Efficient and Effective Metasearch for Text Databases Incorporating Linkages among Documents

January 16, 2001

Abstract

Linkages among documents have a significant impact on the importance of documents, as it can be argued that important documents are pointed to by many documents or by other important documents. Metasearch engines can be used to facilitate ordinary users for retrieving information from multiple local sources (text databases). There is a search engine associated with each database. In a large-scale metasearch engine, the contents of each local database is represented by a representative. Each user query is evaluated against the set of representatives of all databases in order to determine the appropriate databases (search engines) to search (invoke). In previous work, the linkage information between documents has not been utilized in determining the appropriate databases to search. In this paper, such information is employed to determine the degree of relevance of a document with respect to a given query. This information is then stored in each database representative to facilitate the selection of databases for each given query. We establish a necessary and sufficient condition to rank databases optimally, while incorporating the linkage information. A method is provided to estimate the desired quantities stated in the necessary and sufficient condition. The estimation method runs in time linearly proportional to the number of query terms. Experimental results are provided to demonstrate the high retrieval effectiveness of the method.

Keywords: Metasearch, Distributed Collection, linkages among documents, Information Retrieval.

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* 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 quickly the documents that contain the term.

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. From a different perspective, as more and more data are put on the Web at faster paces, the coverage of the Web by any single search engine has been steadily decreasing. A solution to these problems 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 *representatives* which provide approximate contents of its underlying search engines in order 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. 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 WAIS [10], ALIWEB [14], gGlOSS [6], SavvySearch [8], ProFusion [5] and D-WISE [26]. The problem of identifying potentially useful databases to search is known as the *database selection problem*. Database selection is done by comparing each query with all database representatives. If a user only wants the m most desired documents across all local databases, for some positive integer m, then the m documents to be retrieved from the identified databases need to be specified and retrieved. This is the *collection fusion problem*.

In this paper, we present an integrated solution to the database selection problem and the collection fusion problem, while taking into consideration the linkages among documents. In the Web environment, documents (web pages) are linked by pointers. These linkages can indicate the degrees of importance of documents. It may be argued that an important document is pointed to by many documents or by important documents. For example, the home page of IBM is an important document, as there are numerous documents on the Web pointing to it. Consider a query which consists of the single term "IBM". There could be thousands of documents in the Internet having this term. However, it is likely that the user is interested in the home page of IBM. One way to retrieve that home page is to recognize that among all the documents containing the term "IBM", the home page of IBM is the most important one due to the numerous links pointing to it. This phenomenon has been utilized in [17, 13] for retrieving documents in a centralized environment. In this paper, we generalize the use of linkages in distributed text database environments to tackle the database selection problem and the collection fusion problem. We believe that this is the first time that the linkage information is utilized in distributed database environments involving a metasearch engine. Our techniques for retrieval in a distributed environment yield retrieval performance that is essentially the same as that of a centralized database containing all the documents.

The rest of the paper is organized as follows. In Section 2, we incorporate the linkage information into a similarity function so that the degree of relevance of a document to a query is determined by both its similarity to the query as well as its degree of importance. In Section 3, we sketch our solutions to the database selection problem and the collection fusion problem. In Section 4, we describe the construction of the representative of a database, which indicates approximately the contents of the database. More importantly, the representative permits the degree of relevance of the most relevant document in the database to be estimated. This estimated quantity is central to the database selection problem and the collection fusion problem. An estimation process is presented. In Section 5, experimental results are provided. Very good performance is obtained for the integrated solution of the database selection and the collection fusion problems. In Section 6, we describe how to obtain the value of a parameter that represents the weight of the similarity of a document to a query relative to the degree of importance of the document in determining the degree of relevance of the document to the query. In Section 7, we provide the experimental results obtained using a collection of documents downloaded from the Internet. The results show that our techniques for retrieving documents in a distributed environment essentially yield the same retrieval effectiveness as that of a centralized database containing the same set of documents. The conclusion is given in Section 8.

1.1 Related Works

In the last several years, a large number of research papers on issues related to metasearch engines or distributed collections have been published. In this subsection, we first identify the critical difference between our work here and existing works. It is followed by other differences which are individually identified. Only the most closely related differences are presented. Please see [30] for a more comprehensive review of other work in this area.

1. We are not aware of any solution utilizing the linkage information among documents in solving the database selection problem and the collection fusion problem in a metasearch engine environment, although such information has been utilized [17, 13] in determining the importance of documents and in their retrieval in a single search engine environment. Thus, this is the first time that such information is utilized in a distributed text database environment.

2. Our earlier work [33] utilizes the similarity of the most similar document in each database to rank databases optimally. This optimal way to rank databases is generalized in this paper to incorporate the linkage information among documents. In order that the linkage information is utilized properly,

the database representatives as well as the procedure to estimate the similarity of the most similar document in each database need to be modified.

3. The gGlOSS project [6] is similar in the sense that it ranks databases according to some measure. However, there is no necessary and sufficient condition for optimal ranking of databases; there is no precise algorithm for determining which documents from which databases are to be returned.

4. A necessary and sufficient condition for ranking databases optimally was given in [11]. However, [11] considered only the databases and queries that are for structured data. In contrast, unstructured text data is considered in this paper. In [1], a theoretical framework was provided for achieving *optimal* results in a distributed environment. Recent experimental results reported in [2] show that if the number of documents retrieved is comparable to the number of databases, then good retrieval effectiveness can be achieved; otherwise, there is substantial deterioration in performance. We show good experimental results in both situations [32]. Our theory differs from that given in [1] substantially.

