# A Highly Scalable and Effective Method for Metasearch

Weiyi Meng and Zonghuan Wu
State University of New York at Binghamton
Clement Yu
University of Illinois at Chicago
and
Zhuogang Li
State University of New York at Binghamton

---

A metasearch engine is a system that supports unified access to multiple local search engines. Database selection is one of the main challenges in building a large-scale metasearch engine. The problem is to efficiently and accurately determine a small number of potentially useful local search engines to invoke for each user query. In order to enable accurate selection, metadata that reflect the contents of each search engine need to be collected and used. This paper proposes a highly scalable and accurate database selection method. This method has several novel features. First, the metadata for representing the contents of all search engines are organized into a single integrated representative. Such a representative yields both computation efficiency and storage efficiency. Second, the new selection method is based on a theory for ranking search engines optimally. Experimental results indicate that this new method is very effective. An operational prototype system has been built based on the proposed approach.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems–distributed databases; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval–search process; selection process; H.3.4 [**Information Storage and Retrieval**]: Systems and Software–information networks

General Terms: Design, Experimentation, Performance

Additional Key Words and Phrases: Metasearch Engine, Resource Discovery, Database Selection, Distributed Text Retrieval

---

Name: Weiyi Meng, Zonghuan Wu, Zhuogang Li
Address: Department of Computer Science, State University of New York at Binghamton, Binghamton, NY 13902; email: meng@cs.binghamton.edu.
Name: Clement Yu
Address: Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607; email: yu@eecs.uic.edu.

## 1. INTRODUCTION

The World Wide Web has become a vast information resource in recent years. By February of 1999, there were already approximately 800 million publicly indexable pages on the Web [Lawrence and Lee Giles 1999]. Finding desired data is one of the most popular ways the Web is utilized. Many search engines have been created to facilitate the retrieval of web pages. Each search engine has a *text database* that is defined by the set of documents that can be searched by the search engine. In this paper, a search engine and its database will be used interchangeably. Usually, an index for all documents in the database is created in advance. For each *term* which represents a content word or a combination of several (usually adjacent) content words, this index can identify the documents that contain the term quickly. In this paper, we consider only search engines that support vector space queries (i.e., queries that can be represented as a set of terms with no Boolean operators). Less than 10% of all user queries use Boolean operators [Jansen et al. 1998].

Several major search engines on the Web, for example AltaVista, Google and NorthernLight, have been attempting to index the entire Web and provide a search capability for all web pages. However, these centralized search engines suffer from a number of limitations [Hawking and Thistlewaite 1999]. For example, the coverage of the Web by each of them is limited [Lawrence and Lee Giles 1998b; Lawrence and Lee Giles 1999] due to various reasons such as robot exclusion and the lack of appropriate links. As another example, as these major search engines get larger, higher percentages of their indexed information are becoming obsolete. Furthermore, many documents indexed by major general-purpose search engines are of low quality due to the lack of good quality-control mechanisms (e.g., do not filter out bad quality documents such as those with little content) and the lack of maintenance effort (e.g., do not identify and remove duplicates). More and more people are having doubt about the search effectiveness and the scalability of the centralized search engine technology for searching the entire Web [Hawking and Thistlewaite 1999; Sugiura and Etzioni 2000].

One way to tackle the problem of limited coverage of the Web by individual search engines is to combine the coverages of multiple search engines through the creation of a *metasearch engine*. A metasearch engine is a system that supports unified access to multiple local search engines. It does not maintain its own index on web pages but a sophisticated metasearch engine often maintains characteristic information about each underlying local search engine in order to provide better service. When a metasearch engine receives a user query, it first passes the query (with necessary reformatting) to the appropriate local search engines, and then collects (sometimes, reorganizes) the results from its local search engines. Most existing metasearch engines employ a small number of general-purpose search engines as their underlying local search engines (e.g., MetaCrawler [Selberg and Etzioni 1995; Selberg and Etzioni 1997], SavvySearch [Dreilinger and Howe 1997], ProFusion [Fan and Gauch 1999; Gauch et al. 1996]). While these metasearch engines can indeed cover a larger portion of the Web than any individual search engine, they do not solve the other problems associated with large general-purpose search engines, for example, the inability of updating the index information quickly and the lack of the mechanism and effort to control the quality of indexed documents.

There are hundreds of thousands of special-purpose search engines that focus on documents in confined domains such as documents in an organization or of a specific subject area [Bergman 2000]. For example, the Cora search engine (cora.whizbang.com) focuses on computer science research papers and Medical World Search (www.mwsearch.com) is a search engine for medical information. Many organizations have their own search engines. A recent survey indicates that the combined coverage of the Web by these special-purpose search engines is hundreds of times larger than that by any single general-purpose search engine [Bergman 2000]. The reason is that many special-purpose search engines have special document collections that are not indexable by Web robots employed by general-purpose search engines. Thus, an approach that can provide the search capability for a much larger portion of the Web than any single general-purpose search engine or the combination of several general-purpose search engines is to combine all these special-purpose search engines. In this paper, we consider a metasearch engine that is aimed at employing all special-purpose search engines as its local search engines. In addition to the significantly increased search coverage of the Web, such a metasearch engine has several other advantages over general-purpose search engines. First, since usually each special-purpose search engine covers only a small portion of the Web it is easier for it to keep its index data up to date (i.e., updating of index data to reflect the changes of documents can be carried out more frequently). Second, the documents indexed by special-purpose search engines are likely to be of better quality due to better quality control and maintenance. Third, the databases of special-purpose search engines are natural clusters of the Web documents. There is evidence that retrieval from special-purpose search engines [Sugiura and Etzioni 2000] and from clusters can yield higher effectiveness [Xu and Croft 1999]. In addition, running a metasearch engine requires much smaller investment in hardware (computers, storages, ...) in comparison to running a large general search engine such as Google which uses thousands of computers.

There are several challenges to implement an effective and efficient metasearch engine. Among the main challenges, the *database selection problem* is to identify, for a given user query, the local search engines that are likely to contain useful documents for the query. The objective of performing database selection is to improve efficiency as by sending each query to only potentially useful search engines, network traffic and the cost of searching useless databases can be reduced. In order to perform database selection well, a *representative* for each database needs to be stored in the metasearch engine to indicate the contents of the database. The *collection fusion problem* is to retrieve documents from selected databases and then merge these documents with the objective of listing more useful documents ahead of less useful ones. Various heterogeneities among multiple search engines often make it very difficult to achieve a good fusion [Meng et al. 1999b]. A good metasearch engine should have the retrieval effectiveness close to that as if all documents were in a single database while minimizing the access cost.

In this paper, we propose a new approach to perform database selection and collection fusion. This method uses the framework that was developed in [Yu et al. 1999b; Yu et al. ] for ranking databases optimally based on the similarity of the most similar document in each local database (see Section 3 for more information). The main contribution of this paper is the development and the experiment of a

new technique for ranking databases. This technique is based on a new measure to rank databases and a novel database representative that has the following features. First, it is highly scalable in terms of both computation and storage requirement. In fact, it can scale to virtually unlimited number of local databases. Second, it is an integrated representative for all local databases in contrast to one representative for each local database in existing approaches. Third, for single-term queries, which occur frequently in the Internet environment, this technique guarantees the correct selection of databases. Fourth, for multi-term queries, certain dependencies among these terms are examined to see if adjacent terms could be combined to simulate phrases. Our experimental results indicate that our new method is not only very scalable but also very accurate. We believe that this method represents a major step forward towards building extremely large-scale metasearch engines. An operational prototype metasearch engine based on our method has been implemented.

The rest of the paper is organized as follows. In Section 2, related work is reviewed and compared. In Section 3, we review a framework of performing database selection and collection fusion using the similarity of the most similar document in each database. In Section 4, we present our new technique based on this framework. Experimental results will be presented in Section 5. We briefly describe our prototype system in Section 6. Finally, we conclude the paper in Section 7.

## 2. RELATED WORK

In the last several years, a large number of research papers on issues related to metasearch engines or distributed collections have been published (e.g., [Baumgarten 1997; Callan et al. 1995; Dreilinger and Howe 1997; Gravano and Garcia-Molina 1995; Gravano and Garcia-Molina 1997; Liu et al. 2001b; Manber and Bigot 1997; Meng et al. 1998; Meng et al. 1999a; Selberg and Etzioni 1997; Sugiura and Etzioni 2000; Voorhees et al. 1995; Yuwono and Lee 1997]).

