Finding the Most Similar Documents across Multiple Text Databases*

Clement Yu[1], King-Lup Liu[1], Wensheng Wu[1], Weiyi Meng[2], Naphtali Rishe[3]
[1] Dept. of EECS, University of Illinois at Chicago, Chicago, IL 60607
[2] Dept. of Computer Science, SUNY – Binghamton, Binghamton, NY 13902
[3] School of Computer Science, Florida International University, Miami, FL 33199

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, databases are ranked in a certain order. Next, documents are retrieved from the databases according to the order and in a particular way. If the databases containing the n most similar documents for a given query can be ranked ahead of other databases, the methodology will guarantee the retrieval of the n most similar documents for the query. A statistical method is provided to identify databases, each of which is estimated to contain at least one of the n most similar documents. Then, a number of strategies is presented to retrieve documents from the identified databases. Experimental results are given to illustrate the relative performance of different strategies.

1 Introduction

The Internet has become a vast information source in recent years and can be considered as the world's largest digital library. To help ordinary users find desired data in this library, many *search engines* have been created. Each search engine has a corresponding *database* that defines the set of documents that can be searched by the search engine. Usually, an 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 papers in a 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 just an interface. 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.

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 we should be careful about which local search engines to invoke for a given query. It is likely that for a given query, only a small fraction 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, we should first identify those search engines that are most likely to provide useful results to each query and then pass the query to only the identified search engines. Examples of systems that employ this approach include WAIS [13], ALIWEB [14], gGlOSS [6], SavvySearch [10] and D-WISE [33]. 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

*This research is supported by the following organizations: NSF (IRI-9509253, CDA-9711582, HRD-9707076), NASA (NAGW-4080, NAG5-5095) and ARO (NAAH04-96-1-0049, DAAH04-96-1-0278).

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 *collection fusion problem*.

In this paper, we study both the database selection and the collection fusion problems. First, we identify a sufficient condition that databases need to be ranked and documents need to be retrieved such that the n most similar documents for a given query can be obtained. Second, in order that databases can be identified to contain the most similar documents, we employ the *combined-term method* [18] to estimate the number of documents in a given database which have high similarities with the given query. This method extends our *subrange-based method* [21] by incorporating certain term dependencies between adjacent terms. We also show that this method, as well as the subrange-based method, will be able to optimally retrieve the n most similar documents for single-term queries. In the Internet, it is known [1, 7, 12] that single-term queries are submitted frequently. Third, from databases supplied by Stanford University, we compare performances of the combined-term method with the high-correlation method [6, 7]. We also compare different ways of retrieving documents from databases which have been ranked in a given order by our combined-term method. The experimental results demonstrate the superior performance of our method.

The paper is organized as follows. In section 2, the database selection problem and the collection fusion problem are discussed in more detailed in the context of this paper. A summary of related work is given in Section 3. In Section 4, a sketch of the subrange-based method and that of the combined-term method are provided. In section 5, a sufficient condition for ranking databases in an optimal order to retrieve the n most similar documents for any given query, for any given n, is given. The subrange-based method and the combined-term method are shown to rank the databases optimally for single-term queries. A strategy is given to retrieve documents from databases which have been ranked. It is shown that for the optimally ranked databases (i.e., databases containing one or more of the n most similar documents are ranked ahead of other databases), all the n most similar documents will be retrieved. In Section 6, experimental results comparing the performances of the high-correlation method and the combined-term method are given. Further experimental results involving the combined-term method in conjunction with the strategy to retrieve documents from the ranked databases are presented to show that the percentages of the n most similar documents retrieved vary from 88% to 95% when n varies from 5 to 30.

2 The Database Selection Problem and the Collection Fusion Problem

We assume that the vector-space model [23] 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 [23]. 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) [23, 32]. It may also depend on the number of documents having the term relative to the total number of documents in the database. A query is similarly transformed into a vector with weights. The similarity between a query and a document can be measured by 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 $(x_1, x_2, ..., x_n)$ is $\sqrt{\sum_{i=1}^{n} x_i^2}$. This is to normalize the similarity between 0 and 1. The similarity function with such a normalization is known as the *Cosine* function [23, 32]. Other similarity functions, see for example [27], are also possible.

Given a similarity function g, and a query q, the n most similar documents are those documents which have the n largest $g(q, d)$ values, where d is a document. The documents are distributed in m databases D_1, .., D_m. The database selection problem is to identify 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 (if there are too many databases, then the representatives can be arranged in a hierarchy so that not all representatives need to be compared against the query — this will not be addressed in this paper). Based on a comparison between q and database D_i, estimates of the number of documents in D_i having similarities $\geq T$, for various values of T, are obtained. By combining these estimates from various databases, it is possible to determine an appropriate threshold T_0 such that the sum of the expected numbers of documents from the m databases having similarities $\geq T_0$ is equal to or slightly larger than n. In Section 4, we will employ the subrange-based method and the combined-term method to solve the database selection problem.