5. In [23], experimental results were given to demonstrate that it was possible to retrieve documents in distributed environments with essentially the same effectiveness as if all data were in one site. However, the results depended on the existence of a *training collection* which have similar coverage of subject matters and terms as the collection of databases to be searched. Upon receiving a query, the training collection is searched, terms are extracted and then added to the query before retrieval of documents from the actual collection takes place. In the Internet environment where data are highly heterogeneous, it is unclear whether such a training collection can in fact be constructed. Even if such a collection can be constructed, the storage penalty could be very high in order to accommodate the heterogeneity. In [24], it was shown that by properly clustering documents, it was possible to retrieve documents in distributed environments with essentially the same effectiveness as in a centralized environment. However, in the Internet environment, it is not clear whether it is feasible to cluster large collections and to perform re-clustering for dynamic changes. Our technique does not require any clustering of documents. In addition, linkage information was not utilized in [24, 23].

6. In a recent paper [18], it is argued that distributed retrieval can yield better results than centralized retrieval. However, no algorithm is provided to perform database selection. Our paper demonstrates that this is possible using a practical database selection algorithm.

2 Incorporating Linkage Information into a Similarity Function

A query in this paper is simply a set of words submitted by a user. It is transformed into a vector of *terms* with *weights* [19], where a term is essentially a content word and the dimension of the vector is the number of all distinct terms. When a term appears in a query, the component of the query vector corresponding to the term, which is the *term weight*, is positive; if it is absent, the corresponding term weight is zero. The weight of a term usually depends on the number of occurrences of the term in the query (relative to the total number of occurrences of all terms in the query) [19, 25]. This is the term frequency weight. The weight of a term may also depend on the number of documents having the term relative to the total number of documents in the database. The weight of a term based on such information is called the inverse document frequency weight [19, 25]. A document is similarly transformed into a vector with weights. The similarity between a query and a document can be measured by the dot product of their respective 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 [19, 25]. For the sake of concreteness, we shall use in this paper the version of the *Cosine* function [22] where the query uses the product of the inverse document frequency weight and the term frequency weight and the document uses the term frequency weight only. Other similarity functions, see for example [21], are also possible.

We first define a function R which assigns the *degree of relevance* of a document d with respect to a query q based on two factors: one based on the similarity between the document d and the query q, and the other based on the *rank* (degree of importance) of the document. The rank of a document in a Web environment usually depends on the linkages between the document and other documents. For example, if a document d is linked to by many documents and/or by important documents, then document d is an important document. Therefore, d will have a high rank. This definition is recursive and an algorithm is given in [17] to compute the ranks of documents. The following function incorporates both similarity and rank.

$$R(q,d) = \begin{cases} w * sim(q,d) + (1-w) * NRank(d), & \text{if } sim(q,d) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $sim()$ is a similarity function such as the *Cosine* function, $NRank(d)$ is the normalized rank of document d and w is a parameter between 0 and 1. In order to avoid the retrieval of very important documents (i.e., documents with high normalized ranks) which are unrelated to the query, the degree of relevance of a document is greater than zero only if its similarity with the query is greater than zero. The normalized rank of a document can be obtained from the rank computed in [17] by dividing it by the maximum rank of all documents in all databases, yielding a value between 0 and 1. The higher the normalized rank a document has, the more important it is. When it takes on the value 1, it is the most important document. The intuition for incorporating rank is that if two documents have the same or about the same similarity with a query, then a user is likely to prefer the more important document, i.e., the document with a higher rank. We assume that the normalized ranks have been pre-computed and we are interested in finding the m most relevant documents, i.e., the m documents with the highest degrees of relevance with respect to a given query as defined by formula (1) in a distributed text database environment. This involves determining the databases containing these m most relevant documents and determining which documents from these databases need to be retrieved and transmitted to the user. In the next section, we shall present our solution to the database selection problem and the collection fusion problem.

3 Two-Level Architecture of Metasearch

In this architecture, the highest level (the root node) contains the representative for the "global database". The global database which logically contains all documents from all local databases does not exist physically. (Recall that all documents searchable by a search engine form a local database. Although we call it a "local" database, it may contain documents from multiple locations.) The representative for the global database contains all terms which appear in any of the databases and for each such term, it stores the number of documents containing the term (i.e., the global document frequency). This permits the global inverse document frequency weight of each query term to be computed. There is only one additional level in the hierarchy. This level contains a representative for each local database. The representative of each local database will be defined precisely in the next section. When a user query is submitted, it is processed first by the metasearch engine against all these database representatives. The databases are then ranked. Finally, the metasearch engine invokes a subset of the search engines and co-ordinates the retrieval and merging of documents from individual selected search engines.

3.1 Optimal Ranking of Databases

We assume that a user is interested in retrieving the m most relevant documents to his/her query for a given m.

Definition 3.1 *A set of databases is said to be optimally ranked in the order $[D_1, D_2, ..., D_p]$ with respect to a given query q if for every positive integer m, there exists a k such that $D_1, D_2, ..., D_k$ contain the m most relevant documents and each D_i, $1 \leq i \leq k$, contains at least one of the m most relevant documents.*

A necessary and sufficient condition to rank databases optimally is as follows. For ease of presentation, we shall assume that all degrees of relevance of the documents with the query are distinct so that the set of the m most relevant documents to the query is unique.

Proposition 3.1 *Databases $[D_1, D_2, \ldots, D_p]$ are ranked optimally if and only if the degree of relevance of the most relevant document in D_i is larger than that of the most relevant document in D_j, if $i < j$.*

Proof:

Sufficiency: Let R_i be the degree of relevance of the most relevant document in database D_i. Suppose $R_1 > R_2 > ... > R_p$. We need to show that $[D_1, D_2, ..., D_p]$ is an optimal order. We establish by induction that for any given m, there exists a k such that $D_1, D_2, ..., D_k$ are the only databases containing the m most relevant documents with each of them containing at least one such document.

For $m = 1$, since $R_1 > R_2 > ... > R_p$, database D_1 contains the most relevant document. Thus, $k = 1$ is found for $m = 1$.