For database selection, most approaches rank the databases for a given query based on certain usefulness measures. For example, gGlOSS uses the sum of the document similarities that are higher than a threshold [Gravano and Garcia-Molina 1995], CORI Net uses the belief that a database contains relevant documents due to the terms in a given query [Callan et al. 1995], D-WISE uses the sum of weighted document frequencies of query terms [Yuwono and Lee 1997], Q-Pilot uses the dot-product similarity between an expansion query and a database description [Sugiura and Etzioni 2000], and one of our approaches uses the expected number of documents whose similarities are higher than a threshold [Meng et al. 1998]. All these database ranking methods are heuristics as they are not designed to produce optimal orders based on some optimality criteria. In [Yu et al. 1999b; Yu et al. ], the measure used to rank a database is the similarity of the most similar document in the database. It is shown that ranking databases in descending order of the similarity of the most similar document in each database is a necessary and sufficient condition to rank databases optimally for retrieving the $m$ most similar documents across all databases for any positive integer $m$. A necessary and sufficient condition for ranking databases optimally was also given in [Kirk et al. 1995]. However, [Kirk et al. 1995] considered only the databases and queries that are for structured data. The database selection method in [Liu 1999] also considered only databases and queries for mostly structured data. In contrast, unstructured text data are

considered in [Yu et al. 1999b; Yu et al. ].

For collection fusion, most earlier approaches use *weighted allocation* to retrieve documents, that is, retrieve proportionally more documents from databases that have higher ranking scores (e.g., CORI Net, D-WISE, ProFusion [Fan and Gauch 1999], and MRDD [Voorhees et al. 1995]), and use *adjusted local similarities* of documents to merge retrieved documents (e.g., D-WISE, and ProFusion). These approaches are all heuristics and are not aimed at guaranteeing the retrieval of all potentially useful documents for a given query. In [Gravano and Garcia-Molina 1997; Meng et al. 1998], to determine what documents to retrieve from a local database, approaches are proposed to find a tight local similarity threshold for the local database based on a global similarity threshold. These approaches aim at guaranteeing the retrieval of all potentially useful documents from each selected database while minimizing the retrieval of useless documents. The problem with this type of approaches is that they must know what local similarity function is used in each search engine but the similarity function is usually proprietary. The Inquirus metasearch engine [Lawrence and Lee Giles 1998a] uses the real global similarities of documents to merge retrieved documents. The advantage is that high quality merging can be achieved. The disadvantage is that documents may need to be fetched to the metasearch engine to enable the computation of their global similarities. The collection fusion approach in [Yu et al. 1999b; Yu et al. ] utilizes an approximate optimal database order to determine what documents to retrieve and uses real global similarities to merge retrieved documents.

The database selection and collection fusion framework used in this paper is based on our previous work in [Yu et al. 1999b; Yu et al. ]. In a nutshell, this framework first tries to rank local databases optimally using the necessary and sufficient condition mentioned above. Next, an algorithm is used to determine what databases should be searched and what documents from each searched database should be returned to the metasearch engine. Finally, the global similarities of returned documents are used to merge all returned documents. This framework will be reviewed in Section 3. The focus of this paper is on improving the scalability of database selection within this framework. Our main contribution in this paper is that we have devised a new database selection method. The new method employs a new measure to rank databases and can on one hand scale to virtually unlimited number of local databases in a metasearch engine in terms of both computation and space requirement and on the other hand essentially maintain the retrieval accuracy of the previous method. We believe that this is a major step forward towards building a very large scale metasearch engine. A recent whitepaper prepared by a working group on resource discovery (database selection) asserted that there are potentially one million repositories on the Web [Arms et al. 1999] and called on the development of highly scalable methods for resource discovery.

Most existing systems/approaches consider only small-scale metasearch engines that have from several to a few hundred local search engines. It is unlikely that these approaches can scale to hundreds of thousands of local search engines and at the same time achieve good effectiveness. The reasons are as follows. First, existing methods compare a given query against all database representatives to perform database selection. This is computationally very expensive as the number of databases is very large. Second, based on existing methods [Callan et al. 1995;

Gravano and Garcia-Molina 1995; Liu et al. 2001b; Meng et al. 1999a; Yuwono and Lee 1997], in order to perform database selection well, a "detailed" representative for each database is needed. Here "detailed" means that one or more pieces of statistical information for each term appearing in a database are used. A detailed representative for a database may have roughly 1% of the size of that of the database. As a result, for a metasearch engine with hundreds of thousands of local search engines, the total size of these representatives could be hundreds or even thousands of times of that of an average database. Consequently, the representatives may have to be stored on slower storage devices (such as disk) instead of on memory, causing the database selection computation to be slowed down.

In contrast, in this paper, we propose a novel integrated representative for all databases, instead of a separate representative for each database in all existing methods. The size of the integrated database representative can be kept below a few GBs, regardless of the number of databases there might be in a metasearch engine. Moreover, only a small constant number of databases needs to be considered for each query during database selection. As a result, our method is highly scalable in both computation and storage. In addition, for typical Internet queries, our approach retrieves close to 100% of the most similar documents.

In [Baumgarten 1997], a theoretical framework was provided for achieving *optimal* results in a distributed environment. Recent experimental results reported in [Baumgarten 1999] show that if the number of documents retrieved is comparable to the number of databases, then good retrieval effectiveness can be achieved; otherwise, the performance deteriorates substantially. In the Internet environment, most users are interested in finding a small number of good documents for their queries. Our method shows good experimental results in this environment (see Section 5). In addition, our theory differs from that given in [Baumgarten 1997] substantially.

In [Xu and Callan 1998], experimental results were given to demonstrate that it was possible to retrieve documents in distributed environments with essentially the same effectiveness as though all data were at one site. However, the results depended on the existence of a *training collection* which has similar coverage of the subject matters and terms as the collection of databases to be searched. In the Internet environment where data are highly heterogeneous, it is unclear whether such a training collection can in fact be constructed. Even if such a collection can be constructed, the storage penalty could be very high in order to accommodate the heterogeneity. In [Xu and Croft 1999], 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.

Please see [Meng et al. ] for a more comprehensive review of other work in the metasearch engine and distributed information retrieval area.

## 3. A FRAMEWORK FOR DATABASE SELECTION AND COLLECTION FUSION

A query in this paper is simply a set of words submitted by a user. It is transformed into a vector of *terms* with *weights* [Salton and McGill 1983], 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 (i.e., *term weight*) of the query vector corresponding to the term is positive; if it is absent, the weight is zero. A document is similarly transformed into a vector with weights. The weight of a term in a query (document) is usually computed based on the *term frequency* (the number of occurrences, denoted by $tf$) of the term in the query (document) and the *document frequency* (the number of documents having the term, denoted by $df$) of the term [Salton and McGill 1983; Sparck Jones 1972; Yu and Meng 1998]. The weight factor based on the $tf$ information is the *tf weight* and the weight factor based on the $df$ information is the *idf weight*. 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 *lengths* of the two vectors to normalize the similarity between 0 and 1. The similarity function with such a normalization is known as the *Cosine* function [Salton and McGill 1983]. When the $idf$ weight of each term is computed based on the global $df$ of the term (i.e., the number of documents containing the term across all databases), the computed similarities are global similarities. Note that if there are no or little overlap among local databases, the sum of local $df$s of a term in all local databases can be used as an approximation of the global $df$ of the term. If there are serious overlaps among local databases, then a sampling technique such as that in [Bharat and Broder 1998] can be extended to estimate the global $df$ of a term.

EXAMPLE 1. Let $q = (q_1, ..., q_n)$ be a query, where $q_i$ is the tf weight of term $t_i$ in $q$. Let $gidf_i$ be the global idf weight of $t_i$. Then the query vector is $q' = (q_1 * gidf_1, ..., q_n * gidf_n)$. Let $d = (d_1, ..., d_n)$ be a document vector, where $d_i$ is the tf weight of $t_i$ in $d$. Then $d_i/|d|$ is the normalized weight of $t_i$ in $d$, where $|d| = \sqrt{d_1^2 + ... + d_n^2}$ is the length of $d$. Based on the *Cosine* similarity function, the global similarity between query $q$ and document $d$ is:

$$sim(q,d) = \frac{\sum_{i=1}^{n} q_i * gidf_i * d_i}{|q'| * |d|} = \left(q_1 * gidf_1 * \frac{d_1}{|d|} + ... + q_n * gidf_n * \frac{d_n}{|d|}\right)/|q'| \quad (1)$$

□

In this section, we review a framework for database selection and collection fusion. This framework was first introduced in [Yu et al. 1999a; Yu et al. 1999b]. Suppose a user is interested in retrieving the $m$ most similar documents for a query $q$ from $N$ databases $D_1, D_2, \ldots, D_N$, where $m$ is any positive integer. This framework can be summarized into one definition on optimal database ranking, a necessary and sufficient condition for ranking databases optimally and an algorithm for integrated database selection and collection fusion based on ranked databases.

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

Intuitively, the ordering is optimal because the $k$ databases containing the $m$ most similar documents to the query are ranked ahead of other databases. Note that the ordering of the databases depends on the query $q$. For ease of presentation,

we shall assume that all similarities of the documents with the query are distinct so that the set of the $m$ most similar documents to the query is unique.

