Towards a Highly-Scalable and Effective Metasearch Engine*

Zonghuan Wu[1], Weiyi Meng[1], Clement Yu[2], Zhuogang Li[1]
[1]Department of Computer Science
SUNY at Binghamton, Binghamton, NY 13902, meng@cs.binghamton.edu
[2]Department of Computer Science
University of Illinois at Chicago, Chicago, IL 60607, yu@eecs.uic.edu

ABSTRACT

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. In this paper, we propose 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, our 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.

Keywords

Metasearch Engine, Resource Discovery, Database Selection, Distributed Text Database

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 [16]. 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 [11].

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 [10]. For example, the coverage of the Web by each of them is limited [14, 16] 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. More and more people are having doubt about the scalability of the centralized search engine technology for searching the entire Web.

On the other hand, there are tens of thousands or more of special-purpose local search engines that focus on documents in confined domains such as documents in an organization or of a specific subject area. 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. There is reason to believe that all these special-purpose search engines combined together can provide a better coverage of the Web than a few major search engines combined. Thus, an alternative approach for providing the search capability for the entire Web is to combine all these special-purpose search engines. This is the *metasearch engine* approach. 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

*This work is supported in part by the following NSF grants: IIS-9902792, IIS-9902872, EIA-9911099, CCR-9816633 and CCR-9803974.

WWW10, May 1-5, 2001, Hong Kong.
ACM 1-58113-348-0/01/0005.

collects (sometimes, reorganizes) the results from its local search engines. In addition to the potential of increased search coverage of the Web, another advantage of such a metasearch engine over a general-purpose search engine is that it is easier to keep index data up to date as each local search engine covers only a small portion of the Web. 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 serious challenges to implement an effective and efficient metasearch engine. Among the main challenges, the *database selection problem* is to identify, for a given user query, the local search engines that are likely to contain useful documents for the query. The 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 [22]. 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 [32, 35, 33] 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 to rank databases. This technique is based on a novel database representative that has the following features. First, it is highly scalable in terms of both computation and space. 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. Our experimental results indicate that our new method is not only very scalable but also very accurate. 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., [1, 4, 6, 8, 9, 17, 19, 20, 21, 26, 27, 28, 32, 37]).

For database selection, most approaches rank the databases for a given query based on certain usefulness measures. For example, gGlOSS uses the sum of document similarities that are higher than a threshold [8], CORI Net uses the probability that a database contains relevant documents due to the terms in a given query [4], D-WISE uses the sum of weighted document frequencies of query terms [37], Q-Pilot uses the dot-product similarity between an expansion query and a database description [27], and one of our approaches uses the expected number of documents whose similarities are higher than a threshold [20]. All these database ranking methods are heuristics as they are not designed to produce optimal orders based on some optimality criteria. In [35, 33], 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 [12]. However, [12] considered only the databases and queries that are for structured data. In contrast, unstructured text data are considered in [35, 33].

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 [7], and MRDD [28]), 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 [9, 20], 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 [15] 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 [35, 33] utilizes an estimated 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 [35, 33]. 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 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 [24] 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 tens of thousands or more 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 there are a large number of databases. Second, based on existing methods [4, 8, 17, 21, 37], 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 tens of thousands or more local search engines, the total size of these representatives could be tens or even thousands of times of that of an average database. Consequently, the representatives may have to be stored on slower storage devices, 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 [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, the performance deteriorates substantially. In contrast, our method shows good experimental results in both situations (see Section 5). In addition, our theory differs from that given in [1] substantially.

In [30], experimental results were given to demonstrate that it was possible to retrieve documents in distributed environments with essentially the same effectiveness as if all data were at one site. However, the results depended on the existence of a *training collection* which has similar coverage of subject matters and terms as the collection of databases to be searched. In the Internet environment where data are highly heterogeneous, it is unclear whether such a training collection can in fact be constructed. Even if such a collection can be constructed, the storage penalty could be very high in order to accommodate the heterogeneity. In [31], 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 [23] 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* [25], 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 (tf)* of the term in the query (document) and the *document frequency (df)* of the term [25, 34]. The weight factor based on the tf information is the *tf weight* and the weight 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 [25, 34]. 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 dfs 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 [3] 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 $|d|$ 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 [32, 35]. 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 whenever the m most similar documents to the query are desired, it is sufficient to examine the first k databases. Note that the ordering of the databases depends on the query q. For ease of presentation, we shall assume that all similarities of the documents with the query are distinct so that the set of the m most similar documents to the query is unique.