Consider the problem of retrieving documents from the databases each of which has been estimated to contain one or more of the n most similar documents. In the Internet environment, local search engines are likely to be autonomous and they may rank locally retrieved documents using different local similarity functions. Since local similarities across multiple databases are not comparable, 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 that with respect to the global database. In our earlier paper [20], we proposed two solutions to this problem. One 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 $\geq T_i$. 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 to a local search engine such that the local similarity of the modified query with a document in that local database is the same as the global similarity of the original user query with that document. In either solution, the actual global similarities of documents from local databases can be determined. Assuming that certain databases have been identified from which documents are to be retrieved, we need to decide which documents to be retrieved from each such identified database to make up the n globally most similar documents to be shown to the user. One simple strategy is to retrieve n documents having the largest global similarities from each such database, merge them into a list, sort them 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, the total number of documents to be retrieved will be $k * n$, where k is the number of databases which have been identified. The issue is whether fewer documents can be retrieved without losing any of the n most similar documents unnecessarily.

A related problem is as follows. Some database selection algorithms rank databases in a certain order without a precise cutoff. In other words, documents from database D_i should be considered before documents from database D_j, if D_i is ahead of database D_j in the ranking. In that case, it is of interest to specify a retrieval strategy to determine which documents from which databases should be retrieved and returned to the user.

3 Related Work

The solutions to the database selection problem can be roughly classified into the following five categories.

1. The *naive approach* does not perform database selection and the metasearch engine simply sends each user query to all search engines. This approach is employed by MetaCrawler [25, 26].

2. *Rough approaches* represent the contents of a database using a rough description, such as a few words or a few paragraphs. The database representatives are typically manually made and are not sufficiently informative for serious database selection. Example systems that employ rough approaches are WAIS [13], ALIWEB [14] and Search Broker [19]).

3. *Qualitative Approaches* use rather detailed information such as the *document frequency* of each term to represent the contents of a database. Based on the information, a ranking score of each database can be computed to reflect the relative usefulness (or quality) of a database to a query. However, the score of a database is not equivalent to the number of potentially useful documents in the database. Qualitative approach is employed by a version of gGlOSS [6], CORI net [3] and D-WISE [33].

4. *Quantitative approaches* use detailed information to represent the contents of a database and they directly estimate the number of potentially useful documents in each database. Our estimation approaches belong to this category. Among quantitative approaches for vector queries, the approach in [31] is for the *binary and independent* case where each document d is represented 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, and terms are assumed to be independent. This method was later extended to the *binary and dependent* case in [17]. A substantial amount of information will be lost when documents are represented by binary vectors. The database representative used in gGlOSS can be used to estimate how many documents in a database are potentially useful [7]. However, the estimation method is very different from ours and will be shown to be much less accurate. In addition, the estimation methods employed in [6, 7] are based on two very restrictive assumptions.

5. *Learning-based Approaches* make use of past retrieval experiences with respect to a database to determine the usefulness of the database. SavvySearch [4] is an example of a learning-based approach. SavvySearch only ranks local search engines and does not estimate the number of potentially useful documents. The method in [2] proposes to select databases based on the estimated probability of relevance distribution of their documents. Relevance feedback is usually needed to obtain reasonable estimation of the distribution.

The two solutions to the database selection problem to be discussed in this paper, the subrange-based method and the combined-term method, are quantitative approaches we developed [21] and extended in [18].

The proposed solutions to the *collection fusion problem* can also be classified into five categories. The *local determination approach* simply allows each local search engine to return all documents it retrieved. NCSTRL (http://www.ncstrl.org/) employs such an approach. 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. More sophisticated *weighted allocation approaches* retrieve proportionally more documents from local search engines whose databases have higher ranking scores. CORI net 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 [28, 29] are based on the use of *training queries*. The probabilistic model [2] is more suitable in a feedback environment, because it may depend on accurate estimates of certain parameter values. Weighted allocation and learning-based approaches are heuristic in nature and they do not guarantee that all globally potentially useful documents will be retrieved from each local search engine. The *guaranteed retrieval approach* aims at guaranteeing such a property. The algorithm in [8] while guaranteeing that all potentially useful documents are retrieved may unnecessarily retrieve many non-similar documents. Our earlier approach in [20] is a guaranteed retrieval approach but with a second goal of minimizing the number of non-similar documents to be 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 solution we propose in this paper has the property that when it is used together with the subrange-based method or the combined-term method, it will retrieve all the n most similar documents for any single-term query. Experimental results will show that with our approach between 88% to 95% of the most similar documents are retrieved for queries of various lengthes.

