of estimated similarity, then this algorithm will give the same retrieval performance of a two-level hierarchy as described in previous sections. The following proposition establishes this fact.

Proposition 4 Consider any two local databases D_i and D_j. Suppose the estimated similarity of the most similar document in D_i is higher than that of the most similar document in D_j, i.e., $est\_msim(q, D_i) > est\_msim(q, D_j)$. Then Step (3.1) of Algorithm Best_First_Search(m, q, Root) will execute on database D_i before it executes on database D_j.

Proof: Consider the parent P of database D_j in the hierarchy. Two cases may occur:
Case 1: D_i's parent is the root node. If $est\_msim(q, D_i) > est\_msim(q, P)$, then D_i will be ahead of P in the list L. As a result, D_i will be executed by Step (3.1) before D_j. Even if $est\_msim(q, P) > est\_msim(q, D_i)$, Step (3.2) will compute $est\_msim(q, D_j)$. Since $est\_msim(q, D_i) > est\_msim(q, D_j)$, D_i will be ahead of D_j in the list L, due to Step (3.2)(c).

Case 2: D_i's parent is not the root. Let C be a child of the root and also an ancestor of D_i. Consider the path, PATH, from C to D_i. For each node N in the PATH, including C but excluding D_i, $est\_msim(q, N) \geq est\_msim(q, D_i)$ by Lemma 2. Since $est\_msim(q, N) \geq est\_msim(q, D_i) > est\_msim(q, D_j)$, repeated application of Step 3 will place C, then the child of C in PATH, followed by the grandchild of C in PATH and eventually D_i in the list L. All these nodes are ahead of D_j, due to the inequality. Thus, D_i will be executed before D_j. ■

Observation: This proposition guarantees that the databases will be searched in descending order of estimated similarity using our method (i.e., equations (6) and (7)) of estimating similarity of the most similar document in a database or superdatabase. The same result holds for any other estimation method as long as the estimation method is a non-decreasing function of the two parameters, namely the maximum normalized weight and the maximum average normalized weight (the average normalized weight in the case of a local database). Due to the non-decreasing property of an estimation function est (i.e. Lemma 2), $est\_msim(q, A) \geq est\_msim(q, B)$ whenever A is an ancestor of B. Thus, Proposition 4 holds.

9 Conclusions

With the increase of the number of search engines on the World Wide Web, providing easy, efficient and effective access to text information from multiple sources has increasingly become necessary. In this paper, we proposed a new methodology to find the most similar documents from multiple text databases. Our contributions consist of

1. A simple condition to rank databases optimally with respect to a given query.

2. Three algorithms to estimate the similarity of the most similar documents in each database; one of these algorithms, namely *Fast-Similarity Method*, is shown to yield much better retrieval effectiveness than the *high-correlation method* and uses the same amount of space. It is linear in time complexity. The other two estimation algorithms are more accurate in ranking databases, but they use more storage and/or require higher time complexity.

3. An algorithm (namely *OptDocRetrv*) was given to provide a cutoff (i.e., determine which databases need to be searched) and to determine which documents need to be transmitted. Experimental results showed that when it is used in conjunction with any of the three estimation algorithms, the number of databases searched by this algorithm is, on the average, only slightly above the number of databases containing the n most similar documents.

4. When the number of databases is very large, we proposed a hierarchy of representatives with the number of levels > 2. We provided an algorithm to search the hierarchy. It was shown that the search will produce the same effectiveness as the corresponding two-level hierarchy (i.e., the query is compared against all database representatives.)

The following are research issues to be examined.

1. The experimental results obtained by us are based on term matching. In other words, if query q and document d have no term in common, then d cannot be retrieved. However, techniques such as Latent Semantic Indexing [13] may permit q and d to match after term transformation. In that case, our database representative should be based on the transformed terms, i.e., statistics such as the maximum term weights are collected on the transformed terms. When a query is submitted, its terms are transformed into a new set of terms before it is matched against the database representatives. Whether this approach yields good results remains to be verified. In [15], queries are expanded by terms having similar contexts. How the expansion is to be carried out effectively in a large number of databases remains to be seen.

2. In the Internet environment, Web pages are extensively linked. Such linkage information provide valuable information about the degrees of importance of documents, see for example [5, 26] and should be utilized for database and document selection.

3. Much larger data collections should be utilized for experimentation. Unfortunately, we are not aware of a large collection which has relevance assessment of documents, large number of short typical Internet queries and extensively linked documents. The authors of [10, 11] partitioned previous TREC collections into databases and used them together with TREC queries as testbeds. However, the TREC

queries for previous TREC collections are much longer than typical Internet queries and there is no linkage information among the documents in these collections. The recent TREC VLC2 collection contains multi-gigabyte of real Web pages. There is potential that a testbed can be created using this collection. At present, this collection may not have extensive relevance assessment information.

4. In order to demonstrate the efficiency of searching a hierarchy of database representatives, good clustering algorithms are needed to cluster the representatives properly so that for most queries, only very few representatives are compared. Experimentation is required.