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

EXAMPLE 2. Consider three databases $D_1, D_2$ and $D_3$. If the global similarities of the most similar documents in the databases $D_1, D_2$ and $D_3$ to a given query are 0.6, 0.75 and 0.5, respectively. Then, the databases should be ranked in the order $[D_2, D_1, D_3]$ for the query. □

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

Based on the optimal order of the databases $[D_1, \ldots, D_N]$, an algorithm, known as **OptDocRetrv**, was developed to perform database selection and collection fusion [Yu et al. 1999b; Yu et al. ]. This algorithm is sketched as follows. Suppose the first $s$ databases have been selected ($s$ is 2 initially if $m \geq 2$). Each of these selected search engines returns the actual global similarity of the most similar document in its database to the metasearch engine which computes the minimum, denote *min_sim*, of these $s$ values. Each of the $s$ search engines then returns to the metasearch engine those documents whose global similarities are greater than or equal to *min_sim*. Note that at most $m$ documents from each search engine need to be returned to the metasearch engine. If $m$ or more documents have been returned from the $s$ search engines, then they are sorted in descending order of similarity and the first $m$ documents are returned to the user. Otherwise, the next database in the optimal order will be selected and the above process is repeated until at least a total $m$ documents are returned to the user.

Note that in the above algorithm, collection fusion is based on the actual global similarities of documents. It has been shown [Yu et al. ] that if the databases are ranked optimally, then algorithm *OptDocRetrv* will guarantee the retrieval of all the $m$ most similar documents.

In order to apply this framework in practice, the following two problems must be solved. First, we need to figure out how to obtain from any local database those documents whose global similarities with a given query are greater than or equal to a given threshold (e.g., the *min_sim* in each iteration of **OptDocRetrv**). Note that local search engines retrieve documents based on local similarity functions and term statistics that may cause the local similarity of a document to be different from the global similarity of the document. This problem has been addressed in [Meng et al. 1998; Yu et al. 1999a] and will not be discussed further in this paper. Second, Proposition 1 cannot be used as is because we cannot afford to search each database to obtain the global similarity of its most similar document. Instead, for

each database, we need to **estimate** the required similarity. In [Yu et al. 1999b; Yu et al. ], an estimation method that uses two types of database representatives was proposed. There is a global representative for all databases and it stores the global *df* for each term in these databases. There is also a separate representative for each database and it stores two pieces of information for each term. This estimation method has a time complexity that is linear in terms of the number of terms in a query. However, the query needs to be compared with each database representative. Thus if the metasearch engine has a large number of databases, this method does not scale very well in terms of computation efficiency and storage space.

## 4. A NEW DATABASE RANKING METHOD

In this section, we propose our new method for database selection based on the framework described in Section 3. A key step is to rank databases according to the global similarity of the most similar document in each database. Our previous methods tried to estimate the similarity of the most similar document in each database directly [Yu et al. 1999b; Yu et al. ]. A substantial amount of information about each database is needed to enable accurate estimation. The new method takes a different approach. Instead of using the similarity of the most similar document to rank databases, we rank the databases based on a different measure. This new method has two appealing features. First, the new measure can be obtained using less information than estimating the similarity of the most similar document. Our novel integrated representative permits the measure to be computed very efficiently. Second, the ranking of databases based on the new measure matches very well with that based on the similarity of the most similar document as indicated by our experimental results to be reported in Section 5. In Section 4.1, we describe our new database ranking measure. In Section 4.2, we introduce our integrated database representative. In Section 4.3, we discuss how to incorporate one type of term dependencies into the solution.

### 4.1 The New Ranking Measure

Consider a term $t_i$ and a local database $D_j$. Let $mnw_{i,j}$ and $anw_{i,j}$ be the *maximum normalized weight* and the *average normalized weight* of $t_i$ in $D_j$, respectively. The quantity $mnw_{i,j}$ is defined as follows. First, if $d = (d_1, ..., d_i, ..., d_n)$ is the vector representation of a document in $D_j$, where $d_i$ is the weight of term $t_i$, then $d_i/|d|$ is the *normalized weight* of $t_i$ in $d$ ($|d|$ is the length of $d$). Next, $mnw_{i,j}$ is the maximum of the normalized weights of $t_i$ in all documents in database $D_j$, that is, $mnw_{i,j} = \max_{d \in D_j} \left\{ \frac{d_i}{|d|} \right\}$. Similarly, $anw_{i,j}$ is simply the average of the normalized weights of $t_i$ over all documents in $D_j$, including documents not containing term $t_i$. Let $gidf_i$ be the global inverse document frequency weight of $t_i$.

Consider a given user query $q$. Suppose the query vector of $q$ is $q' = (q_1 * gidf_1, \ldots, q_n * gidf_n)$ (see Example 1). Then the global similarity of the most similar document of database $D_j$ with respect to $q$ can be estimated by [Yu et al. ]:

$$\max_{1 \le i \le n} \left\{ q_i * gidf_i * mnw_{i,j} + \sum_{\substack{k=1 \\ k \neq i}}^{n} q_k * gidf_k * anw_{k,j} \right\} / |q'| \qquad (2)$$

By comparing Formula (1) and Formula (2), the intuition for having this estimate can be described as follows. The most similar document in a database is likely to have the maximum normalized weight on one of the query terms, say term $t_i$. This yields the first half of the above expression within the braces. For each of the other query terms, the document takes the average normalized value. This yields the second half. Then, the maximum is taken over all $i$, since the most similar document may have the maximum normalized weight on any one of the $n$ query terms. Normalization by the query norm, $|q'|$, yields a value less than or equal to 1. We shall drop $|q'|$ for ease of presentation. This will not have any impact as the relative similarity values of the most similar documents of the different databases are not changed.

Through a large number of experiments, we observed that the maximum normalized weight of a term is typically two or more orders of magnitude larger than the average normalized weight of the term as the latter is computed over all documents including those not containing the term. This feature implies that in Formula (2), if all query terms have the same $tf$ weight (a reasonable assumption as in a typical query, each term appears once), $q_i * gidf_i * mnw_{i,j}$ is likely to dominate $\sum_{k=1,\, k\neq i}^{n} q_k * gidf_k * anw_{k,j}$, especially when the number of terms, $n$, in a query is small (which is typically true in the Internet environment [Jansen et al. 1998; Kirsch 1998]). In other words, whether database $D_j$ is going to be ranked high (i.e., whether its most similar document is going to have a relatively large similarity) with respect to a given query $q$ is largely determined by the value of $\max_{1\leq i\leq n}\left\{q_i * gidf_i * mnw_{i,j}\right\}$.

The above discussion is summarized below. For a given term $t_i$ and database $D_j$, let $am_{i,j} = gidf_i * mnw_{i,j}$ be the *adjusted maximum normalized weight* of term $t_i$ in $D_j$. Let $t_i, i = 1, ..., n$, be the $n$ terms in a query $q$. We define the *ranking score* (or **rs** for short) of database $D_j$ with respect to $q$ as follows:

$$rs(q, D_j) = \max_{1\leq i\leq n}\{q_i * am_{i,j}\} \tag{3}$$

The ranking score defined above will be our new measure to rank databases. For a given database, this measure can be computed for any query by a database representative that stores only one piece of information per distinct term in the database. For term $t_i$ in database $D_j$, this piece of information is $am_{i,j}$. Note that Formula (3) has a **linear time complexity** in terms of the number of terms in the query. In the rest of the paper, we will attempt to establish, by both theory and experimental results, that by ranking databases based on their *ranking scores* for short queries (typical of Internet queries [Jansen et al. 1998; Kirsch 1998]), the ranking is very close to the optimal ranking based on the similarity of the most similar document in each database.

### 4.2 Integrated Representative of Databases

In order to compute the ranking score of a database with respect to any given query, the adjusted maximum normalized weight of each term in the database needs to be obtained and stored. If all the documents in a database are accessible, then

the needed statistical information can be easily obtained. There are several situations where the documents in a database can be accessible. First, the database is under the control of the developer of the metasearch engine such as in the case of an Intranet environment. Second, the documents can be independently obtained. For example, the search engine at www.binghamton.edu/search/ is for searching all Web pages at Binghamton University. But these Web pages can also be independently obtained by using a Web spider (robot) starting with the home page of the university (www.binghamton.edu). Third, a local search engine is cooperative. For example, in metasearch engine NCSTRL (Networked Computer Science Technical Reference Library, cs-tr.cs.cornell.edu), all local databases must sign up to join the metasearch engine. In this case, the metasearch engine may simply request/require each local search engine to provide the statistical information needed for database selection. Clearly, there will be cases where the documents of a database cannot be independently obtained and a local search engine is un-cooperative. In these cases, a technique known as *query sampling* [Callan et al. 1999] could be adopted to estimate the needed statistics. For the rest of this paper, we assume that the adjusted maximum normalized weights have already been obtained.