4 Methods for Usefulness Estimation

In this section, we present our solutions to the *database selection problem.* In Section 4.1, we review the subrange-based method [21] based on the *Non-binary Independence Model* (NBIM) to estimate the *usefulness* of a database with respect to a given query. For a given query and a given global similarity threshold, the usefulness of a database is defined to be the number of documents in the database that have global similarities with the query greater than the global threshold. In NBIM, term weights can be any non-negative real numbers and the occurrences of different terms in each document are assumed to be independent. In Section 4.2., we present the combined-term method which is an extension of the subrange-based method by incorporating one type of term dependencies between adjacent terms.

4.1 Subrange-based Method with NBIM

Consider a database D with 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 t_i in representing the document, $1 \le i \le m$. Each query can be similarly represented. 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 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 [23]. 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.

A database D with m distinct terms can be represented as m pairs $\{(p_i, w_i) \mid i = 1, \ldots, m\}$, where p_i is the probability that term t_i appears in a document in D and w_i is the average of the weights of t_i in the set of documents containing t_i. For a given query $q = (u_1, u_2, ..., u_m)$, the database representative is used to estimate the usefulness of D. Without loss of generality, we assume that only the first r u_i's are non-zero, $0 < r \le m$. Therefore, q becomes $(u_1, u_2, ..., u_r)$ and $sim(q, d)$ becomes $u_1 * d_1 + \cdots + u_r * d_r$. This implies that only the first r terms in each document in D need to be considered.

Consider the following generating function:

$$(p_1 * X^{w_1 * u_1} + (1 - p_1)) * (p_2 * X^{w_2 * u_2} + (1 - p_2)) * \cdots * (p_r * X^{w_r * u_r} + (1 - p_r)) \quad (1)$$

where X is a dummy variable. The following proposition [20] relates the coefficients of the terms in the above function with the probabilities that documents in D have certain similarities with q.

Proposition 1. Let q and D be defined as above. If the terms are independent and the weight of term t_i whenever present in a document is w_i, which is given in the database representative ($1 \leq i \leq r$), then the coefficient of X^s in function (1) is the probability that a document in D has similarity s with q.

Example 4.1 Let q be a query with three terms with all weights equal to 1, i.e., $q = (1, 1, 1)$. (for ease of understanding, the weights of terms in the query and documents are not normalized). Suppose database D has five documents and their vector representations are (only components corresponding to query terms are given): (2, 0, 2), (0, 1, 1), (2, 0, 0), (0, 0, 3) and (0, 0, 0). Namely, the first document has query term 1 and query term 3, and their corresponding weights are both 2. Other document vectors can be interpreted similarly. From the five documents in D, we have $(p_1, w_1) = (0.4, 2)$ as 2 out of 5 documents have term 1 and the average weight of term 1 in the two documents is 2. Similarly, $(p_2, w_3) = (0.2, 1)$ and $(p_3, w_3) = (0.6, 2)$. Therefore, the corresponding generating function is:

$$(0.4 * X^2 + 0.6)(0.2 * X + 0.8)(0.6 * X^2 + 0.4) \quad (2)$$

Consider the coefficient of X^3 in the function. Clearly, it is the sum of $p_1 * p_2 * (1 - p_3)$ and $(1 - p_1) * p_2 * p_3$. The former is the probability that a document in D has exactly the first two query terms and the corresponding similarity with q is $w_1 + w_2$ (=3). The latter is the probability that a document in D has exactly the last two query terms and the corresponding similarity is $w_2 + w_3$ (=3). Therefore, the coefficient of X^3, namely, $p_1 * p_2 * (1 - p_3) + (1 - p_1) * p_2 * p_3 = 0.104$, is the estimated probability that a document in D has similarity 3 with q. ■

By expanding generating function (1) and merging the terms with the same X^s, we obtain

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

We assume that the terms in (3) are listed in descending order of the exponents, i.e., $b_1 > b_2 > ... > b_c$. By Proposition 1, a_i is the probability that a document in D has similarity b_i with q. In other words, if database D contains N documents, then $N * a_i$ is the expected number of documents that have similarity b_i with query q.

Suppose a user query requests that the n most similar documents across all local databases be retrieved. The expansions (3), one for each local database, can be used to convert such a number n to the global threshold T for the user query. The idea is to convert the expansion (3) for each database into a similarity distribution list such that, for any given global threshold, it is easy to determine from the list the number of documents in the database having similarities greater than the threshold. After the similarity distribution lists of all databases are obtained, a desired global threshold T can easily be found such that the total number of documents having similarities greater than T across all databases is either n or slightly larger than n (the latter case can be caused when different documents may have the same global similarities). The following example illustrates the conversion of n to T.

