# Determining Text Databases to Search in the Internet

Weiyi Meng[1], King-Lup Liu[2], Clement Yu[2], Xiaodong Wang[1], Yuhsi Chang[1], Naphtali Rishe[3]

[1] Dept. of Computer Science, SUNY – Binghamton, Binghamton, NY 13902, meng@cs.binghamton.edu

[2] Dept. of EECS, University of Illinois, Chicago, IL 60607, {yu, kliu}@eecs.uic.edu

[3] School of Computer Science, Florida International Univ., Miami, FL 33199, rishen@cs.fiu.edu

## Abstract

Text data in the Internet can be partitioned into many databases naturally. Efficient retrieval of desired data can be achieved if we can accurately predict the usefulness of each database, because with such information, we only need to retrieve potentially useful documents from useful databases. In this paper, we propose two new methods for estimating the usefulness of text databases. For a given query, the usefulness of a text database in this paper is defined to be the number of documents in the database that are sufficiently similar to the query. Such a usefulness measure enables naive-users to make informed decision about which databases to search. We also consider the collection fusion problem. Because local databases may employ similarity functions that are different from that used by the global database, the threshold used by a local database to determine whether a document is potentially useful may be different from that used by the global database. We provide techniques that determine the best threshold for a given local database.



**Proceedings of the 24th VLDB Conference**
**New York, USA, 1998**

## 1 Introduction

How to find desired data in the Internet in a timely manner is a problem with wide interest. In this paper, we focus on the retrieval of text data. One popular way to find desired information in the Internet is to query a search engine. To support *querying*, an inverted file index is usually created for all documents. However, since the amount of data accessible through the Internet is huge and is increasing at a very high rate, it is not realistic to use a single index for all data in the Internet.

Data in the Internet are often organized into databases naturally. For example, all posts associated with a newsgroup can be considered as a database, so are all html files associated with an organization. Larger (global) databases can be constructed from many smaller (local) databases. Typically, a global database does not maintain its own index. A global database is actually an interface (we will use the phrase *global interface* in this paper) created to provide uniform and integrated access to underlying local databases. When a global interface receives a user query, it first passes the query to its local databases, then merges the results from local databases and finally presents the merged result to the user.

Many global interfaces have been built but most of them pass each query to all underlying databases indiscriminately (e.g., MetaCrawler [SeEt95, SeEt97] and NCSTRL [NCS]). If a local database contains no useful documents to a query, then passing the query to the database causes unnecessary network traffic and local resource waste. A better approach is to first identify those local databases that are most likely to provide useful results to the query and then search only the identified local databases for desired documents. Examples of systems that employ this approach are WAIS [KaMe91], ALIWEB [Kost94], gGlOSS [GrGM95a], SavvySearch [HoDr97] and D-WISE [YuLe97]. With such an approach, the problem of processing a user query consists of the following two subproblems:

1. Select local databases that need to be searched and estimate the number of globally most similar documents in each local database.

2. Decide which documents from each selected local database to retrieve. Even when the number of globally most similar documents in a local database with respect to a query can be estimated correctly, the documents to be retrieved have yet to be determined, due to possibly different similarity functions or term weights used by the global interface and by the local database. The problem of deciding which documents should be retrieved from local databases is known as the *collection fusion problem.*

In this paper, we attack both of the above two subproblems, namely, the database selection problem and the collection fusion problem.

The current solutions to the database selection problem is to rank all underlying databases for each query using some metadata that describe the contents of each database. Often, the ranking is based on some measure which ordinary users may not be able to utilize to fit their needs. In other words, for a given query, the current approach can tell the user to some degree of accuracy which database is likely to be the most useful, the second most useful, etc. While such a rank can be helpful, it cannot tell the user *how useful* any particular database is. In this paper, the usefulness of a database to a query is defined to be the number of documents in the database that have high potentials to be useful to the query, that is, the *similarities* between the query and the documents as measured by a certain global similarity function are higher than a specified threshold[1]. This usefulness can be defined precisely as follows:

$$usefulness(T, q, D) = |\{d | d \in D \text{ and } sim(q, d) > T\}| \quad (1)$$

where T is a threshold, D is a database, $sim(q, d)$ is the similarity (closeness) between a query $q$ and a document $d$ in D, and $|X|$ denotes the cardinality of set $X$. A query is simply a set of words submitted by a user. It is transformed into a vector of terms by eliminating non-content words, stemming, etc [SaMc83]. In practice, users may not know how to relate thresholds to the number of documents they like to retrieve. Therefore, users are more likely to tell the system the number of most similar documents (to their query) they like to retrieve directly. Such a number can be translated into a threshold by computing the usefulnesses of each database in decreasing thresholds.