For $m = i$, suppose $D_1, D_2, ..., D_s$ contain the i most relevant documents with each of them containing at least one of the i most relevant documents. When $m = i+1$, consider the $(i+1)$-th most relevant 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 relevant documents and $k = s$. In the latter case, the $(i+1)$-th most relevant document must appear in D_{s+1} because $R_{s+1} > R_{s+2} > ... > R_p$. Thus, for the latter case, $k = s+1$.

Necessity: Suppose the optimal rank order of the databases is $[D_1, D_2, ..., D_p]$. We now show that $R_1 > R_2 > ... > R_p$. When $m = 1$, the most relevant document is in database D_1. Thus, $R_1 > R_j$, $2 \leq j \leq p$. Let m be increased to i_1 so that the most relevant $i_1 - 1$ documents appear in database D_1 and the i_1-th most relevant document appears in another database D. This i_1-th most relevant document must be the most relevant document in D and because $D_1, D_2, ..., D_p$ are optimally ranked, the database D must be D_2. Thus, $R_2 > R_j$, $3 \leq j \leq p$. Let m be increased from i_1 to i_2 so that the i_1-th to (i_2-1)-th most relevant documents appear in database D_1 or database D_2 and the i_2-th most relevant document appears in another database D$'$. Again by the optimal rank ordering of $[D_1, D_2, ..., D_p]$, database D$'$ must be D_3 and hence, $R_3 > R_j$, $4 \leq j \leq p$. By repeatedly increasing m in the manner described above, we obtain $R_4 > R_j$, $5 \leq j \leq p$, ..., $R_{p-1} > R_p$. By combining all these derived inequalities, we obtain $R_1 > R_2 > ... > R_p$. ■

Example 3.1 Suppose there are 4 databases. For a query q, suppose the degrees of relevance of the most relevant documents in databases D_1, D_2, D_3 and D_4 are 0.8, 0.5, 0.7 and 0.2, respectively. Then the databases should be ranked $[D_1, D_3, D_2, D_4]$. ■

This result applies to any similarity function as well as any function which assigns degrees of importance to documents. It is also independent of data types. Thus, it can be applied to any type of databases, including text databases, image databases, video databases and audio databases. The necessary and sufficient condition to rank databases optimally is also independent of the number of documents desired by the user.

In [33], we ranked databases in descending order of the similarity of the most similar document in each database. The result turns out to be generalizable to capture the degrees of relevance of documents.

3.2 Estimation of the Degree of Relevance of the Most Relevant Document in Each Database

In the last subsection, we showed that the best way to rank databases is based on the degree of relevance of the most relevant document in each database. Unfortunately, it is not practical to retrieve the most relevant document from each database, obtain its degree of relevance and then perform the ranking of the databases. However, it is possible to estimate the degree of relevance of the most relevant document in each database, using an appropriate choice of a database representative. This will be given in Section 4.

3.3 Coordinate the Retrieval of Documents from Different Search Engines

Suppose the databases have been ranked in the order $[D_1, \ldots, D_p]$, we now briefly review an algorithm [31] which determines (1) the value of k such that the first k databases are searched, and (2) which documents from these k databases should be used to form the list of m documents to be returned to the user. Suppose the first s databases have been invoked from the metasearch engine. Each of these search engines returns the degree of relevance of its most relevant document to the metasearch engine which then computes the minimum of these s values. Each of the s search engines returns documents to the metasearch engine whose degrees of relevance are greater than or equal to the minimum. (If the number of documents in a single search engine whose degrees of relevance are greater than or equal to the minimum value is greater than m, then that search engine returns the m documents with the largest degrees of relevance. The remaining ones will not be useful as the user wants only m documents with the largest degrees of relevance.) If an accumulative total of $m + add\_doc$, where add_doc is some constant which can be used as a tradeoff between effectiveness and efficiency of retrieval, or more documents have been returned from multiple search engines to the metasearch engine, then these documents are sorted in descending order of degree of relevance and the first m documents are returned to the user. Otherwise, the next database in the order will be invoked and the process is repeated until $m + add\_doc$ documents are returned to the metasearch engine. It can be shown that if the databases have been ranked optimally (with the databases containing the desired documents ahead of other databases) for a given query, then this algorithm will retrieve all the m most relevant documents with respect to the query. (The proof is essentially the same as that given in [31], except we replace the similarity of a document by its degree of relevance.) When $add\_doc > 0$, more documents will be received by the metasearch engine (in comparison to the case add_doc =0). As a result, it can select the m best documents from a larger set of retrieved documents, resulting in better retrieval effectiveness. However, efficiency will decrease.

4 Estimate the Degree of Relevance of the Most Relevant Document in Each Database

We first define the representative of a database so that the degree of relevance of the most relevant document can be estimated. It consists of all terms which appear in any document in the database. For each term, three quantities are kept. They are the *maximum integrated normalized weight*, the *average normalized weight* and the *normalized rank* of the document where the maximum integrated normalized weight is attained. They are defined as follows. The normalized weight of the ith term of a document $d = (g_1, ..., g_i, ..., g_n)$ is $d_i = g_i/|d|$, where $|d|$ is the length or norm of document vector d. The average normalized weight of the ith term, aw_i, is simply the average value of the normalized weights over all documents in the database, including documents not containing the term. The integrated normalized weight of the ith term in the document d is $w * d_i + (1 - w) * r$, if d_i is positive; otherwise, the integrated normalized weight is defined to be 0. In the above expression, r is the normalized rank of document d and w is the weight of similarity relative to normalized rank in determining the degree of relevance of a document (see Formula (1)). The maximum integrated normalized weight of the term, miw_i, is the maximum value of the integrated normalized weights over all documents in the database. Suppose the maximum value is attained by a document with normalized rank r_i. Then, for term t_i, the three quantities (miw_i, aw_i, r_i) are kept.