Example 4.2 Continue Example 4.1. When expression (2) is expanded, we have

$$0.048 * X^5 + 0.192 * X^4 + 0.104 * X^3 + 0.416 * X^2 + 0.048 * X + 0.192 \quad (4)$$

Based on the expanded expression and Proposition 1, we can obtain the following similarity distribution list for this database: {(5, 0.24), (4, 1.20), (3, 1.72), (2, 3.80), (1, 4.04), (0, 5)}. In each pair, the first number is a global similarity and the second number indicates the estimated number of documents in the database whose similarities are greater than or equal to the first number. The first pair (5, 0.24) is obtained from the first term in expression (4) and the second pair (4, 1.20) is obtained by combining the first two terms in the same expression. Other pairs can be computed similarly. When the second number in each pair is rounded up and pairs whose second numbers are zero (after the rounding) are discarded, the following modified similarity distribution list is obtained: {(4, 1), (3, 2), (2, 4), (1, 4), (0, 5)}. This (rounded) distribution list indicates that database D contains 1 document having similarities 4 or higher with the query q, 2 documents having similarities 3 or higher with q, etc.

From the similarity distribution lists of all local databases, determining a desired global threshold T for the n most similar documents across all databases can be done easily. Suppose for another database we have the following (rounded) similarity distribution list: $\{(5, 1), (4, 2), (3, 2), (2, 2), (1, 4), (0, 4)\}$. In this case, if only one most similar document is desired, then any threshold between 4 and 5 (excluding 4) can be used; if two most similar documents are needed, then any threshold between 3 and 4 (excluding 3) can be used. In general, by trying the global threshold in decreasing values, a desired threshold can be found. ■

For a given similarity threshold T, let C be the largest integer to satisfy $b_C > T$. Then, the *usefulness* of D for query q based on threshold T, namely, the number of documents whose similarities with query q are greater than T, can be estimated as:

$$est\_usefulness(T, q, D) = \sum_{i=1}^{C} N * a_i = n \sum_{i=1}^{C} a_i \quad (5)$$

There are two unrealistic assumptions in Proposition 1. First, the assumption about the weight of a term being uniform among documents containing the term is not realistic. Second, the terms are not necessarily independent. As a result, the estimated usefulness may be inaccurate. One of the methods we proposed to remove the first assumption is the following subrange-based method [21]. In Section 4.2, we will present a method for tackling the second assumption.

Let t_i be a term in a query. Based on the above solution, the polynomial $p_i * X^{w_i * u_i} + (1 - p_i)$ will be a factor in the generating function (1). As an example, consider 8 documents of which 4 have unnormalized weight of 2 and another 4 have unnormalized weight of 6 for term t_i. The average unnormalized weight of the term for a document having the term is 4. Using the above estimation method, a polynomial of the form $p_i * X^4 + (1 - p_i)$ will be formed for the term, assuming that the query has weight 1 for the term (again, unnormalized weights instead of normalized weights are used for ease of understanding). Since the weight distribution is not uniform, we can improve the modelling by partitioning the documents having the term into two subsets. One subset contains documents having lower unnormalized term weight (term weight = 2 in this example) and another involving documents having larger weights (term weight = 6 in this example). The corresponding polynomial is given by $p_{i1} * X^6 + p_{i2} * X^2 + (1 - p_i)$, where $p_{i1} = p_{i2} = p_i/2$. This example can be generalized as follows. Let $w_{i1}, w_{i2}, ..., w_{ik}$ be the actual weights of t_i in the set of documents having the term, where $k = p_i * n$ is the number of documents having the term and n is the total number of documents in the database. Suppose $w_{i1} \leq w_{i2} \leq ... \leq w_{ik}$. Suppose we partition the weight range of t_i into l subranges of possibly different lengths. Let wm_{ij} be the median of the weights in the j-th subrange, $j = 1, ..., l$. With the assumption that the weight distribution of the term is *normal*, wm_{ij}'s can be approximated using the average weight w_i and the standard deviation σ_i very easily [21]. Then, the distribution of the term weights of t_i may be approximated by the following distribution: The term has a uniform weight of wm_{ij} for the documents whose term weights fall in the j-th subrange, $j = 1, ..., l$. With this weight approximation, for a query containing term t_i, the polynomial $p_i * X^{w_i * u_i} + (1 - p_i)$ in the generating function (1) can be replaced by the following polynomial:

$$p_{i1} * X^{wm_{i1} * u_i} + p_{i2} * X^{wm_{i2} * u_i} + \cdots + p_{il} * X^{wm_{il} * u_i} + (1 - p_i) \quad (6)$$

where p_{ij} is the probability that term t_i occurs in a document and has a weight in the jth subrange, $j = 1, ..., l$. Essentially, polynomial (6) is obtained from $p_i * X^{w_i * u_i} + (1 - p_i)$ by decomposing the probability p_i into l probabilities corresponding to the l subranges. After the above polynomial replacement, the rest of the estimation process is identical to that described earlier. To incorporate the maximum normalized weight into expression (6), we add the component $p_{i0} X^{mw_i * u_i}$ where mw_i is the maximum normalized weight of the term and $p_{i0} = 1/N$ (N being the number of documents in the local database). Thus, the expected number of documents having the maximum normalized weight is exactly 1. This is reasonable as it is unlikely that there is another document with exactly the same normalized weight. Since the sum of probabilities $p_{i0} + p_{i1} + \cdots + p_{i1} + (1 - p_i)$ should be 1, we adjust p_{i1} to be $(p_{i1} - p_{i0})$.