PROPOSITION 1. [33] 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 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 [35, 33]. This algorithm is sketched as follows. Suppose the first s databases have been selected (s is 2 initially). Each of these selected search engines returns the actual global similarity of the most similar document to the metasearch engine which computes the minimum, 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 [33] that if the databases are ranked optimally, then algorithm *OptDocRetrv* will guarantee the retrieval of all the m desired 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 result in the local similarity of a document being different from the global similarity of the document. This problem has been addressed in [20, 32] 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 and obtain the global similarity of its most similar document. Instead, for each database, we need to **estimate** the required similarity. In [33, 35], 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. Previous methods tried to estimate the similarity of the most similar document in each database directly. 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 measure can be obtained using less information than estimating the similarity of the most similar document. Our novel integrated representative makes obtaining the measure highly scalable. 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.

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. $mnw_{i,j}$ is defined as follows. First, if $d = (d_1, ..., d_i, ..., d_n)$ is 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, where $|d|$ 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 having 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 [33]:

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

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 of 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 much 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), $gidf_i * mnw_{i,j}$ is likely to dominate $\sum_{k=1,\, k \ne i}^{n} gidf_k * anw_{k,j}$, especially when the number of terms, n, in a query is small (which is typically true in the Internet environment [11, 13]). In other words, whether database D_j is going to be ranked high (i.e., whether it's going to have a large similarity of the most similar document) with respect to a given query q is largely determined by the value of $\max_{1 \le i \le n} \left\{ q_i * gidf_i * mnw_{i,j} \right\}$.

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 $t_i, i = 1, ..., k$, be the k terms in a query q. We define the *ranking score* (or **rs** for short) of database D_j with respect to q as follows:

$$rs(q, D_j) = \max_{1 \le i \le k} \{ q_i * am_{i,j} \} \quad (3)$$

The ranking score defined above will be our new measure to rank databases. Note that the above formula 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 [11, 13]), 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 documents in a database are accessible, then the needed statistical information can be easily obtained. There are several situations in which 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* [5] could be adopted to estimate the needed statistics. For the rest of this 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** algo-

rithm reviewed in Section 3 can be used to select databases and retrieve documents.

The above database representative stores only one piece of information per term and is already more scalable than most existing database selection approaches (e.g., [4, 8, 17, 21]) 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 feasible 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 site 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. Thus, for a query having k terms, at most $k * r$ ranking scores are computed. This is independent of the number of databases. In the Internet environment, k is usually very small ($k = 2.2$ on the average [13]). 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 (a small u, say 20, will be sufficient for most users if relevant search engines can be selected), then r can be set to u. 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 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 practice, 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 this 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 [11]), 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 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.

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 than that expected if the two terms were independently distributed in the database. One way to remedy this assumption is to recognize dependent adjacent terms and treat them as new terms. This is similar to recognizing phrases. Due to space limitation, discussions on recognizing dependent adjacent terms and incorporating them into the above approach will not be provided in this paper (interested readers please see [29]).

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.

1,000 Internet queries by real users are used in our experiments. These queries were collected at Stanford University for evaluating the performance of the gGlOSS database selection method [8]. 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 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 [11]. Another observation made in [11] 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 effectiveness (quality) of retrieval while the last two measures reflect 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 **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**. 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 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 β. 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 β to control when to terminate algorithm *OptDocRetrv*. Specifically, β 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 β, we would like to see whether more desired documents can be retrieved when larger β values are used and what are the trade-offs.

The experimental results are shown in Figures 1 to 4.

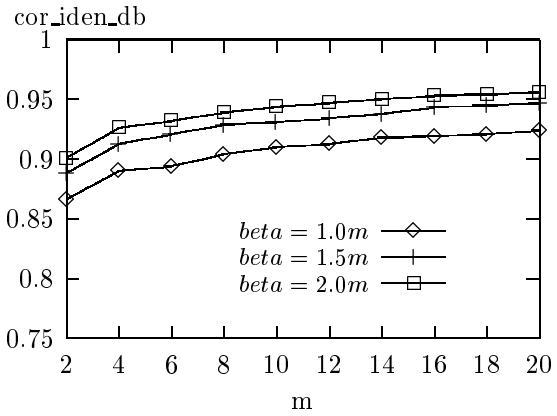

Figure 1: Result for cor_iden_db

The experimental results can be summarized as follows.