Consider a query $q = (q_1, ..., q_i, ..., q_n)$, where both term frequency information and inverse document frequency information have been integrated into the query vector and the components have been normalized.

Consider a document $d = (d_1, ..., d_i, ..., d_n)$, where d_i is the normalized weight of term t_i and r is the normalized rank of d. Its degree of relevance with respect to query q is $w * sim(q, d) + (1 - w) * r = w*d_1*q_1+w*\sum_{k=2}^{n} d_k*q_k+(1-w)*r = w*d_1*q_1+q_1*(1-w)*r+w*\sum_{k=2}^{n} d_k*q_k+(1-w)*r-q_1*(1-w)*r = q_1 * (w * d_1 + (1 - w) * r) + w * \sum_{k=2}^{n} d_k * q_k + (1 - w) * r * (1 - q_1)$.

This document d may have the maximum integrated normalized weight for the term t_1 and the average normalized weights for the other terms. In that case, the above expression becomes $q_1*miw_1+w*\sum_{k=2}^{n} aw_k * q_k + (1 - w) * r * (1 - q_1)$. Finally, the rank of d is the normalized rank of the document where the maximum integrated normalized weight (miw_1) for the term t_1 is attained. As described earlier, this rank is stored in the database representative. Let it be denoted by r_1 as it is associated with term t_1.

In general, we estimate the degree of relevance of the most relevant document in the database by assuming that the document contains one of the query terms having the maximum integrated normalized weight. Thus, its degree of relevance may be estimated by the following expression

$$\max_i \{q_i * miw_i + w * \sum_{k=1, k \neq i}^{n} aw_k * q_k + (1 - w) * r_i * (1 - q_i)\} \tag{2}$$

where the maximum is over all query terms. It is easy to see that the estimation takes time linearly proportional to the number of query terms.

One important property of this method is that it guarantees optimal retrieval for single term queries which are submitted frequently in the Internet environment. (A study in [9] indicates that single-term queries account for about 30% of all Internet queries.) The reason is that this method estimates the degree of relevance of the most relevant document for any given single-term query exactly. As a result, the necessary and sufficient condition for ranking the databases optimally is satisfied. For optimally ranked databases, the algorithm to co-ordinate the retrieval of documents from multiple databases guarantees optimal retrieval.

Lemma 4.1 *For any single term query, the estimate given by the above estimation method for the degree of relevance of the most relevant document in a database is exact.*

Proof: For a single term query, say $q = (1, 0, ..., 0)$, the estimate given by the above method $= miw_1$. This can be obtained by setting $i = 1, q_i = 1$ and $q_k = 0$ for $k \neq 1$ in expression (2). A document $d = (d_1, ..., d_n)$ having that term has degree of relevance $f = w * d_1 + (1 - w) * r$, where r is the rank of the document d. By the definition of the maximum integrated normalized rank of term t_1, $f \leq miw_1$. Thus, since miw_1 is actually achieved by a document in the database, it is the degree of relevance of the most relevant document in the database. ■

Proposition 4.1 *For any single term query, optimal retrieval of documents for the query using this method and the co-ordination algorithm is guaranteed.*

Proof: By Lemma 4.1, the degree of relevance of the most relevant document in each database is estimated exactly. Using these estimates for each database guarantees optimal ranking of databases, since the necessary and sufficient condition for optimal ranking is satisfied. Finally, the co-ordination algorithm guarantees optimal retrieval of documents, if databases are optimally ranked. ■

Our estimation process essentially assumes that terms are distributedly independently in the documents in each database. Word combinations, for example phrases usually do not satisfy the independent distribution assumption. In order to avoid the use of independence assumptions for certain word combinations, we construct combined terms from individual terms and store the statistics associated with the combined terms in addition to those associated with individual terms.

Suppose the ith term and the jth term are combined into a term. We can then define the integrated normalized weight of the combined term to be $w * (idf_i * d_i + idf_j * d_j)/idf_{ij} + (1 - w) * r$, if d_i and d_j are positive; otherwise, the integrated normalized weight of the combined term is 0. (In practice, we assume that that there is a precise process to recognize phrases. See for example [15].) In the above expression idf_i is the inverse document frequency weight of the ith-term, while $idf_{i,j}$ is $\sqrt{idf_i^2 + idf_j^2}$. The maximum integrated normalized weight of the combined-term, miw_{ij}, is the maximum value of the integrated normalized weight of the combined term over all documents in the database. In the database representative, in addition to the three quantities associated with each individual term, we store the same three quantities (the maximum integrated normalized weight, the normalized rank of the document which achieves the maximum integrated normalized weight and the average normalized weight) for each combined term. It is easy to see that for a two term query containing both term i and term j, the document having the maximum integrated normalized weight in a database is the one having the largest degree of relevance with the query in that database. Since the maximum integrated normalized weight is stored in the database representative, the degree of relevance of the most relevant document in the database will be obtained exactly. This allows two-term queries to be processed optimally. Together with Proposition 4.1, a query involving a single word or two words can be processed optimally. Since on the average each Internet query has 2.2 terms [12], this implies for vast majority of Internet queries, our method can achieve the optimal retrieval.

5 Experimental Results

In this section, we report some experimental results. Two sets of data and queries with different characteristics are utilized. The first set of data consists of 15 databases. These databases are formed from articles posted 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 the number of documents in each database in the first data set. There are altogether 6,597 queries submitted by real users. Both the data and the queries were

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

Table 1: Databases in Data Set 1 Used in Experiments

used in the gGlOSS project [6]. 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. These are *long queries*.