For the collection fusion problem, most existing global interfaces do not guarantee that all (or a high percentage of all) globally most similar documents be retrieved from each local database (see next section for more discussion). In this paper, we also study the problem of guaranteeing all globally most similar documents from each local database be retrieved. Such a guarantee could be important for legal and medical applications. Suppose the global interface sets a threshold T and uses a global similarity function G such that any document $d$ satisfying $G(q, d) > T$ is to be retrieved, where $q$ is the user query. A local database may use a different similarity function, say L. The problem is to determine a proper threshold $T'$ used by the local database such that all globally most similar documents which can be found in the database can be retrieved using L, i.e., if $G(q, d) > T$, then $L(q, d) > T'$ and $T'$ is as large as possible.

The contributions of this paper are:

- We provide two new estimation methods to estimate the usefulness of a database. The methods have solid theoretical foundations. Experimental results are obtained to demonstrate the superiority of these methods over existing methods.

- We provide two techniques to obtain the best local threshold (i.e., the largest $T'$) while guaranteeing all globally most similar documents be retrieved from the local database. By an example used by others, we show that the threshold computed by one of our techniques is much better than the one computed using an earlier methodology.

- We provide two techniques to retrieve documents in a local database when the similarity functions for both the global interface and the local database are the same popular *Cosine* function [SaMc83] (although the functions are identical, the weight of a term which depends on the number of documents having the term may change from the local database to the global interface). The first one modifies the query so that the local database computes the global similarity for each local document. The second one computes an optimal local threshold.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents our methods for estimating the usefulness of text

[1] Ideally, for a given query, the usefulness of a database should be defined as the number of *relevant* documents in the database. However, the relevance of documents is highly subjective as it can only be determined by the issuer of the query. As a result, this definition is not suitable for our problem in practice, i.e., the global interface cannot determine whether a document is relevant. Similarity-based measure is also used in [GrGM95a].

databases as well as experimental results. The problem of how to find globally most similar documents for any given query from multiple text databases will be studied in Section 4. Section 5 concludes the paper.

## 2 Related Work

To be able to identify useful databases to a query, the global interface must keep some characteristics information about each database. We call such information the *representative* of a database. Different database selection methods can be developed based on the representatives used.

Database selection has been employed by several systems. However, the database representatives used in most systems cannot be used to estimate the number of globally most similar documents in each local database [CLBC95, Kost94, KaMe91, MaBi97, YuLe97]. In gGlOSS [GrGM95a], each database is represented by the document frequency of each term and the sum of the weights of each term over all documents in the database. The database usefulness used in gGlOSS is different from the one defined in Formula (1). However, the representative of gGlOSS can be used to estimate the number of useful documents in a database [GrGM95b]. The methods used in gGlOSS are very different from ours. The estimation methods employed in [GrGM95a, GrGM95b] are based on two very restrictive assumptions. One is the *high-correlation assumption* (for any given database, if query term j appears in at least as many documents as query term k, then every document containing term k also contains term j) and the other is the *disjoint assumption* (for a given database, for all term j and term k, the set of documents containing term j is disjoint with the set of documents containing term k). Due to the restrictiveness of the above assumptions, the estimates provided by these two methods are not accurate.

[YuLS78] proposed a method to estimate the number of useful documents in a database for the *binary and independent* case. In this case, each document $d$ is represented as a binary vector such that a 0 or 1 at the ith position indicates the absence or presence of term $t_i$ in $d$; and the occurrences of terms in different documents are assumed to be independent. This method was later extended to the *binary and dependent* case in [LaYu82], where dependencies among terms are incorporated. A substantial amount of information will be lost when documents are represented by binary vectors. As a result, it is seldom used in practice. Our proposed solutions to the usefulness estimation problem are extensions of those in [YuLS78, LaYu82] by permitting the use of arbitrary term weights in representing documents and by incorporating term dependencies.