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.

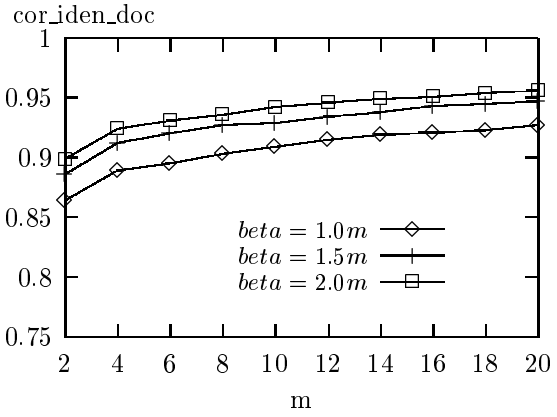

Figure 2: Result for cor_iden_doc

2. When β 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 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 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 documents may be worthwhile.

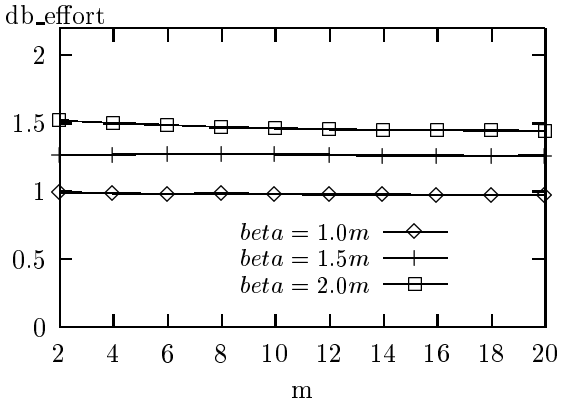

Figure 3: Result for db_effort

3. From Figure 3, we observe that *db_effort* can 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 most similar document in D' with 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.

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 β documents with positive similarities in the searched databases, then we should have $doc\_effort \geq 2$.

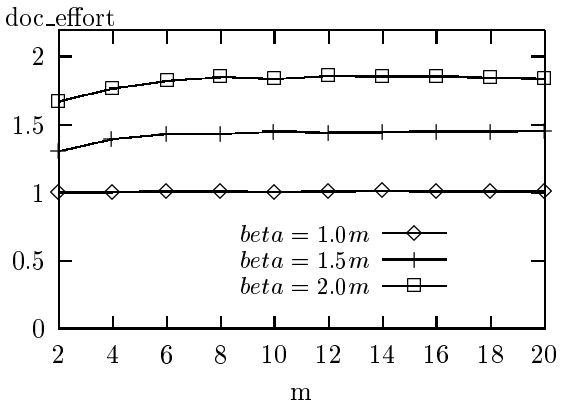

Figure 4: Result for doc_effort

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.

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. Table 1 lists 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 Table 1 is 885 instead of 1,000. The reason is that 115 of the orig-

inal 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 when $m = \beta = 10$

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://slate.cs.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, the user can enter search terms. The user can also indicate how many documents are desired, whether or not you want search statistics (e.g., cor_iden_db and cor_iden_doc) to be reported, whether or not you want the combined-term method (not discussed in this paper) be used. 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 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 can be 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 a 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 metasearch engine environment. The new approach significantly improved the scalability of previous methods in both computation and space. Specifically, the new method uses an integrated database representative that can on one hand scale to unlimited number of databases and on the other hand permits efficient selection of promising databases for any given query. 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://slate.cs.binghamton.edu:8080/CSams/).

The retrieval accuracy can be further improved by taking into consideration certain dependencies among terms into our solution. For example, we can recognize dependent adjacent term pairs and treat each such a pair of terms as a new term. To incorporate the new terms into our solution, we need to compute and store the r largest adjusted maximum normalized weights for each new term in the integrated database representative. Our experiments [29] indicate that, with the new terms taken into account, when the number of desired documents 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 not considering the new terms 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*.

One of the issues we are currently studying is how to adopt the query sampling technique proposed in [5] to estimate the adjusted maximum normalized weight of a term from an un-cooperative search engine. A pilot study has been carried out to estimate a related statistic (i.e., the maximum normalized weight) and preliminary results indicate that the technique is promising [18].

Acknowledgement: We would like to thank L. Gravano and H. Garcia-Molina for providing us with the set of Internet queries used in the experiments.