The second set of data consists of a subset of TREC (Text REtrieval Conference) data which were collected by NIST (National Institute of Standards & Technology of the United States). We use the first 30 databases (20 from FBIS — Foreign Broadcast Information Service and 10 from CR — Congressional Record) and all of the 400 queries used in the first 8 TREC conferences. There are approximately 57,000 documents in these 30 databases. A typical TREC query consists of three parts (i.e., topic, description and narrative). Since real user queries are typically short, we only used the topic part of each query in our experiments. They are on the average longer than the Stanford queries. Specifically, the average numbers of words in a short query (no more than 6 words each) and in a long query (more than 6 words each) in the TREC query set are 3.29 and 12.62, respectively; the corresponding numbers for the Stanford query set are 2.3 and 10.3, respectively. Studies show that a typical Internet queries has 2.2 terms [12]. Thus, the Stanford short queries resemble typical Internet queries better.

One problem we have is that the documents in these two sets do not have linkages among themselves. In order to simulate the effect of linkages among documents on their degrees of importance, we assign normalized ranks to documents based on some distribution. We note that if normalized ranks were randomly assigned to documents, then since each database has quite a few documents, then the maximum normalized rank and the average normalized rank of each database would be close to 1 and 0.5, respectively. This would not reflect reality. Instead, for each database, we randomly generate a number, say x. Then the normalized ranks of the documents in that database will be randomly assigned in the range between 0 and x. In this way, some databases have much higher maximum normalized rank than others.

The performance measures of an algorithm to search for the m most relevant documents in a set of databases are given as follows. The first two measures provide effectiveness (quality) of retrieval while the last two measures provide efficiency of retrieval.

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

2. The percentage of the sum of the degrees of relevance of the m most relevant documents retrieved, that is, the ratio of the sum of the degrees of relevance of the m retrieved documents over the sum of the degrees of relevance of the m most relevant documents. This percentage is denoted by **per_rel_doc**. This represents the percentage of the expected number of relevant documents retrieved.

   Suppose a user is interested in retrieving the most relevant document which has degree of relevance 0.95. If the retrieval system retrieves the second most relevant document with degree of relevance 0.9 but not the most relevant document. According to the measure **cor_iden_doc**, the percentage is 0, as none of the m ($m = 1$ in this example) most relevant documents is retrieved. Using the measure **per_rel_doc**, the percentage is 0.9/0.95. Thus, this measure is more lenient than the measure **cor_iden_doc**. In information retrieval, the standard recall and precision measures are related with the number of retrieved relevant documents. They are not concerned with specific documents. In other words, if k relevant documents out of m retrieved documents are replaced by k other relevant documents, then both recall and precision remain unchanged. Thus, the measure **per_rel_doc** is more indicative of the standard recall and precision measure than the measure **cor_iden_doc**.

3. The database search effort is the ratio of the number of databases searched by the algorithm over the number of databases which contain one or more of the m most relevant documents. This ratio is denoted by **db_effort**. The ratio is usually more than 1.

4. The document search effort is the ratio of the number of documents received by the metasearch engine (see Section 3.3) over m. This is a measure of the transmission cost. This ratio is denoted by **doc_effort**.

The experimental results for short and long queries **without the use of combined terms** for the Stanford data when the parameter w in formula (1) is 0.8 are presented in Tables 2 and 3, respectively. Those for the TREC data are given in Tables 4 and 5, respectively. The reasons for choosing $w > 0.5$ are as follows: (i) the similarity between a document and a query should play a more important role in determining its degree of relevance between the document and the query; (ii) the way the normalized ranks of documents are assigned makes it possible for the normalized rank of a document to be close to 1, while in most cases, the similarity between a similar document and a query is usually no higher than 0.3 (since a document has many more terms than the number of terms in common between the document and a query).

m	cor_iden_doc	db_effort	doc_effort	per_rel_doc
5	96.1%	122.0%	135.7%	99.7%
10	97.6%	116.2%	132.2%	99.8%
20	98.2%	111.0%	123.2%	99.8%
30	98.5%	108.2%	118.9%	99.9%

Table 2: Short queries with $w = 0.8$ for Stanford data

m	cor_iden_doc	db_effort	doc_effort	per_rel_doc
5	94.7%	132.5%	150.8%	99.6%
10	95.4%	121.5%	156.0%	99.7%
20	96.7%	114.5%	163.7%	99.8%
30	97.5%	111.8%	165.3%	99.8%

Table 3: Long queries with $w = 0.8$ for Stanford data

m	cor_iden_doc	db_effort	doc_effort	per_rel_doc
5	87.8%	107.8%	123.5%	98.8%
10	91.0%	104.3%	129.7%	99.2%
20	94.0%	101.5%	123.8%	99.5%
30	96.0%	101.7%	126.0%	99.7%

Table 4: Short queries with $w = 0.8$ for TREC data

A summary of the results from Tables 2 to 5 are given as follows.

1. The method gives very good retrieval effectiveness for short queries. For the Stanford data, the percentages of the m most relevant documents retrieved range from 96% to 98%; the corresponding figures for the TREC data are from 88% to 96%. Recall that the short queries and in particular the Stanford queries resemble the Internet queries. The percentage of the number of relevant documents retrieved is more impressive; it ranges from 99.7% to 99.9% for the Stanford data; the range is from 98.8% to 99.7% for the TREC data. Thus, there is essentially little or no loss of the number of useful documents using the retrieval algorithm in the distributed environment versus the environment in which all documents are placed in one site. As the number of documents to be retrieved increases, it is usually the case that both the percentage of the most relevant documents retrieved and the percentage

m	cor_iden_doc	db_effort	doc_effort	per_rel_doc
5	84.3%	106.9%	132.4%	98.4%
10	86.8%	101.0%	156.8%	99.0%
20	90.0%	96.6%	150.1%	99.3%
30	92.3%	95.5%	153.6%	99.5%

Table 5: Long queries with $w = 0.8$ for TREC data

of the number of relevant documents retrieved increase. The reason is that as the number of databases accessed increases, the chance of missing desired databases will be reduced.