If we follow the example of existing approaches, we would create a separate *database representative* for each database. In this case, the representative for database $D$ would contain the adjusted maximum normalized weight for each term in $D$. When a query is received by the metasearch engine, the query information and the representative of each database will be used to compute the ranking score of each database. After the databases are ranked, the **OptDocRetrv** algorithm reviewed in Section 3 can be used to select databases and retrieve documents.

The database representative introduced in Section 4.1 stores only one piece of information per term and is already more scalable than most existing database selection approaches that use detailed database representatives (e.g., [Callan et al. 1995; Gravano and Garcia-Molina 1995; Liu et al. 2001b; Meng et al. 1999a]) in terms of the storage space required. For metasearch engines that have up to a few hundreds of local databases, we probably can afford to have a separate representative for each database and store all of them in the metasearch engine. However, if our goal is to build a metasearch engine that may have hundreds of thousands of local search engines so that the entire Web can be potentially searched by the metasearch engine, then it may not be economical to have a separate representative for each search engine. Computing hundreds of thousands of ranking scores for each query is very time consuming. Our solution to this problem is to create a novel integrated representative for all databases.

For a given positive integer $r$ and term $t_i$, let $LAM(t_i, r)$ contain the $r$ largest $am_{i,j}$'s over all $D_j$'s. In other words, $LAM(t_i, r)$ contains only the $r$ largest adjusted maximum normalized weights of $t_i$ across all local databases. The integrated representative that we propose for all local databases is as follows. For each term $t_i$, a set of up to $r$ pairs of the format $(did_{i,j}, am_{i,j})$ is kept in the integrated representative, where $am_{i,j} \in LAM(t_i, r)$ and $did_{i,j}$ is the identifier of the database having $am_{i,j}$. Thus, for each term, the $r$ largest adjusted maximum normalized weights and their corresponding database ids are stored. The idea is to store only the information associated with the most important databases for each potential query term.

When evaluating a query $q$ using the integrated database representative, we compute the ranking scores for only those databases whose id appears in at least one $LAM(t_i, r)$, where $t_i$ is a query term. Specifically, for a database $D_j$ which has the largest adjusted maximum normalized weights for a subset $S$ of query terms, the ranking score of this database is computed by $\max_{t_i \in S}\{am_{i,j} * q_i\}$, where $q_i$ is the weight of term $t_i$ in the query. Thus, for a query having $n$ terms, at most $n * r$ ranking scores are computed. This is independent of the number of databases. In the Internet environment, $n$ is usually very small ($n = 2.2$ on the average [Kirsch 1998]). The value of $r$ is also a small constant (see next paragraph). As a result, our proposed method is highly scalable in terms of computation.

One way to determine the value $r$ is as follows. If the metasearch engine is designed to search no more than $u$ search engines for any given query, then $r$ can be set to $u$. In practice, a small $u$, say 20, is likely to be sufficient for most users if relevant search engines can be selected. The above integrated representative can scale to virtually unlimited number of local databases in terms of storage. The reason is as follows. First, suppose a rough bound of the number of distinct terms, say $M = 10$ millions, exists regardless of the number of local databases participating in the metasearch engine. Next, for each term, only a small constant number ($2 * r$) of quantities ($r$ largest adjusted maximum normalized weights and $r$ database identifiers) are stored in the integrated representative. Therefore, the total size of this representative is bounded by $(10+4*2*r)*M$ bytes, assuming that each term occupies 10 bytes on the average and each quantity occupies 4 bytes. When $r = 20$ and $M = 10,000,000$, $(10 + 4 * 2 * r) * M = 1.7$ GB, well within the memory capacity of a well equipped server. In reality, there may not be a clear bound to the number of distinct terms and there may be more than $M$ terms. However, the scalability of the integrated representative approach is still very good as it stores only a small constant number of quantities for each term regardless of how many databases may contain the term. In contrast, in non-integrated representatives, the number of pieces of information stored for each term is a constant factor of the number of databases. In summary, our integrated representative approach is highly scalable in both computation and storage.

Intuitively, a database selection method is effective if the most desired documents are contained in a relatively small number of databases selected by this method. In Section 5, we will conduct experiments to evaluate the effectiveness of our method based on more rigorous measures. The proposition below shows that for any single-term query (which constitutes about 30% of all Internet queries [Jansen et al. 1998]), the local databases selected by the integrated representative are guaranteed to contain the $m$ most similar documents in all databases with respect to the query when $m \leq r$.

PROPOSITION 2. For any single-term query, if the number of documents desired by the user, $m$, is less than or equal to $r$ — the number of adjusted maximum normalized weights stored in the integrated representative for the query term — then all of the $m$ most similar documents to the query are contained in the $r$ local databases whose adjusted maximum normalized weights for the query term are stored in the integrated representative.

PROOF. Note that the maximum normalized weight of the (single) query term

in each database is also the similarity of the most similar document in the database with respect to the query. This means that for any single term query, if we rank the databases in descending order of the maximum normalized weights of the term, the databases will be ranked optimally for the query. Note that the order based on the maximum normalized weights will be identical to that based on the adjusted maximum normalized weights as the two types of weights differ only by the *gidf* weight of the term. However, for a single term, the *gidf* weight is a constant for all documents. Since the $r$ adjusted maximum normalized weights stored in the integrated representative for the query term are the largest, the corresponding $r$ databases will be ranked ahead of other databases. Meanwhile, the $m$ most similar documents with respect to the query will be contained in no more than $m$ databases. Since $r \geq m$, the $r$ databases must contain the $m$ most similar documents to the query. □

Please note that the *gidf* weights are useful only when there are multiple terms in a query.

### 4.3 Combining Terms

The above estimation is based on the assumption that terms are independently distributed. This assumption is not entirely realistic. For example, the two terms "computer" and "algorithm" may appear together more frequently in documents in a database containing computer science literature than that expected if the two terms were independently distributed in the database. In this subsection, we introduce a method to remedy this assumption through the incorporation of one type of dependencies between two adjacent terms. Note that most phrases consist of two terms.

This method works as follows. When a multi-term query $q$ is received, we first examine if certain adjacent term pairs can be combined and treated as a single term. This process is described below. First, we generate all candidate term pairs from the terms in $q$. Precisely, $(t_i, t_k)$ is a *candidate term pair* if after stopwords are removed, $t_i$ and $t_k$ are next to each other (it does not matter whether $t_i$ precedes $t_k$ or $t_k$ precedes $t_i$). Each candidate term pair is a potential phrase. Next, for each local database $D$, we determine whether a candidate term pair $(t_i, t_k)$ is *combinable*. Suppose a document has normalized weight $nw_i$ for $t_i$ and normalized weight $nw_k$ for $t_k$. The (adjusted) maximum normalized weight for the combined term (if the two terms $t_i$ and $t_k$ are combined) is defined to be:

$$mnw_{ik} = \max_{d \in D}\{gidf_i * nw_i + gidf_k * nw_k\} \tag{4}$$

The estimated (adjusted) maximum normalized weight for the combined term assuming the two terms are independent is:

$$emnw_{ik} = \max\{gidf_i * mnw_i + \delta, gidf_k * mnw_k + \delta\} \tag{5}$$

where $mnw_i$ and $mnw_k$ are the maximum normalized weights of $t_i$ and $t_k$, respectively, and $\delta$ is a small positive constant. Ideally, $emnw_{ik}$ should involve the average normalized weights of terms $t_i$ and $t_k$, but since they are not stored in the integrated database representative, a small positive constant, $\delta$, is used. One way

to implement $\delta$ is to let it be the average of the average normalized weights of all terms. Now $(t_i, t_k)$ is said to be *combinable* if $mnw_{ik} > emnw_{ik}$. That two terms are combinable indicates that the two terms are worth combining.

Now for each local database $D$, we determine whether a combinable pair $(t_i, t_k)$ should be combined. A combinable pair is not always combined. For example, suppose term $t_u$ involves two combinable pairs $(t_i, t_u)$ and $(t_u, t_k)$. In this case, only one pair will be actually combined and the choice is made based on which pair, if combined, yields more benefit. We consider the following two cases:

**Case 1:** $(t_i, t_k)$ is the only combinable pair in a query. In this case, $(t_i, t_k)$ should be combined.

**Case 2:** A list of combinable term pairs $(t_1, t_2), (t_2, t_3), ..., (t_s, t_{s+1})$ exists in a query. We first identify the most worthy combinable pair. Let $diff_{ik} = mnw_{ik} - emnw_{ik}$ be the difference between $mnw_{ik}$ and $emnw_{ik}$ for term pair $(t_i, t_k), i = 1, ..., s$ and $k = i + 1$. Let $diff_{xy}$ be the largest difference, $y = x + 1, 1 \leq x \leq s$. Then $(t_x, t_y)$ is the most worthy combinable pair. Now we first combine $(t_x, t_y)$ and then repeat Case 1 or Case 2 for two sublists $(t_1, t_2), ..., (t_{x-2}, t_{x-1})$ and $(t_{y+1}, t_{y+2}), ..., (t_s, t_{s+1})$.