Acknowledgement: This work is supported in part by: NSF (IIS-9902792, IIS-9902872, CCR-9816633, CCR-9803974, CDA-9711582, HRD-9707076), NASA (NAGW-4080, NAG5-5095) and ARO (NAAH04-96-1-0049, DAAH04-96-1-0278). We are grateful to L. Gravano and H. Garcia-Molina for providing us with the collection of documents and queries used in our experiments and to W. Wu for supplying some of the programs. We also would like to thank the anonymous reviewers for their constructive comments to an earlier version of the paper.

References

[1] G. Abdulla, B. Liu, R. Saad, and E. Fox. *Characterizing World Wide Web Queries.* TR-97-04, Virginia Polytechnic Institute and State University, 1997.

[2] C. Baumgarten. *A Probabilistic Model for Distributed Information Retrieval.* ACM SIGIR Conference, Philadelphia, 1997.

[3] C. Baumgarten. *A Probabilistic Solution to the Selection and Fusion Problem in Distributed Information Retrieval.* ACM SIGIR Conference, 1999.

[4] N. J. Belkin, P. Kantor, E. A. Fox and J. A. Shaw. *Combining the Evidence of Multiple Query Representations for Information Retrieval.* Information Processing & Management, 31(3), pp. 431-448, May-June 1995.

[5] S. Brin, and L. Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine". WWW7 Conference, 1998.

[6] J. Callan, Z. Lu, and W. Bruce Croft. *Searching Distributed Collections with Inference Networks.* ACM SIGIR Conference, Seattle, 1995.

[7] A. Chakravarthy, and K. Haase. *NetSerf: Using Semantic Knowledge to Find Internet Information Archives.* ACM SIGIR Conference, pp.4-11, Seattle, 1995.

[8] D. Dreilinger, and A. Howe. Experiences with Selecting Search Engines Using Metasearch. ACM TOIS, 15(3), July 1997.

[9] Y. Fan, and S. Gauch. *Adaptive Agents for Information Gathering from Multiple, Distributed Information Sources.* AAAI Symposium on Intelligent Agents in Cyberspace, Stanford University, March 1999.

[10] J. French, A. Powell, C. Viles, T. Emmitt, and K. Prey. *Evaluating Database Selection Techniques: A Testbed and Experiment.* ACM SIGIR Conference, 1998.

[11] J. French, A. Powell, J. Callan, C. Viles, T. Emmitt, K. Prey, and Yun Mou. *Comparing the Performance of Database Selection Algorithms.* ACM SIGIR Conference, 1999.

[12] N. Fuhr. *A Decision-Theoretic Approach to Database Selection in Networked IR.* ACM Transactions on Information Systems, 17:3, July 1999, pp. 229-249.

[13] G. Furnas, S. Deerwester, S. Dumais, T. Landauer, R. Harshman, L. Streeter and K. Lochbaum. *Information Retrieval Using a Singular Value Decomposition Model of Latent Semantic Structure.* ACM SIGIR Conference, 1988.

[14] S. Gauch, G. Wang, and M. Gomez. *ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines.* Journal of Universal Computer Science, 2(9), pp. 637-649, 1996.

[15] S. Gauch, J. Wang, S. Rachakonda. *A Corpus Analysis Approach for Automatic Query Expansion and Its Extension to Multiple Databases.* ACM Transactions on Information Systems, 17:3, July 1999, pp. 250-269.

[16] L. Gravano, C. Chang, H. Garcia-Molina, and A. Paepcke. *STARTS: Stanford Proposal for Internet Meta-Searching.* ACM SIGMOD Conference, Tucson, May 1997, pp. 207-218.

[17] L. Gravano, and H. Garcia-Molina. *Generalizing GlOSS to Vector-Space databases and Broker Hierarchies.* International Conferences on Very Large Data Bases, 1995.

[18] L. Gravano, and H. Garcia-Molina. *Generalizing GlOSS to Vector-Space databases and Broker Hierarchies.* Technical Report, Computer Science Dept., Stanford University, 1995. (This report discussed how to estimate the database usefulness used defined in this paper for the high-correlation and disjoint scenarios. Such discussion did not appear in [17].)

[19] L. Gravano, and H. Garcia-Molina. *Merging Ranks from Heterogeneous Internet sources.* International Conferences on Very Large Data Bases, 1997.

[20] D. Harman. *Overview of the First Text Retrieval Conference.* Edited by D. Harman, Computer Systems Technology, U.S. Department of Commerce, NIST, 1993.

[21] A. Howe, and D. Dreilinger. *SavvySearch: A Meta-Search Engine that Learns Which Search Engines to Query.* AI Magazine, 18(2), 1997.

[22] Information and Data Management: Research Agenda for the 21st Century, Information and Data Management Program, National Science Foundation, March 29-31, 1998.

[23] 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.

[24] B. Kahle, and A. Medlar. *An Information System for Corporate Users: Wide Area information Servers.* Technical Report TMC199, Thinking Machine Corporation, April 1991.