8. REFERENCES

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

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

[3] K. Bharat, and A. Broder. *A Technique for Measuring the Relative Size and Overlap of Public Web Search Engines.* 7th WWW Conference, April 1998.

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

[5] J. Callan, M. Connell, and A. Du. *Automatic Discovery of Language Models for Text Databases.* ACM SIGMOD, 1999.

[6] D. Dreilinger, and A. Howe. *Experiences with Selecting Search Engines Using Metasearch.* ACM TOIS, 15(3), July 1997, pp. 195-222.

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

[8] L. Gravano, and H. Garcia-Molina. *Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies.* VLDB, 1995.

[9] L. Gravano, and H. Garcia-Molina. *Merging Ranks from Heterogeneous Internet Sources.* VLDB, 1997.

[10] D. Hawking, and P. Thistlewaite. *Methods for Information Server Selection*. ACM Transactions on Information Systems, 17(1), January 1999.

[11] 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.

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

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

[14] S. Lawrence, and C. Lee Giles. *Searching the World Wide Web*. Science, 280, April 1998.

[15] S. Lawrence, and C. Lee Giles. *Inquirus, the NECi Meta Search Engine*. Seventh International World Wide Web Conference, 1998.

[16] S. Lawrence, and C. Lee Giles. *Accessibility of Information on the Web*. Nature, 400, July 1999.

[17] K. Liu, C. Yu, W. Meng, W. Wu, and N. Rishe. *A Statistical Method for Estimating the Usefulness of Text Databases*. IEEE TKDE (to appear).

[18] K. Liu, C. Yu, and W. Meng. *Discovering the Representative of a Search Engine*. Technical Report, DePaul University, 2000.

[19] U. Manber, and P. Bigot. *The Search Broker*. USENIX Symposium on Internet Technologies and Systems, 1997.

[20] W. Meng, K. Liu, C. Yu, X. Wang, Y. Chang, N. Rishe. *Determine Text Databases to Search in the Internet*. VLDB, 1998.

[21] W. Meng, K. Liu, C. Yu, W. Wu, and N. Rishe. *Estimating the Usefulness of Search Engines*. ICDE, 1999.

[22] W. Meng, C. Yu, and K. Liu. *Detection of Heterogeneities in a Multiple Text Database Environment*. CoopIS, 1999.

[23] W. Meng, C. Yu, and K. Liu. *Building Effective and Efficient Metasearch Engines*. Submitted to ACM Computing Surveys (under revision).

[24] Resource Discovery in a Globally-Distributed Digital Library. Working Group Report, 1999 (http://www.iei.pi.cnr.it/ DELOS/ NSF/ resourcediscovery.htm)

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

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

[27] A. Sugiura, and O. Etzioni. *Query Routing for Web Search Engines: Architecture and Experiments*. WWW9 Conference, 2000.

[28] E. Voorhees, N. Gupta, and B. Johnson-Laird. *The Collection Fusion Problem*. TREC-3, 1995.

[29] Z. Wu, W. Meng, C. Yu, and Z. Li. *Towards a Highly-Scalable and Effective Metasearch Engine*. Technical Report, Dept. of CS, SUNY at Binghamton, 2001.

[30] J. Xu, and J. Callan. *Effective Retrieval with Distributed Collections*. ACM SIGIR Conference, pp. 112-120, Melbourne, Australia, 1998.

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

[32] C. Yu, K. Liu, W. Wu, W. Meng, and N. Rishe. *Finding the Most Similar Documents across Multiple Text Databases*. IEEE ADL'99, 1999.

[33] C. Yu, K. Liu, W. Meng, Z. Wu, and N. Rishe. *A Methodology for Retrieving Text Documents from Multiple Databases*. IEEE Transactions on Knowledge and Data Engineering (to appear).

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

[35] C. Yu, W. Meng, K. Liu, W. Wu, and N. Rishe. *Efficient and Effective Metasearch for a Large Number of Text Databases*. CIKM'99, 1999.

[36] C. Yu, W. Meng, K. Liu, W. Wu, and N. Rishe. *Efficient and Effective Metasearch for Text Databases Incorporating Linkages among Documents*. ACM SIGMOD Conference, 2001 (to appear).

[37] B. Yuwono, and D. Lee. *Server Ranking for Distributed Text Resource Systems on the Internet*. DASFAA'97, 1997.