4.2 The Combined-Term Method

The assumption that terms are independently distributed in Proposition 1 is not entirely realistic. For example, the two terms "computer" and "algorithm" may appear together more frequently than any arbitrarily chosen two words. In this subsection, we present the combined-term method which remedies the term independence assumption in the subrange-based method by incorporating one type of dependency 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. As pointed out earlier, it is unlikely that another document in the database has the same maximum normalized weight under the combined term. If the two terms were independently distributed in the documents of the database, then the probability that a document in the database has the normalized sum of term weights mnw_{ij} under the two terms t_i and t_j can be estimated using the subrange-based estimation method. Specifically, a polynomial representing the probability that a document has the maximum normalized term weight for t_i, followed by the probabilities that a document has certain percentiles of weights for term t_i can be written (see Example 3). Similarly, another such polynomial can be written for term t_j. By multiplying these two polynomials together, the desired probability can be estimated. The criteria that the two terms t_i and t_j should be combined into a single term t_{ij} is that the estimated probability 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 latter condition is to ensure that if the combined term is used, it will not lead to smaller similarities. The former condition is implemented by computing the difference in absolute value between $1/N$ and the estimated probability and then comparing to a preset threshold. If the difference exceeds the threshold, then the two terms should be combined. The difference for the term pair t_i and t_j is denoted by d_{ij} and is stored together with the combined term t_{ij}.

If the two terms are combined, then we obtain from the documents containing both terms the distribution of the sum of the normalized weights of the two terms. From the distribution, we apply the subrange-based method for the combined term. For a combined term t_{ij}, we store the maximum normalized sum mnw_{ij}, the average normalized sum, its standard deviation, its probability of occurrence and its difference d_{ij}. The last quantity is utilized to determine which term should be combined with a given term in a query and will be explained later.

Example 4.3 Suppose that the user's query q is "computer algorithm", and that normalized term weight is used in this example.

Let the maximum normalized weight for the terms "computer" and "algorithm" be $mw_1 = 0.458$ and $mw_2 = 0.525$, respectively. Suppose that the polynomials for the two terms are

$$0.0013 * X^{0.458} + 0.00016 * X^{0.374} + 0.0054 * X^{0.316}$$
$$+0.0279 * X^{0.198} + 0.0174 * X^{0.101}$$
$$+0.0174 * X^{0.0056} + 0.93$$

and

$$0.0013 * X^{0.525} + 0.0225 * X^{0.428} + 0.039 * X^{0.356}$$
$$+0.252 * X^{0.234} + 0.157 * X^{0.128}$$
$$+0.157 * X^{0.0223} + 0.37$$

Suppose that the maximum normalized weight of the combined term "computer algorithm" $mnw_{12} = 0.825$ which is greater than mw_1 and mw_2. By multiplying the above polynomials, the probability that a document has a total normalized weight (associated with these two terms) of mnw_{12} or higher is $3.878 * 10^{-5}$. This probability is based on the assumption that the two terms were independent. The actual probability is $1/N = 0.0013$, where $N = 761$ in this example. Since the estimated probability and the actual probability differ substantially, the two terms should be combined. ■

In general, $O(m^2)$ term pairs need to be tested for possible combination, where m is the number of terms. When m is large, the testing process may become too time consuming. In order that the process can be easily carried out, we restrict the terms to be query terms (i.e., terms appeared in previously submitted queries) and each pair of terms to be in adjacent locations in a query. The latter condition is to simulate phrases since the components of a phrase are usually in adjacent locations.

Given a query, we need to estimate the distribution of the similarities of the query with the documents in the database, while taking into consideration that certain terms in the query may be combined. We shall restrict a combined term to contain two individual terms only. It is essential to decide for a given term of the query whether it is to be combined, and if the term is to be combined, which term should be combined with it. Specifically, consider three adjacent terms t_i, followed by t_j and then followed by t_k in a query. If term t_i has been combined with its preceding term, then it will not be combined with term t_j (because a phrase usually consists of two words and it is simpler to recognize phrases containing two words than phrases containing three or more words); otherwise, check if the combined term t_{ij} exists. If the combined term t_{ij} exists, then check if the combined term t_{jk} exists. If both combined terms exist, then compare the differences d_{ij} and d_{jk}. The larger difference indicates which term should be combined with term t_j for this query. For example, if d_{jk} is larger than d_{ij}, then term t_j is combined with t_k and the distribution of the combined term should be used to estimate the distribution of the similarities of the documents with this query. If only one of the combined term exists, then that combined term will be used. If none of the two combined terms exists, then term t_j is not combined with any term.