Suppose it is decided that two terms $t_i$ and $t_k$ should be combined for local databases $D_1, ..., D_n$. The integrated representative will be modified as follows. First, a new term (i.e., the combined term) $t_{ik}$ will be created. Second, the $r$ largest (adjusted) maximum normalized weights ($mnw_{ik}$'s) for $D_1, ..., D_n$ will be stored in the integrated representative under the combined term.

To incorporate combined terms into the query evaluation process, two adjustments need to be made. Let $q$ be a query under consideration.

(1) When selecting databases to compute ranking scores, both combined terms and uncombined terms will be considered. As an example, suppose $q$ has two terms $(t_i, t_k)$ and the two terms are combined for some databases but not combined for other databases. In this case, the query is treated as having three terms in this step, namely $t_i$, $t_k$ and the combined term $t_{ik}$. For each term, a set of $r$ databases that have the $r$ largest adjusted maximum normalized weights are identified from the integrated representative. Now we compute the ranking scores for only those databases that appear in at least one of the sets.

(2) When computing the ranking score of a database $D$ with respect to $q$, we have two cases. First, if no terms in $q$ need to be combined for $D_j$, then Formula (3) is used directly to compute $rs(q, D_j)$. Second, if two terms in $q$, say $t_i$ and $t_k$, are combined into a new term $t_{ik}$, then Formula (3) needs to be modified slightly before being used. Specifically, the two components corresponding to $t_i$ and $t_k$, namely $q_i * am_{i,j}$ and $q_k * am_{k,j}$, are replaced by $q_{ik} * mnw_{ik,j}$, where $q_{ik}$ is the weight of the combined term $t_{ik}$ in the query after $t_i$ and $t_k$ are replaced by $t_{ik}$, and $mnw_{ik,j}$ is the (adjusted) maximum normalized weight of $t_{ik}$ in database $D_j$.

In practice, it may be too slow to determine whether two terms in a query should be combined on the fly. We suggest to use the following solution to address this issue. First, common phrases can be compiled in advance and stored in an online dictionary (We assume that there is a precise process to recognize phrases. See for

example [Lima and Pedersen 1999].). Each phrase can be treated as a candidate combined term and whether it should be really combined in each local database is determined offline following the procedure discussed above. When a user query is received, it is first examined to see if it contains a known phrase. If yes, then for databases where the phrase is treated as a combined term, the modified formula is used to compute their ranking scores; for databases where the phrase is not treated as a combined term, the original formula is used. Next, for phrases that are not in the dictionary, a learning process can be implemented. We can keep track of user queries submitted to the system and identify new phrases that have occurred in a number of previous queries. For example, if two adjacent terms appearing in some query have been submitted to the system at least $p$ times for some small integer $p$, then the two terms may be treated as a potential phrase. These potential phrases, if combinable, can be added to the dictionary to benefit future queries that contain them.

## 5. EXPERIMENTAL RESULTS

In this section, we report some experimental results. 221 databases are used in our experiments. These databases are obtained from five TREC document collections created by NIST (National Institute of Standards and Technology of the US). The five collections are CR (Congressional Record of the 103rd Congress), FR (Federal Register 1994), FT (articles in *Financial Times* from 1992 to 1994), FBIS (articles via Foreign Broadcast Information Service) and LAT (randomly selected articles from 1989 & 1990 in *Los Angeles Times*). These collections are partitioned into databases of various sizes ranging from 222 documents (about 2 MB) to 7,760 documents (about 20 MB). A total of more than 558,000 documents ($\approx$ 2 GB in size) are in these databases. There are slightly over 1 million distinct terms in these databases. Generating test databases by partitioning TREC collections to evaluate database selection algorithms has also been used in [French et al. 1998; French et al. 1999].

1,000 Internet queries by real users are used in our experiments. These queries were collected at Stanford University and were used to evaluate the performance of the gGlOSS database selection method [Gravano and Garcia-Molina 1995]. The 1,000 queries used in our experiments are the first 1,000 queries, each having no more than 6 terms, from among about 6,600 queries available. Among the 1,000 queries, 2 queries have no terms (after stopwords are removed), 343 queries are single-term queries, 323 queries have two terms, 185 queries have three terms, 94 queries have four terms, 29 queries have 5 terms and 24 queries have six terms. The average length of these queries is about 2.21. The query length distribution of the 1,000 test queries matches very well with that of over 50,000 queries submitted to the Excite search engine and analyzed in [Jansen et al. 1998]. Another observation made in [Jansen et al. 1998] is that about 97% of all Internet queries have no more than 6 terms. TREC collections come with about 400 queries. The reason that we did not use TREC queries is that their average length is much longer than that of typical Internet queries.

The performance measures of a method to search for the $m$ most similar documents in a set of databases are given as follows. The first two measures indicate the effectiveness (quality) of retrieval while the last two measures reflect the efficiency

of retrieval.

(1) The percentage of correctly identified databases, that is, the ratio of the number of databases which contain one or more of the $m$ most similar documents and are searched by the method over the number of databases which contain one or more of the $m$ most similar documents. This percentage is denoted by **cor_iden_db**.

(2) The percentage of correctly identified documents, that is, the ratio of the number of documents retrieved among the $m$ most similar documents over $m$. This percentage is denoted by **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 similar documents. This ratio is denoted by **db_effort**.

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

For a given set of queries, the measures reported in this paper are averaged over all queries in the set that contain at least one real term. In all experiments, $r = m$ will be used, where $m$ is the number of documents desired by the user and $r$ is the number of adjusted maximum normalized weights stored in the integrated representative for each term.

We also experimented with the following parameter $\beta$. The original algorithm *OptDocRetrv* terminates when at least $m$ documents have been returned to the metasearch engine by local search engines (see Section 3). We use $\beta$ to control when to terminate algorithm *OptDocRetrv*. Specifically, $\beta$ could be chosen to be greater than $m$ — the number of desired documents. For example, when $\beta = 2m$, the algorithm will not stop until at least $2m$ documents have been returned to the metasearch engine by local search engines. From these $2m$ (or more) documents, the most similar $m$ documents are presented to the user. By experimenting with different $\beta$, we would like to see whether more desired documents can be retrieved when larger $\beta$ values are used and what are the trade-offs.

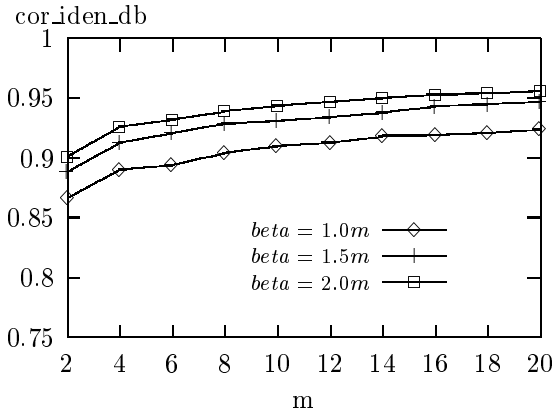


Fig. 1. Result for cor_iden_db

We first show the experimental results when combined terms are not used (see Figures 1 to 4). The results can be summarized as follows

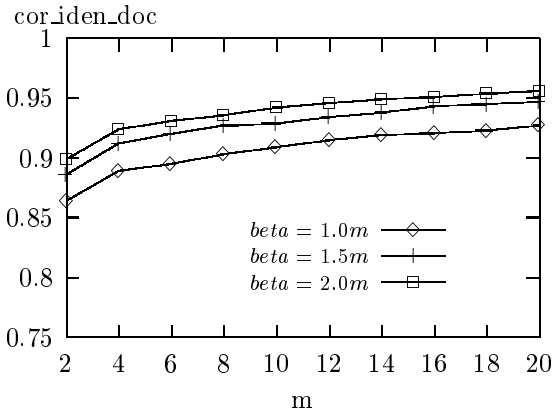


Fig. 2. Result for cor_iden_doc

(1) When $\beta = m$, as $m$ varies from 2 to 20, on the average, 86.4% to 92.3% of correct databases are identified and 86.4% to 92.7% of correct documents are identified while the number of databases searched is no more than the number of databases containing all desired documents and the number of documents retrieved is only at most 1.1% beyond the desired number of documents. The performance tends to improve for all measures when $m$ increases.
To appreciate the good performance of this method, let us consider the case when $m = 2$. The user wants to find the 2 most similar documents from more than 558,000 documents stored in 221 databases for each query. Our method searches approximately only 2 databases and transmits approximately only 2 documents to the metasearch engine for each query on the average. Yet 86.4% of the desired documents are found by our method.