2. For long queries, the retrieval performance of the method is still very good, although there is a degradation in performance. The degradation in the percentage of the most relevant documents varies from less than 1% to 2.2% for the Stanford data; it is less than 5% for the TREC data. As the number of terms in a query increases, the estimation accuracy decreases, causing the degradation. The degradation is higher for the TREC data, because TREC queries are longer. When the percentage of the number of relevant documents retrieved is considered, there is essentially no change for the Stanford data; for the TREC data, the percentage varies from 98.4% to 99.5%, essentially giving the same performance as if all data were in one location.

3. The number of databases accessed by the methods is on the average at most 32.5% more than the ideal situation in which only databases containing the desired documents are accessed. In the tables, there are a few cases where the number of databases accessed is less than the ideal situation. The reason is that when an undesired database is accessed, the degree of relevance, say d, of the most relevant document in that database is obtained. If d is significantly less than the degree of relevance of the most relevant document in a desired database D, it will cause substantial number of documents, including some undesired documents, to be retrieved from D. When the total number of retrieved documents is m or higher, the co-ordination algorithm terminates without accessing some desired databases.

4. The number of documents transmitted from the databases to the metasearch engine by the method can on the average be up to 165.3% times the number of documents desired by the user. The reason for this behavior is, based on thoroughly analyzing the results for several queries, as follows. Suppose for a query q, the databases containing the desired documents are $\{D_1, D_2, D_3\}$ and m documents are to be retrieved. Suppose the ranking of the databases is $D_1, D_2, D_3, D_4, \ldots$. This gives optimal retrieval, guaranteeing that all the m most relevant documents to be retrieved. However, the number of documents retrieved can be very substantial, because after accessing the first 3 databases, finding the degrees of the most relevant documents in these databases, taking the minimum of these degrees and then retrieving all documents from these databases with degrees of relevance larger than or equal

to the minimum, it is possible that fewer than m documents are retrieved. As a consequence, after accessing database D_4, obtaining the degree of relevance of the most similar document in D_4 and using it to retrieve more documents from the first three databases, then a lot more documents are retrieved from these databases. However, it should be noted that in practice, the actual documents are not transmitted from the databases to the metasearch engine. Instead, only the titles of the documents and their URLs are transmitted. Thus, the transmission costs would not be high.

There is a tradeoff between efficiency and effectiveness of retrieval which can be achieved using the coordination algorithm given in Section 3.3. The results in Tables 2 to 5 were produced when the metasearch engine received at least m documents. If the number of documents received by the metasearch engine is at least $m + add\_doc$ with the m most relevant documents being returned to the user, then higher effectiveness will be achieved at the expense of more retrieved documents and more accessed databases. Tables 6 to 9 show the results when add_doc is 5, i.e., 5 more documents are to be received at the metasearch engine. It is observed from these tables that as far as measure **per_rel_doc** is concerned, close to optimal retrieval (at least 99.1% of the number of relevant documents) is achieved. If the measure to retrieve the m most relevant documents is used, close to optimal retrieval results are obtained for both the short and the long queries for the Stanford data, but there is some room for improvement for the long queries for the TREC data. The number of databases accessed is on the average at most 93.4% beyond what are required and the number of documents transmitted is on the average at most 164.3% beyond what are required. These excess high values are obtained when the method is used on long queries and $m = 5$. When $add\_doc = 5$, at least 10 documents are to be retrieved by the metasearch engine and thus doc_effort is at least 200%. As m increases and add_doc stays constant, the percentage of additional documents received at the metasearch engine decreases and thus doc_effort decreases. Although doc_effort is still reasonably high for $m = 30$, the transmission cost should not be excessive, as usually only document titles and URLs are transmitted. The number of databases accessed on the average varies from 19.5% to 93.4% higher than the ideal situation.

m	cor_iden_doc	db_effort	doc_effort	per_rel_doc
5	99.0%	171.2%	220.6%	99.9%
10	99.0%	138.2%	175.3%	99.9%
20	98.9%	121.4%	145.8%	99.9%
30	99.0%	114.8%	134.4%	99.9%

Table 6: Short queries with $w = 0.8$ for Stanford data, with 5 additional documents retrieved

The final set of results involves the use of combined terms (but with at most two terms combined into one) and for each query, 5 additional documents are retrieved (as described in the last set of experiments.) Since close to optimal results have been obtained for the Stanford data, we perform the new experiments for the TREC data only. As seen in the Tables 10 and 11, close to optimal results are also obtained for TREC

m	cor_iden_doc	db_effort	doc_effort	per_rel_doc
5	98.4%	193.4%	257.2%	99.9%
10	98.4%	154.8%	220.3%	99.9%
20	98.3%	128.5%	192.4%	99.9%
30	98.3%	119.5%	183.1%	99.9%

Table 7: Long queries with $w = 0.8$ for Stanford data, with 5 additional documents retrieved

m	cor_iden_doc	db_effort	doc_effort	per_rel_doc
5	93.4%	169.7%	240.4%	99.5%
10	93.4%	129.5%	184.7%	99.5%
20	95.4%	114.1%	158.0%	99.6%
30	96.7%	108.3%	147.3%	99.7%

Table 8: Short queries with $w = 0.8$ for TREC data, with 5 additional documents retrieved

m	cor_iden_doc	db_effort	doc_effort	per_rel_doc
5	90.6%	157.0%	264.3%	99.1%
10	89.7%	118.5%	213.5%	99.1%
20	92.3%	106.1%	183.5%	99.5%
30	92.9%	100.7%	174.3%	99.5%

Table 9: Long queries with $w = 0.8$ for TREC data, with 5 additional documents retrieved

data.