5 A Sufficient Condition for Databases to Be Ranked Optimally

In this section, we present a method to retrieve the n most similar documents to a given query across multiple databases. Based on this method, we prove a sufficient condition for ranking databases so that the n most similar documents will be retrieved.

Notice that our database usefulness estimation methods (see Section 4) can estimate the global similarity of the most similar document in each local database. Suppose we rank the databases in descending order of the estimated similarity of the most similar document in each database. More precisely, let sim_i be the estimated similarity of the most similar document in database D_i. If $sim_1 \geq sim_2 \geq \cdots \geq sim_k$, then the databases are ranked in the order $D_1, D_2, \ldots, D_k$.

We now present an algorithm, *OptDocRetrv*, to retrieve the n most similar documents to a given query from these databases. The basic idea is that we retrieve documents from the databases in the order $D_1, D_2, \ldots, D_k$ until the n most similar documents from these databases are obtained. 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 obtain all documents whose actual global similarities are greater than or equal to *m-asim*. If n or more documents have been obtained, then the algorithm can be terminated. Otherwise, the next database in the given order, namely D_{s+1}, will be considered and its most similar document will be retrieved. 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 obtain all documents from these $s+1$ databases whose actual global similarities are greater than or equal to this threshold.

Algorithm OptDocRetrv

1. retrieve the most similar document from database D_1 and let $asim_1$ be its actual global similarity;
2. *m-asim* := $asim_1$;
3. j := 1; /* D_j is the database whose most similar document has similarity equal to the current *m-asim*. */
4. i := 2;

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

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

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

      i. retrieve documents from databases $D_1, \ldots, D_{i-1}$ such that each of these documents has actual global similarity $\geq asim_i$;

      ii. *m-asim* := $asim_i$ and $j := i$;

      }

      else { retrieve from database D_i all documents having actual global similarities $\geq$ *m-asim*;

      }

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

   }
5. Sort all the retrieved documents in descending order of their actual global similarities and return the top n documents.

This algorithm has the following properties.

Proposition 2: For a given query, suppose the set of the n most similar documents S is unique (note that in general S may not be unique due to identical similarities of different documents) and databases $D_1, D_2, \ldots, D_k$ contain the documents in S. If a database selection method ranks the databases $D_1, D_2, \ldots, D_k$ higher than other databases, then the n most similar documents to the query will be retrieved.

Proof: Without loss of generality, we assume that each of the k databases contains at least one of the n documents in S. Let *m-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-asim*, $1 \leq j \leq k$. Let the actual global similarity of the n-th most similar document be *minsim*. Then, *m-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-asim* = *minsim*, then all documents in S would have been retrieved based on the definition of *minsim*. If *minsim* < *m-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. By step 4 of algorithm *OptDocRetrv*, we know that the either the top k or $k+1$ databases will be examined depending whether *m-asim* = *minsim* is true. Then the n most similar documents will be retrieved if the top k databases are correctly identified. ∎

Note that if the set 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 is a rather surprising result. It 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. In other words, there is no need to estimate accurately the number of most similar documents in each database, as long as the databases are ranked properly. In practice, if a database selection method can produce reasonably accurate but not necessarily perfect estimates, then ranking databases based on the estimates should give good results. Also, from the proof of Proposition 2, we can see that 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 one will be examined by the algorithm.

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 either the subrange-based method or the combined-term method, when used in conjunction with the document retrieval strategy given in algorithm *OptDocRetrv*.

Proof: For our subrange-based method and the combined-term method, there is a maximum normalized weight associated with each term for 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. 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: At the end of step 4 of algorithm *OptDocRetrv*, n or slightly more than n documents will be retrieved. When another database, say D_p, needs to be examined, the number of documents retrieved so far must be less than n and the additional documents to be retrieved are those from D_p or in a previously examined database whose actual global similarities are greater than or equal to the actual global similarity of the most similar document in database D_p. It is likely that the additional number of documents to be retrieved in step 4 is less than n.

Observation 2: An important assumption implicit in the algorithm *OptDocRetrv* is that the most similar document within each database with respect to the global similarity be obtained. 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 (i.e., we can employ a database selection algorithm to retrieve c times the estimated number of most similar documents from each database for some positive constant c) 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 4 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. Then, the modified query is $q' = (q_1 * a_1/b_1,\ q_2 * a_2/b_2)$. Then, 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 modelled 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 3: In algorithm *OptDocRetrv*, we examine one database at a time. Another alternative is as follows. Apply a database selection method such as the subrange-based method or the combined-term method to identify the databases which are likely to contain the n most similar documents. 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) retrieve 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 retrieved, then terminate; otherwise, another database is examined. In this way, there will be fewer iterations and the efficiency of the algorithm may be improved.