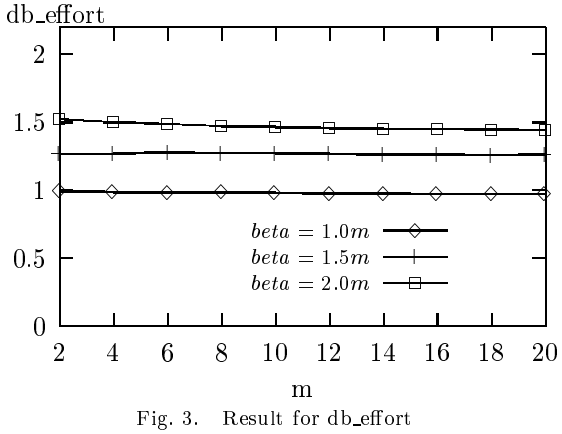


Fig. 3. Result for db_effort

(2) When $\beta$ increases, more correct databases and documents can be identified at the expense of searching more databases and retrieving more documents. Specifically, comparing with the performance of $\beta = m$, when $\beta = 1.5m$, approximately 2 more percentage points of correct databases and documents can be identified on the average while searching approximately 27% more databases and retrieving approximately 45% more documents. When $\beta = 2m$, approximately 3.5 more percentage points of correct databases and documents can be identified on the average while searching approximately 50% more databases and retrieving approximately 80% more documents.
For applications where finding a high percentage of correct documents is essential, searching a small number of additional databases and retrieving a small number of additional documents may be worthwhile.

(3) From Figure 3, we observe that *db_effort* can be less than 1. This means that the average number of databases searched can be less than the number of databases containing all the most similar documents. The reason of this phenomenon is explained as follows. Note that databases are ranked based on their ranking scores (see Formula (3)). Since the ranking may be imperfect, the databases may not be ranked optimally. As a result, a non-desired database (i.e., it does not contain one of the $m$ most similar documents), say $D'$, may be ranked ahead of some desired database(s). Let $d'$ be the most similar document in $D'$ with an actual global similarity $s'$. According to algorithm *OptDocRetrv*, when $D'$ is encountered, documents from all previously examined databases (including $D'$) that have similarities $\geq s'$ will be returned to the metasearch engine. Since $d'$ is not a desired document, its similarity $s'$ can be rather low and as a result, it is possible to find $m$ or more documents from previously examined databases with similarities $\geq s'$. This causes the retrieval algorithm to terminate prematurely without searching other databases (including desired databases ranked behind $D'$). If all the desired databases are ranked ahead of all other databases, then *db_effort* will be at least 1.

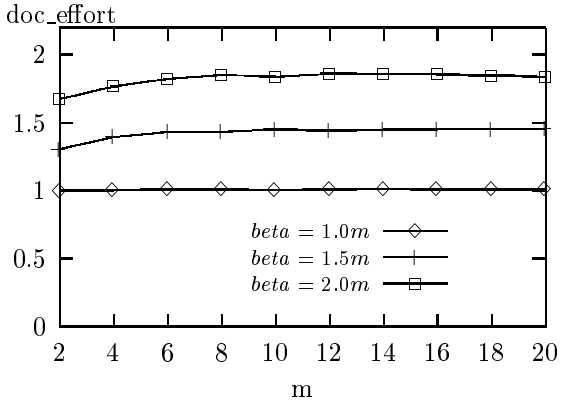


Fig. 4. Result for doc_effort

(4) From Figure 4, we observe that when $\beta = 2m$, *doc_effort* is less than 2. This is due to the fact that for a number of queries, very few documents in the entire

collection have positive similarities with these queries. In general, if for each query there are at least $\beta$ documents with positive similarities in the searched databases, then we should have $doc\_effort \geq 2$.

The above experimental results indicate that our database selection and document retrieval method can achieve close to the ideal performance as though all documents were at one site and in one database.

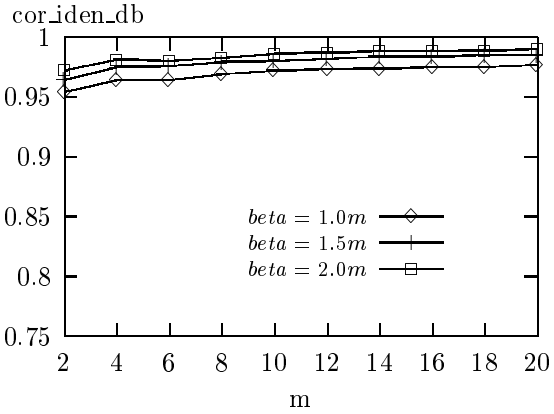


Fig. 5. Result for cor_iden_db with Combined Terms

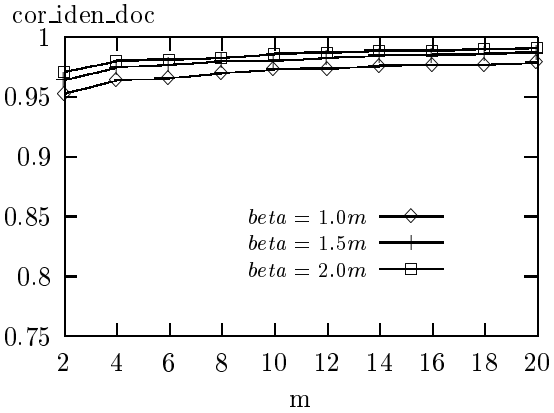


Fig. 6. Result for cor_iden_doc with Combined Terms

Figures 5-8 show the experimental results when combined terms are used. When comparing the results shown in Figures 5-6 to those shown in Figures 1-2, we can see that the use of combined terms further improves the retrieval performance. More specifically, when $\beta = m$, as $m$ varies from 2 to 20, on the average, 95.4% to 97.7% of correct databases are identified and 95.3% to 97.6% of correct documents are identified. The improvements over the cases when combined terms are not used vary from 5.3 to 8.8 percentage points for *cor_iden_db* and from 5.2 to 8.9 percentage points for *cor_iden_doc*. When $\beta = 2 * m$, as $m$ varies from 2 to 20, on the average, 97.2% to 99.0% of correct databases are identified and 97.1% to 98.7% of correct

documents are identified. This is very close to the ideal performance. Again, the performance tends to improve for all measures when $m$ increases.

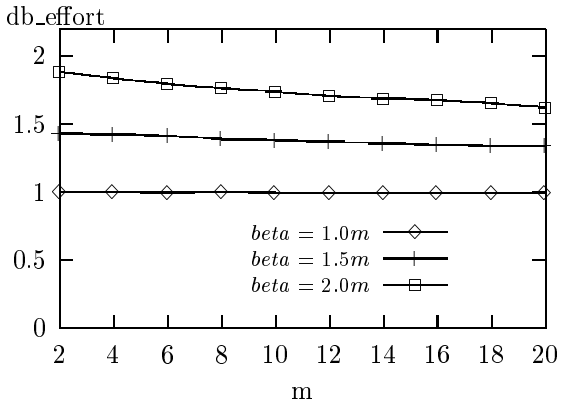


Fig. 7. Result for db_effort with Combined Terms

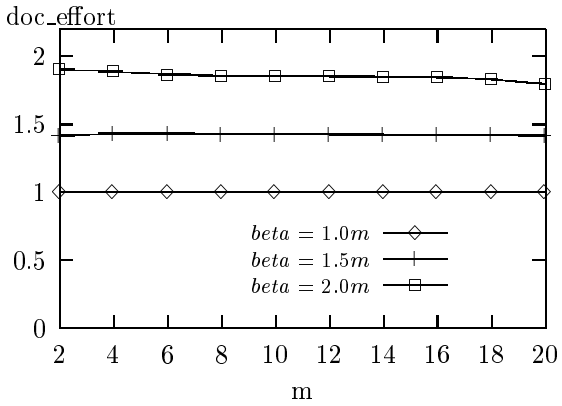


Fig. 8. Result for doc_effort with Combined Terms

From Proposition 2 we know that our proposed method will guarantee the correct retrieval of the $m$ most similar documents for single-term queries if $m \leq r$. Our experimental results indicate that our method performs very well even for multi-term queries. In general, our method tends to perform better for shorter queries. Tables 1 and 2 list the results for queries of different lengths (i.e., the number of terms in a query) when $m = 10$ and $\beta = m$. The results for other cases are very similar. The total number of queries in these tables is 885 instead of 1,000. The reason is that 115 of the original 1,000 queries do not share any common terms with the databases used in our experiments.

| *query length* | *#of queries* | cor_iden_db | cor_iden_doc | db_effort | doc_effort |
|---|---|---|---|---|---|
| 1 | 235 | 1.00 | 1.00 | 1.00 | 1.00 |
| 2 | 321 | 0.94 | 0.94 | 0.99 | 1.00 |
| 3 | 183 | 0.85 | 0.85 | 0.96 | 1.01 |
| 4 | 93 | 0.80 | 0.81 | 0.95 | 1.01 |
| 5 | 29 | 0.71 | 0.71 | 0.88 | 1.00 |
| 6 | 24 | 0.74 | 0.75 | 0.93 | 1.05 |