m	cor_iden_doc	db_effort	doc_effort	per_rel_doc
5	97.4%	177.3%	236.1%	99.89%
10	97.2%	135.6%	180.1%	99.85%
20	98.2%	118.6%	156.1%	99.90%
30	98.6%	112.7%	143.9%	99.93%

Table 10: Short queries for TREC data with combined terms, $w = 0.8$, 5 additional documents retrieved

m	cor_iden_doc	db_effort	doc_effort	per_rel_doc
5	94.6%	162.0%	256.3%	99.54%
10	93.3%	124.8%	207.1%	99.50%
20	94.7%	110.8%	167.9%	99.70%
30	95.6%	106.5%	171.0%	99.73%

Table 11: Long queries for TREC data with combined terms, $w = 0.8$, 5 additional documents retrieved

6 Determination of w, the weight of the similarity relative to normalized rank

In Equation (1), the degree of relevance of a document d to a query q is defined to be

$$w * sim(q, d) + (1 - w) * NRank(d)$$

if $sim(q, d) > 0$, where $sim(q, d)$ is the similarity of d to q, $NRank(d)$ is the normalized rank of d and w is a parameter between 0 and 1 that represents the weight of the similarity $sim(q, d)$ relative to the normalized rank $NRank(d)$. We now sketch how this relative weight w can be estimated in practice.

We assume that for each query Q_j, in a set of training queries $\mathcal{Q}$, the 100 most similar documents are retrieved. Among these 100 documents, the relevant documents are identified. Let d_i be the ith retrieved document. If d_i is relevant, then its degree of relevance to Q_j should be high, i.e., its degree of relevance should be bigger than some threshold T_j. Thus, the following inequality should be satisfied.

$$w * sim(Q_j, d_i) + (1 - w) * NRank(d_i) > T_j \quad (3)$$

On the other hand, if d_i is an irrelevant document, then its degree of relevance should be smaller than or equal to the threshold T_j. The corresponding inequality is as follows.

$$w * sim(Q_j, d_i) + (1 - w) * NRank(d_i) \leq T_j \quad (4)$$

From the 100 documents most similar to Q_j, 100 inequalities of the above form (3) or (4) are constructed. This is repeated for all queries in the set of training queries $\mathcal{Q}$. Altogether there are $100*|\mathcal{Q}|$ inequalities. Ideally all these inequalities are satisfied for some w and thresholds T_j, $Q_j \in \mathcal{Q}$. In practice, this may not be possible. Consider, for example, the following possible senario in which for a query, the similarity and normalized rank of an irrelevant document are higher than the similarity and normalized rank of a relevant document respectively. It is clear that the degree of relevance of the irrelevant document to the query must be always higher than that of the relevant document for any value of w. Thus, the equalities associated with these two documents can never be satisfied at the same time. As a consequence, we aim to find the w and the threshold values which will maximize the number of satisfied inequalities.

In general, in the retrieval result of a query, if the number of documents relevant to the query is small, much more emphasis is placed on retrieving all these relevant documents than rejecting the irrelevant documents. Hence, we consider a satisfied inequality associated with a relevant document $(100-r)/r$ times as important as a satisfied inequality associated with an irrelevant document, where r is the number of relevant documents. As an example, if there are 5 relevant documents for a query, then a satisfied inequality associated with a relevant document is 19 times as important as a satisfied inequality associated with an irrelevant document. (Note that if the number of relevant documents is high, the importance placed on a relevant document will be lower than that of an irrelevant document. However, this is rare. The number of retrieved documents relevant to a query is usually much lower than the number of retrieved irrelevant documents.)

Based on the above discussion, we will seek the relative weight w that will maximize the following weighted sum of the numbers of satisfied inequalities:

$$\sum_{Q_j \in \mathcal{Q}} \left(\frac{100 - r_j}{r_j} \times R_j + I_j \right)$$

where R_j is the number of satisfied inequalities among those inequalites associated with the r_j relevant documents for query Q_j and I_j is the number of satisfied inequalities among those inequalities associated with the $(100 - r_j)$ irrelevant documents.

To determine the optimal relative weight w, the following heuristics is used. We begin with the interval from 0 to 1. For each value of $w = 0, 0.1, \ldots, 1$, the weighted sum of the numbers of satisfied inequalities is computed. Among these 11 numbers, the largest sum is noted. Suppose it occurs at $w = t$. Consider the two adjacent points $t-0.1$ and $t+0.1$. If the weighted sum of the numbers of satisfied inequalities at $w = (t+0.1)$ is larger than that at $w = (t - 0.1)$, then we seek the optimal w in the subinterval from t to $(t + 0.1)$ by repeating the above process with the computation of the weighted sums at $w = t, t + 0.01, \ldots, t + 0.1$; else, the process is applied to find w in the subinterval from $(t - 0.1)$ to t. The iteration is terminated when a subinterval width is sufficiently small. Although the value of w thus obtained may not be the optimal w, experimental results in the next Section indicates that essentially ideal retrieval performance is attained when the degrees of relevance are computed using this value of w.

7 Further experimental results

We are not aware of a document collection with linkage information and a query collection which resembles typical Internet queries (having an average of 2.2 terms per query) and having relevance judgments of documents against the queries. As a result, we created a collection of documents and user queries as follows. Real users were asked to submit queries. A total of 71 queries were obtained. The average number of terms per query is 2.3. For each query, the top 50 documents retrieved by Yahoo were downloaded. For each such retrieved document, all documents to which it was linked were also downloaded. Relevant documents among the downloaded documents were identified. Altogether, the collection has 29222 documents. This collection was arbitrarily divided into 30 databases. For each query, the relevant documents were randomly distributed into 4 to 10 randomly chosen databases. This forms our test collection.

From the linkage information in the downloaded documents, we computed the normalized rank of each document [17]. Then, we determined the value of the relative weight w as described in the previous section. The value we obtained for w is 0.77 (fairly close to 0.8, the value of w we chose for the experiments with the Stanford and TREC data). This value of w, 0.77, was then used to perform distributed information retrieval in the 30 databases.