Although the document retrieval strategy described in algorithm *OptDocRetrv* has the nice property as reflected in Propositions 2 and 3, it also has a serious drawback. The problem arises when a database is incorrectly identified to contain one or more of the n most similar documents but in reality the actual global similarity of its most similar document is significantly below that of the n-th most similar document across all databases. This is illustrated by the following example. Suppose databases D_1, D_2, D_3 and D_4 contain the n most similar documents but a database selection method ranks the databases in the order $D_1, D_5, D_2, D_3, D_4, \ldots$. Suppose the actual global similarity of the most similar document in D_5 is much smaller than that of the n-th most similar document and there are many documents in D_1 whose actual global similarities are higher than the most similar document in D_5. Then, these documents in D_1 will be retrieved, preventing the most similar documents from D_2, D_3 and D_4 from being retrieved. This problem drawback can be avoided to some extent by the modified algorithm in Observation 3, in which a number of databases are examined together. For the experimental results presented in the next section, the remedy given above is not used and yet the results are very promising.

6 Experimental results

In this section, we report some experimental results. 15 databases are used in our experiments. These databases are formed from articles posed to 52 different newsgroups in the Internet. These articles were collected at Stanford University [6]. Each newsgroup that contains more than 500 articles forms a separate database. Smaller newsgroups are merged to produce larger databases. Table 1 shows for each database its size and the number of distinct terms appearing in its collection of documents. 1,000 queries by real users are used in our experiments. These queries were also collected at Stanford University [6].

database	#documents	#distinct terms
1	761	16065
2	1014	29780
3	705	15400
4	682	15588
5	661	12737
6	622	18858
7	526	10986
8	555	10245
9	629	11584
10	588	11502
11	558	15821
12	526	12168
13	607	23338
14	648	11787
15	564	15023

Table 1: Databases Used in Experiments

We first compare the performance of the *high-correlation method* [6, 7] and the *combined-term method* relative to the *ideal method* which determines the databases containing the n most similar documents optimally. For each query, the *ideal method* is implemented by computing the actual global similarity of each document directly and the similarities are then used to rank the databases. Clearly, the *ideal method* is not applicable in read metasearch engines. It is used for comparison purpose only. The *high-correlation method* is chosen because it has similar capabilities with the *combined-term method*, i.e., they both can rank databases and estimate the number of documents in each local database having similarities greater than or equal to a given threshold.

The *high-correlation method* estimates the number of documents in a local database having similarities greater than or equal to a given threshold based on the following assumption: For any given database, if query term t_j appears in at least as many documents as query term t_k, then every document containing term t_k also contains term t_j. Let $f_1 \le f_2 \le ... \le f_n$ be the document frequencies of query terms $t_1, t_2, ..., t_n$, respectively. By the high-correlation assumption, each document that contains t_k also contains $t_j, j > k$. Therefore, these f_1 documents containing t_1 has the highest similarity which is: $\sum_{i=1}^{n} q_i * (W_i / f_i)$, where q_i is the weight of term t_i in the query, W_i is the sum of the weights of t_i over all documents in the database. Similarly, the $f_2 - f_1$ documents containing t_2 but not t_1 has similarity $\sum_{i=2}^{n} q_i * (W_i / f_i)$. In general, the $f_k - f_{k-1}$ documents that contain t_k but not t_{k-1} has similarity $\sum_{i=k}^{n} q_i * (W_i / f_i)$. Let p be the largest number to satisfy $\sum_{i=p}^{n} q_i * (W_i / f_i) > T$, where T is the threshold. Then the number of documents in the database having similarities greater than the threshold T can be estimated to be f_p.

In our earlier papers [20, 21], we have compared our estimation methods with the high-correlation method in terms of the estimation accuracy for individual databases only. Here, the comparison is based on the retrieval of the n most similar documents from multiple databases.

The following two measures are used in the comparison: (1) the percentage of correctly identified databases (i.e., the databases containing the n most similar documents); and (2) the percentage of correctly identified documents (i.e., the n most similar documents.)

One of the experiments we performed is designed as follows. First, the *ideal method* is used for each query such that a set of M databases containing the n most similar documents is obtained, for some positive integer M. This number M is then used by both the *combined-term method* and the *high-correlation method* to determine which databases should be chosen. In other words, each of these two estimation methods will use their top M databases to retrieve the n most similar documents. Clearly, the choices of databases by the two methods may differ. For each method and each database chosen by it, it will retrieve $c * est\_number$ documents from the database, where est_number is the number of documents in the database which are estimated to be among the n most similar documents and c is a parameter which varies from 1.5 to 4. If the estimated number of documents is very precise, then it is sufficient to retrieve from each chosen database exactly est_number documents, as the sum of these estimated numbers over the chosen databases should be n or slightly above it. Since the estimated number may not be that accurate, we try other values of $c > 1$.