Table 1. Results for Queries of Different Lengths with $m = \beta = 10$ When Combined Terms Are Not Used

| *query length* | *#of queries* | cor_iden_db | cor_iden_doc | db_effort | doc_effort |
|---|---|---|---|---|---|
| 1 | 235 | 1.00 | 1.00 | 1.00 | 1.00 |
| 2 | 321 | 1.00 | 1.00 | 1.00 | 1.00 |
| 3 | 183 | 0.96 | 0.96 | 0.99 | 1.00 |
| 4 | 93 | 0.90 | 0.91 | 0.99 | 1.01 |
| 5 | 29 | 0.85 | 0.87 | 0.95 | 1.00 |
| 6 | 24 | 0.83 | 0.85 | 0.95 | 1.01 |

Table 2. Results for Queries of Different Lengths with $m = \beta = 10$ When Combined Terms Are Used

## 6. A PROTOTYPE SYSTEM

Based on the metasearch algorithm we described in the previous sections, we have implemented a demonstration prototype metasearch engine called *CSams* (Computer Science Academic MetaSearch engine; URL: http://www.data.binghamton.edu:8080/CSams/). The system has 104 databases with each containing Web pages from a Computer Science department in a US univerity. These Web pages are fetched using a Web spider (robot) we implemented. Duplicate Web pages are identified and removed. Each database is treated like a search engine in the demo system.

From the Web interface (see Figure 9), the user can enter search terms. The user can also indicate how many documents are desired, whether or not he/she wants search statistics (e.g., cor_iden_db and cor_iden_doc) to be reported, whether or not he/she wants the combined-term method to be used, and whether or not he/she wants an online dictionary to be used to replace the combined-term method (see the discussion at the end of Section 4). The dictionary employed by CSams is expanded from an online dictionary on computing (FOLDOC) from http://wombat.doc.ic.ac.uk/foldoc/contents.html. This dictionary currently has close to 7,000 two-term phrases in computer science.

After a query is processed, the resulting page will display the desired number of most similar documents found by our metasearch algorithm. For each retrieved document, its rank, document id, corresponding database id, global similarity and the URL will be displayed. In addition, when the option "Display Search Statistics" is selected, some rank numbers will be displayed in bold red color but some rank numbers will not have any color. This is explained as follows. Suppose a user wants to retrieve the 10 most similar web pages (across all databases). A number in red indicates that the corresponding web page is indeed among the actual 10

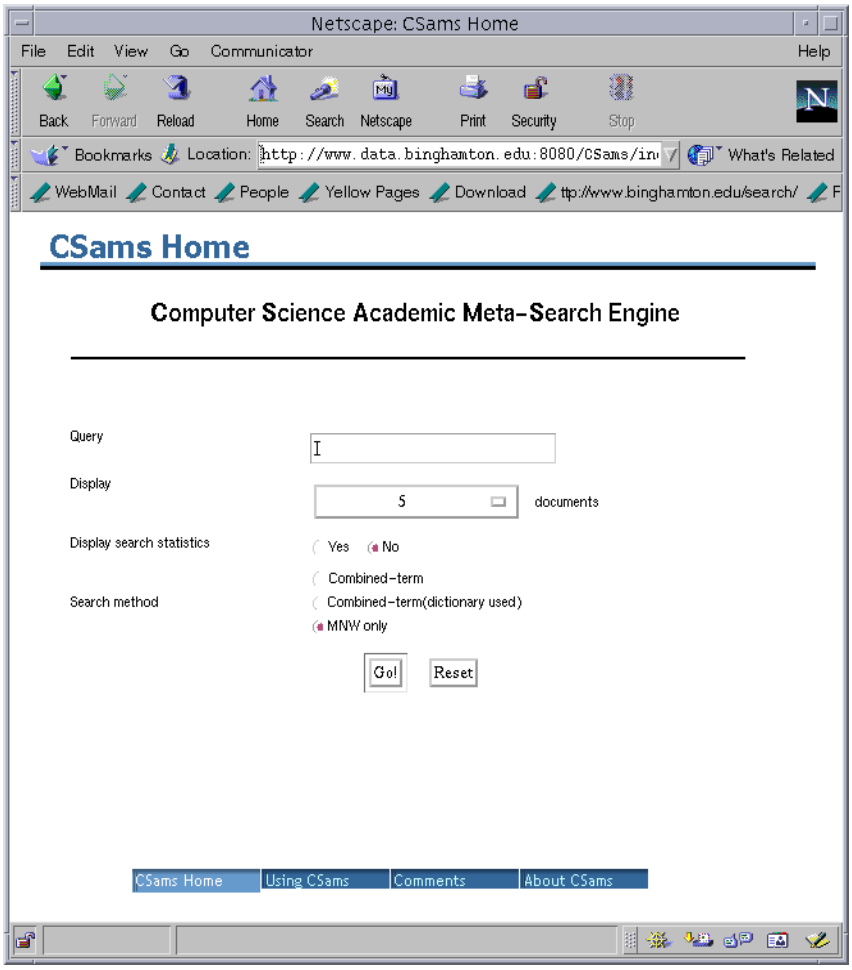


Fig. 9. Cover Page of CSams

most similar web pages to the query based on the ideal ranking. Ideal ranking is obtained based on that all documents are placed into a single collection and every document in the collection is ranked. When a query is received by CSams and when the option "Display Search Statistics" is selected, two evaluations are actually performed. The first is based on the metasearch engine approach (i.e., database selection and collection fusion are performed) and the second is based on the ideal ranking. The effectiveness of the metasearch engine is good if the rank numbers of all or nearly all returned documents are red.

## 7. CONCLUDING REMARKS

In this paper, we proposed a new method to solve the database selection problem in a large scale metasearch engine environment where tens of thousands or more of special-purpose search engines may be used. The new approach significantly improved the scalability of previous methods in both computation and space. Specifically, the new method uses a new measure to rank databases and employs an integrated database representative. By keeping only information associated with a small constant number of most important databases for each potential query term, the new representative can on one hand scale to virtually unlimited number of databases and on the other hand permits efficient selection of promising databases for any given query. In addition, a method is described to incorporate certain dependencies among terms into our solution. Experimental results indicate that very good retrieval accuracy can be achieved by the proposed solution. A prototype system based on the proposed method has been implemented (see http://www.data.binghamton.edu:8080/CSams/).

This paper focuses on finding the most similar documents for a given query. However, documents with high similarities are not necessarily relevant (useful), especially when the user query is short. This is because the particular meaning of a term in a short query often cannot be identified correctly. Several methods exist to remedy this problem. One is to incorporate the importance of a document as determined by linkages between documents (e.g., PageRank [Page et al. 1998]) with the similarity of the document to define the *degree of relevance* of the document [Yu et al. 2001]. With an appropriate database representative, it is possible to estimate the degree of relevance of the most relevant document in a database. This enables the retrieval of the most relevant documents [Yu et al. 2001]. Another method is to associate databases with concepts [Fan and Gauch 1999; Ipeirotis et al. 2001; Meng et al. 2001; Wang et al. 2000]. When a query is received by the metasearch engine, it is first mapped to a number of appropriate concepts and then those databases associated with the mapped concepts are used for database selection. The concepts associated with a database/query are used to provide some contexts for terms in the database/query. As a result, the meanings of terms can be more accurately determined

We are currently working on incorporating the above remedies into our metasearch engine solution. Another issue we are studying is how to adopt the query sampling technique in [Callan et al. 1999] to estimate the adjusted maximum normalized weight of a term from un-cooperative search engines. A pilot study has been carried out to estimate a related statistic (the maximum normalized weight) and the preliminary results indicate that the technique is promising [Liu et al. 2001a].

ACKNOWLEDGMENTS

This work is supported in part by the following NSF grants: IIS-9902792, IIS-9902872, EIA-9911099, CCR-9816633 and CCR-9803974. We would like to thank L. Gravano and H. Garcia-Molina for providing us with the set of Internet queries used in the experiments. We also would like to thank the anonymous reviewers for their valuable suggestions.

REFERENCES

Arms, W., Bowman, C., Fuhr, N., Gravano, L., Kapidakis, S., Kovacs, L., Lagoze, C., LeVan, B., Papazoglou, M., and Smeaton, A. 1999. *Resource Discovery in a Globally-Distributed Digital Library.* Digital Library Collaborative Working Groups Report, http://www.iei.pi.cnr.it/DELOS/NSF/resourcediscovery.htm.

Baumgarten, C. 1997. A probabilistic model for distributed information retrieval. In *Proceedings of the ACM SIGIR Conference, Philadelphia, PA* (July 1997), pp. 258–266.