The results are given in Table 12. The columns *db_effort*, *doc_effort* and *per_rel_doc* have the same semantics as those used in the tables presented in section 5. The second column **rel_doc_ratio** represents the ratio of the total number of relevant documents retrieved by the metasearch engine for the 71 queries to the total number of relevant documents retrieved by a single search engine containing the same set of documents. In the row for $m = 5$, for example, the number 0.993 means that the total number of relevant documents found among the top 5 documents of the retrieval results from the metasearch engine is 0.993 times the total number of relevant documents found among the top 5 documents of the retrieval results from a single search engine containing the same set of documents.

m	rel_doc_ratio	db_effort	doc_effort	per_rel_doc
5	0.993	173.9%	218.9%	99.6%
10	1.0077	136.9%	161.5%	99.6%
20	1.0042	117.5%	134.5%	99.6%
30	0.9969	109.9%	125.4%	99.7%

Table 12: Experimental results for a collection of Web pages

It can be seen that essentially the same retrieval effectiveness as if all documents were placed in a single database is obtained. In fact, when the numbers of documents retrieved are 10 and 20, the distributed retrieval yields even better results than centralized retrieval. The reason is that databases which have quite a few documents with very high degrees of relevance have a higher concentration of relevant documents than databases which have a small number of documents with high degrees of relevance. The database selection algorithm we supply tends to retrieve more documents from the former databases, thus yielding more relevant documents than the centralized retrieval. Consider, for example, the query "Traditional Chinese herbal medicine". Table 13 shows the centralized retrieval result, that is the retrieval result that would be obtained if all data from the distributed database were placed in a centralized database. The column **m** represents the number of top documents in the retrieval result. Each of the other columns represents a local database. A non-empty entry in these columns contains two numbers. The number *enclosed by a pair of brackets* is the number of *relevant* documents retrieved from the corresponding local database and the other number is the number of documents retrieved from the local database. The pair 10 (9) under column db12 and in the row of $m = 20$, for example, means that among the top 20 documents for the query, 10 were retrieved from local database db12 and 9 of these 10 retrieved documents are relevant. Table 14 is a similar table. It represents the *actual* retrieval result from the metasearch engine. From these two tables, note that for the top 20 documents retrieved in the centralized case, three of them came from the databases db6, db13 and db19 and all these three are irrelevant. However, in the actual retrieval, these 3 databases were not picked by our metasearch engine; instead, the search effort was focused on db28, db9 and db12, those databases with higher proportions of relevant documents.

m	db28	db9	db12	db23	db8	db6	db13	db19
5	1 (1)	2 (2)	1 (1)	1 (0)				
10	2 (2)	2 (2)	4 (4)	1 (0)	1 (0)			
20	2 (2)	3 (3)	10 (9)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)

Table 13: Centralized Retrieval for the query "Traditional Chinese herbal medicine"

m	db28	db9	db12	db23	db8
5	1 (1)	2 (2)	1 (1)	1 (0)	
10	2 (2)	2 (2)	5 (5)	1 (0)	
20	3 (3)	3 (3)	12 (11)	1 (0)	1 (0)

Table 14: Actual Retrieval for the query "Traditional Chinese herbal medicine"

8 Conclusion

We have shown that linkage information between documents can be utilized in their retrieval from distributed databases in a metasearch engine. This is the first time the information is employed in a metasearch engine.

Our experimental results show that the techniques we provide yield retrieval effectiveness essentially the same as that of a centralized database containing all the documents. The strengths of our techniques are their simplicity (the necessary and sufficient conditions for optimal ranking of databases, the co-ordination algorithm which guarantees optimal retrieval if the databases are optimally ranked) and flexibility (the estimation algorithms to rank databases, while taking into consideration the linkage information between documents). The techniques given here are readily generalizable to the situation where there are numerous databases. In that case, it may be desirable to place database representatives in a hierarchy and search the hierarchy so that most database representatives need not be searched and yet the same retrieval effectiveness is achieved as if all database representatives were searched [33].

Acknowledgement: We are grateful to L. Gravano and H. Garcia-Molina for providing us with one of three collections of documents and queries used in our experiments.

References

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

[2] C. Baumgarten. *A Probabilistic solution to the selection and fusion problem in distributed Information Retrieval*, ACM SIGIR Conference, 1999.

[3] 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), 431-448, May-June 1995.

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

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

[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. *Merging Ranks from Heterogeneous Internet sources.* International Conferences on Very Large Data Bases, 1997.

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

[9] 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.

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

[11] T. Kirk, A. Levy, Y. Sagiv, and D. Srivastava. *The Information Manifold.* AAAI Spring Symposium on Information Gathering in Distributed Heterogeneous Environments. 1995.

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

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

[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] E. Lima and J. Pedersen. *Phrases Recognition and Expansion for Short, Precision-biased Queries based on a Query Log.* ACM SIGIR Conference, August 1999.

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

[17] L. Page, S. Brin, R. Motwani, and Terry Winograd. *The PageRank Citation Ranking: Bring Order to the Web.* Technical Report, Stanford University, 1998.

[18] A. Powell, J. French, J. Callan, M. Connell and C.Viles. *The Impact of Database Selection on Distributed Searching.* ACM SIGIR 2000.

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

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

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

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

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

[24] J. Xu, and B. Croft. *Cluster-based Language Models for Distributed Retrieval.* ACM SIGIR Conference 1999.

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

[26] 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.

[27] reference removed for double blind reviewing

[28] reference removed for double blind reviewing

[29] reference removed for double blind reviewing

[30] reference removed for double blind reviewing

[31] reference removed for double blind reviewing

[32] reference removed for double blind reviewing

[33] reference removed for double blind reviewing