[25] S. Kirsch. *The Future of Internet Search: Infoseek's Experiences Searching the Internet.* ACM SIGIR Forum, 32:2, pp. 3-7, 1998.

[26] J. Kleinberg. *Authoritative sources in Hyperlinked Environment.* ACM-SIAM Symposium on Discrete Algorithms, 1998.

[27] M. Koster. *ALIWEB: Archie-Like Indexing in the Web.* Computer Networks and ISDN Systems, 27:2, 1994, pp. 175-182 (http://www.cs.indiana.edu/aliweb/form.html).

[28] G. Kowalski. *Information Retrieval Systems, Theory and Implementation.* Kluwer Academic Pub., 1997.

[29] K. Kwok and M. Chan. *Improving Two-Stage Ad-Hoc Retrieval for Short Queries*, ACM SIGIR Conference, 1998.

[30] S. Lawrence, and C. Lee Giles. *Searching the World Wide Web.* Science, 280, April 1998, pp. 98-100.

[31] S. Lawrence, and C. Lee Giles. *Accessibility of Information on the Web.* Nature, 400, July 1999, pp. 107-109.

[32] K. Lam, and C. Yu. *A Clustered Search Algorithm Incorporating Arbitrary Term Dependencies.* ACM Transactions on Database Systems, September 1982.

[33] K. Liu, C. Yu, W. Meng, W. Wu, and N. Rishe. *A Statistical Method for Estimating the Usefulness of Text Databases.* IEEE Transactions on Knowledge and Data Engineering (to appear).

[34] U. Manber, and P. Bigot. *The Search Broker.* USENIX Symposium on Internet Technologies and Systems (NSITS'97), Monterey, California, 1997, pp. 231-239.

[35] W. Meng, K-L. Liu, C. Yu, X. Wang, Y. Chang, and N. Rishe. *Determining Text Databases to Search in the Internet.* International Conferences on Very Large Data Bases, New York City, 1998.

[36] W. Meng, K. Liu, C. Yu, W. Wu, and N. Rishe. *Estimating the Usefulness of Search Engines.* 15th IEEE International Conference on Data Engineering (ICDE'99), Sydney, Australia, March 1999.

[37] Networked Computer Science Technical Reports Library, http://lite.ncstrl.org:3803/.

[38] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* New York: McCraw-Hill, 1983.

[39] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison Wesley, 1989.

[40] E. Selberg, and O. Etzioni. *Multi-Service Search and Comparison Using the MetaCrawler.* 4th International World Wide Web Conference, December 1995.

[41] E. Selberg, and O. Etzioni. *The MetaCrawler Architecture for Resource Aggregation on the Web.* IEEE Expert, 1997.

[42] A. Singhal, C. Buckley. and M. Mitra. *Pivoted Document Length Normalization.* ACM SIGIR, 1996.

[43] K. Sparck Jones. *Statistical Interpretation of Term Specificity and Its Application in Retrieval.* J. of Documentation, 1972.

[44] G. Towell, E. Voorhees, N. Gupta, and B. Johnson-Laird. *Learning Collection Fusion Strategies for Information Retrieval.* 12th Int'l Conf. on Machine Learning, 1995.

[45] E. Voorhees, N. Gupta, and B. Johnson-Laird. *Learning Collection Fusion Strategies.* ACM SIGIR Conference, Seattle, 1995.

[46] J. Xu, and J. Callan. *Effective Retrieval with Distributed Collections.* ACM SIGIR Conference, 1998.

[47] T. W. Yan, and H. Garcia-Molina. *SIFT – A Tool for Wide-Area Information Dissemination.* USENIX 1995 Technical Conference, 1995.

[48] Y. Yang, and C. Chute. *An Example-based Mapping Method for Text Categorization and Retrieval.* ACM Transactions on Information Systems, 1994, pp. 252-277.

[49] C. Yu, K. Liu, W. Wu, W. Meng and N. Rishe. *Finding the Most Similar Documents across Multiple Text Databases.* IEEE Conference on Advances in Digital Libraries (ADL'99), pp. 150-162, Baltimore, Maryland, May 1999.

[50] C. Yu, W. Meng, K. Liu, W. Wu, and N. Rishe. *Efficient and Effective Metasearch for a Large Number of Text Databases.* Eighth ACM International Conference on Information and Knowledge Management (CIKM'99), Kansas City, November 1999.

[51] C. Yu, W. Luk and M. Siu. *On the Estimation of the Number of Desired Records with respect to a Given Query.* ACM Transactions on Database Systems, March 1978.

[52] C. Yu, and W. Meng. *Principles of Database Query Processing for Advanced Applications.* Morgan Kaufmann, San Francisco, 1998.

[53] B. Yuwono, and D. Lee. *Server Ranking for Distributed Text Resource Systems on the Internet.* 5th Int'l Conf. On Database Systems For Adv. Applic. (DASFAA'97), April 1997, pp. 391-400.