Baumgarten, C. 1999. A probabilistic solution to the selection and fusion problem in distributed information retrieval. In *Proceedings of the ACM SIGIR Conference, Berkeley, CA* (August 1999), pp. 246–253.

Bergman, M. 2000. *The Deep Web: Surfacing the Hidden Value.* BrightPlanet, www.completeplanet.com/Tutorials/DeepWeb/index.asp.

Bharat, K. and Broder, A. 1998. A technique for measuring the relative size and overlap of public web search engines. In *Proceedings of the Seventh World Wide Web Conference, Brisbane, Australia* (April 1998), pp. 379–388.

Callan, J., Connell, M., and Du, A. 1999. Automatic discovery of language models for text databases. In *Proceedings of the ACM SIGMOD Conference, Philadelphia, PA* (June 1999), pp. 479–490.

Callan, J., Lu, Z., and Croft, W. 1995. Searching distributed collections with inference networks. In *Proceedings of the ACM SIGIR Conference, Seattle* (1995), pp. 21–28.

Dreilinger, D. and Howe, A. 1997. Experiences with selecting search engines using metasearch. *ACM Transactions on Information Systems 15*, 3 (July), 195–222.

Fan, Y. and Gauch, S. 1999. Adaptive agents for information gathering from multiple, distributed information sources. In *Proceedings of the 1999 AAAI Symposium on Intelligent Agents in Cyberspace, Stanford University* (March 1999), pp. 40–46.

French, J., Powell, A., Callan, J., Viles, C., Emmitt, T., Prey, K., and Mou, Y. 1999. Comparing the performance of database selection algorithms. In *Proceedings of the ACM SIGIR Conference, Berkeley, CA* (August 1999), pp. 238–245.

French, J., Powell, A., and Viles, C. 1998. Evaluating database selection techniques: a testbed and experiment. In *Proceedings of the ACM SIGIR Conference, Melbourne, Australia* (August 1998), pp. 121–129.

Gauch, S., Wang, G., and Gomez, M. 1996. Profusion: Intelligent fusion from multiple, distributed search engines. *Journal of Universal Computer Science 2*, 9, 637–649.

Gravano, L. and Garcia-Molina, H. 1995. Generalizing gloss to vector-space databases and broker hierarchies. In *Proceedings of the International Conferences on Very Large Data Bases, Zurich, Switzerland* (September 1995), pp. 78–89.

Gravano, L. and Garcia-Molina, H. 1997. Merging ranks from heterogeneous internet sources. In *Proceedings of the International Conferences on Very Large Data Bases, Athens, Greece* (August 1997), pp. 196–205.

Hawking, D. and Thistlewaite, P. 1999. Methods for information server selection. *ACM Transactions on Information Systems 17*, 1 (January), 40–76.

Ipeirotis, P., Gravano, L., and Sahami, M. 2001. Probe, count, and classify: Categorizing hidden-web databases. In *Proceedings of the ACM SIGMOD Conference, Santa Barbara* (2001), pp. 67–78.

Jansen, B., Spink, A., Bateman, J., and Saracevic, T. 1998. Real life information retrieval: A study of user queries on the web. *ACM SIGIR Forum 32*, 1, 5–17.

KIRK, T., LEVY, A., SAGIV, Y., AND SRIVASTAVA, D. 1995. The information manifold. In *AAAI Spring Symposium on Information Gathering in Distributed Heterogeneous Environments* (1995).

KIRSCH, S. 1998. Internet search: Infoseek's experiences searching the internet. *ACM SIGIR Forum 32*, 2, 3–7.

LAWRENCE, S. AND LEE GILES, C. 1998a. Inquirus, the neci meta search engine. In *Proceedings of the Seventh International World Wide Web Conference, Brisbane, Australia* (April 1998), pp. 95–105.

LAWRENCE, S. AND LEE GILES, C. 1998b. Searching the world wide web. *Science 280*, 98–100.

LAWRENCE, S. AND LEE GILES, C. 1999. Accessibility of information on the web. *Nature 400*, 107–109.

LIMA, E. AND PEDERSEN, J. 1999. Phrases recognition and expansion for short, precision-biased queries based on a query log. In *Proceedings of the ACM SIGIR Conference, Berkeley, CA* (August 1999), pp. 145–152.

LIU, K., YU, C., AND MENG, W. 2001a. Discovering the representative of a search engine. In *Technical Report, DePaul University* (2001).

LIU, K., YU, C., MENG, W., WU, W., AND RISHE, N. 2001b. A statistical method for estimating the usefulness of text databases. *IEEE Transactions on Knowledge and Data Engineering (to appear)*.

LIU, L. 1999. Query routing in large-scale digital library systems. In *Proceedings of the IEEE International Conference on Data Engineering, Sydney, Australia* (March 1999), pp. 154–163.

MANBER, U. AND BIGOT, P. 1997. The search broker. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, California* (December 1997), pp. 231–239.

MENG, M., LIU, K., YU, C., WANG, X., CHANG, Y., AND RISHE, N. 1998. Determine text databases to search in the internet. In *Proceedings of the International Conferences on Very Large Data Bases, New York City* (August 1998), pp. 14–25.

MENG, M., LIU, K., YU, C., WU, W., AND RISHE, N. 1999a. Estimating the usefulness of search engines. In *Proceedings of the IEEE International Conference on Data Engineering, Sydney, Australia* (March 1999), pp. 146–153.

MENG, W., WANG, W., SUN, H., AND YU, C. 2001. Concept hierarchy based text database categorization. *International Journal on Knowledge and Information Systems (to appear)*.

MENG, W., YU, C., AND LIU, K. Building effective and efficient metasearch engines. *ACM Computing Surveys (to appear)*.

MENG, W., YU, C., AND LIU, K. 1999b. Detection of heterogeneities in a multiple text database environment. In *Proceedings of the Fourth IFCIS Conference on Cooperative Information Systems, Edinburgh, Scotland* (September 1999), pp. 22–33.

PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. 1998. The pagerank citation ranking: Bring order to the web. In *Technical Report, Stanford University* (1998).

SALTON, G. AND MCGILL, M. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

SELBERG, E. AND ETZIONI, O. 1995. Multi-service search and comparison using the metacrawler. In *Proceedings of the Fourth World Wide Web Conference, Boston, Massachusetts* (December 1995), pp. 195–208.

SELBERG, E. AND ETZIONI, O. 1997. The metacrawler architecture for resource aggregation on the web. *IEEE Expert 12*, 1, 8–14.

SPARCK JONES, K. 1972. Statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation 28*, 1, 11–20.

SUGIURA, A. AND ETZIONI, O. 2000. Query routing for web search engines: architecture and experiments. In *Proceedings of the Ninth World Wide Web Conference, Amsterdam* (May 2000), pp. 417–429.

VOORHEES, E., GUPTA, N., AND JOHNSON-LAIRD, B. 1995. Learning collection fusion strategies. In *Proceedings of the ACM SIGIR Conference, Seattle* (July 1995), pp. 172–179.

WANG, W., MENG, W., AND YU, C. 2000. Concept hierarchy based text database categorization in a metasearch engine environment. In *Proceedings of the First International Conference on Web Information Systems Engineering, Hong Kong* (June 2000), pp. 283–290.

XU, J. AND CALLAN, J. 1998. Effective retrieval with distributed collections. In *Proceedings of the ACM SIGIR Conference, Melbourne, Australia* (1998), pp. 112–120.

XU, J. AND CROFT, B. 1999. Cluster-based language models for distributed retrieval. In *Proceedings of the ACM SIGIR Conference, Berkeley, California* (August 1999), pp. 254–261.

YU, C., LIU, K., MENG, W., WU, Z., AND RISHE, N. A methodology for retrieving text documents from multiple databases. *IEEE Transactions on Knowledge and Data Engineering (to appear).*

YU, C., LIU, K., WU, M., W., W., AND RISHE, N. 1999a. Finding the most similar documents across multiple text databases. In *Proceedings of the IEEE Conference on Advances in Digital Libraries, Baltimore, Maryland* (May 1999), pp. 150–162.

YU, C. AND MENG, W. 1998. *Principles of Database Query Processing for Advanced Applications.* Kaufmann Publishers, San Francisco, CA.

YU, C., MENG, W., LIU, K., WU, W., AND RISHE, N. 1999b. Efficient and effective metasearch for a large number of text databases. In *Proceedings of the Eighth ACM International Conference on Information and Knowledge Management, Kansas City* (November 1999), pp. 217–224.

YU, C., MENG, W., WU, W., AND LIU, K. 2001. Efficient and effective metasearch for text databases incorporating linkages among documents. In *Proceedings of the ACM SIGMOD Conference, Santa Barbara, CA* (May 2001), pp. 187–198.

YUWONO, B. AND LEE, D. 1997. Server ranking for distributed text resource systems on the internet. In *Proceedings of the 5th International Conference On Database Systems For Advanced Applications, Melbourne, Australia* (April 1997), pp. 391–400.