Clearly, the *ideal method* identifies 100% of the correct databases containing the n most similar documents and 100% of the desired documents. Since the *combined-term method* and the *high-correlation method* identify the same number (but not necessarily the same set) of databases, the percentages of correctly identified databases and documents are usually below 100%. Tables 2 and 3 show that the *combined-term method* retrieves 83% to 93% of correctly identified databases versus 51% to 64% of correctly identified databases by the *high-correlation method*, when the numbers of most similar documents are 5, 10, 20 and 30. In these tables, *ciDb* denotes the percentage of correctly identified databases and *ciDoc* denotes the percentage of correctly identified documents. The average improvement of the *combined-term method* over the *high-correlation method* in identifying the databases is 55.5%. The improvements range from 45.3% to 61.9% with the larger improvements for smaller values of n. Tables 2 and 3 also show that the results of correctly retrieving the most similar documents for c = 1.5, 2, 3 and 4, and for the different values of n described above. The average improvement of the *combined-term method* over the *high-correlation method* in identifying documents is 21%. The improvements range from 15.9% to 29.2% with the larger improvements for smaller values of n and c.

		ciDoc			
n	ciDb	$c = 1.5$	$c = 2$	$c = 3$	$c = 4$
5	51.5%	53.6%	54.6%	55.1%	55.3%
10	54.1%	58.2%	59.4%	60.6%	60.6%
20	59.8%	64.0%	65.5%	66.8%	67.5%
30	64.3%	68.0%	69.8%	71.2%	72.0%

Table 2: High-correlation Method

Our second set of experiments consists of applying the *combined-term method* with the retrieval algorithm *OptDocRetrv* to identify the desired databases and the desired documents. Unlike the first set of experiments, the number of desired databases for a given query is not known to the algorithm. The *combined-term method* ranks the databases using the estimated similarity of the most similar document in each database and then invokes the retrieval strategy to retrieve documents in the ranked order of the databases. Table 4 shows the results according to the

		ciDoc			
n	ciDb	$c = 1.5$	$c = 2$	$c = 3$	$c = 4$
5	83.3%	69.3%	69.9%	70.5%	71.0%
10	87.7%	72.5%	73.5%	74.4%	75.0%
20	91.3%	75.4%	76.9%	78.4%	79.3%
30	93.4%	79.1%	80.9%	82.6%	83.5%

Table 3: Combined-term Method

two measures ciDb and ciDoc for $n = 5$, 10, 20 and 30. It is shown that the algorithm identifies 85% to 91% of the desired databases and 88% to 95% of the desired documents. As the number of the most similar documents to be retrieved increases, the percentages of desired databases and desired documents also increase. This is not surprising because when n reaches the number of documents in all databases, both percentages should be 100%.

n	ciDb	ciDoc
5	85.48%	88.12%
10	86.15%	90.02%
20	89.41%	93.59%
30	91.63%	95.73%

Table 4: Experimental Results Using Retrieval Algorithm *OptDocRetrv*

7 Conclusion

In this paper, we developed an algorithm to retrieve the n most similar documents with respect to a given query from a collection of databases. A sufficient condition for our algorithm to retrieve all the desired documents is that those databases containing the desired documents are ranked higher than other databases. The rank of a database is based on the similarity of the most similar document in the database with respect to the query. Our algorithm does not require us to have an accurate estimate of the number of most similar documents in each database. The experimental results show that the *combined-term method* when used in conjunction with the proposed retrieval strategy is promising. When the number of desired documents was 5, we retrieved on the average about 88% of them. For all other cases, we retrieved on the average more than 90% of the desired documents. However, the experimental results are limited in scope. We will perform many more experiments with much larger collections and many more queries in the near future.

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

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

[5] 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.

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

[7] 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 [6].)

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

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

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

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

[12] 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.

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

[14] 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).

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

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

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

[18] K.L. Liu, C. Yu, W. Meng, W. Wu and N. Rishe. *A Statistical Method for Estimating the Usefulness of Text Databases.* Technical Report. Department of EECS, University of Illinois at Chicago, 1998.

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

[20] 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.

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

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

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

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

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

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

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

[28] 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.

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

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

[31] 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.

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

[33] B. Yuwono, and D. Lee. *Server Ranking for Distributed Text Resource Systems on the Internet.* 5th International Conference On Database Systems For Advanced Applications (DASFAA'97), Melbourne, Australia, April 1997, pp. 391-400.