The *collection fusion problem* has received a lot of attention recently. For a given query, most existing global interfaces, after the number of documents to retrieve from a local database is determined (let k denote the number), the global interface lets the local database retrieve the top k documents from the local database, based on the local similarity function. For example, MetaCrawler [SeEt95, SeEt97] and SavvySearch [HoDr97] let the user specify the maximum number of documents to be retrieved from each local database. D-WISE [YuLe97] and CORI net [CLBC95] retrieve proportionally more documents from databases that are ranked higher or have higher ranking scores. [TVGJ95, VGJL95] provides several learning based approaches. A problem common to all the above approaches is that none of them guarantees that all globally most similar documents from each database will be retrieved. The algorithm in [GrGM97] while guaranteeing that all globally most similar documents will be retrieved may unnecessarily retrieve many documents that are not globally most similar. Our proposed solutions in this paper aim at minimizing the number of documents that are not globally most similar to be retrieved while guaranteeing that all globally most similar documents are retrieved.

## 3 Two New Methods for Usefulness Estimation

In section 3.1, we consider a special case — the *Non-binary and Independent* case, where *non-binary* means that term weights are not limited to 0 or 1 (can be any real numbers) and *independent* means that the occurrences of different terms in each document are independent. Under two assumptions, one is the "term independence" and the other is that all documents having a term have the same weight for the term in a database, our method can accurately estimate the usefulness of a database. In section 3.2, we relieve the former assumption by incorporating term dependence into the basic solution. In section 3.3, we relieve the latter assumption by allowing dynamic adjustment to term weights. As a result, our estimation becomes more accurate. In summary, our first estimation method is for the non-binary and independent case with dynamic adjustment to term weights and other relevant information, and our second method is for the non-binary and dependent case with dynamic adjustment to term weights and other relevant information. Experimental results are reported in section 3.4. The applicability of the two methods in practice is discussed in section 3.5.

### 3.1 The Non-binary and Independent Case

Consider a database D with $m$ distinct terms. Each document $d$ in this database can be represented as a vector $d = (d_1, \ldots, d_m)$, where $d_i$ is the weight (or significance) of the $i$th term $t_i$ in representing the document, $1 \leq i \leq m$. Each query can be similarly represented. Consider query $q = (u_1, u_2, ..., u_m)$, where $u_i$ is the weight of $t_i$ in the query, $1 \leq i \leq m$. The similarity between $q$ and document $d$ can be defined as the dot product of their respective vectors, namely $sim(q, d) = u_1 * d_1 + \cdots + u_m * d_m$ (note that normalization that yields similarities between 0 and 1, for example using the *Cosine* function [SaMc83], can be incorporated by re-defining the weights $d$'s and $u$'s. Normalization is used in our experiments (see section 3.4)).

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

Consider the following generating function:

$$\prod_{i=1}^{r} [p_i * X^{w_i * u_i} + (1 - p_i)] \quad (2)$$

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

**Proposition 1**[2] Let $q$ and $D$ be defined as above. If the terms are independent and the weight of term $t_i$ whenever present in a document is $w_i$, which is given in the database representative ($1 \leq i \leq r$), then the coefficient of $X^s$ in function (2) is the probability that a document in $D$ has similarity $s$ with $q$.

**Example 1** Let $q$ = (1, 1, 1) be a query with three terms with all weights equal to 1. Suppose database $D$ has five documents and their vector representations are (only components corresponding to query terms are given): (2, 0, 2), (0, 1, 1), (2, 0, 0), (0, 0, 3) and (0, 0, 0). Namely, the first document has query term 1 and query term 3, and their corresponding weights (e.g., the numbers of occurrences of the terms in the document) are both 2. Other document vectors can be interpreted similarly. From the five documents in D, $(p_1, w_1) = (0.4, 2)$, $(p_2, w_2) = (0.2, 1)$, and $(p_3, w_3) = (0.6, 2)$ can be computed. Therefore, the corresponding generating function is:

$$(0.4 * X^2 + 0.6)(0.2 * X + 0.8)(0.6 * X^2 + 0.4) \quad (3)$$

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

Suppose after generating function (2) has been expanded and the terms with the same $X^s$ have been combined, we obtain:

$$a_1 * X^{b_1} + a_2 * X^{b_2} + \cdots + a_c * X^{b_c} \quad (4)$$

where $b_1 > b_2 > ... > b_c$. By Proposition 1, $a_i$ is the probability that a document in D has similarity $b_i$ with $q$. For a given similarity threshold $T$, let C be the largest integer to satisfy $b_C > T$. Let n be the number of documents in D. Then, the usefulness of D for query $q$ based on threshold $T$ can be estimated to be the expected number of documents in D whose similarities with query $q$ are greater than $T$:

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

**Example 2** (Continue Example 1). When formula (3) is expanded, we have:

$$0.048 * X^5 + 0.192 * X^4 + 0.104 * X^3 + 0.416 * X^2 + 0.048 * X + 0.192 \quad (6)$$

Using formula (5), the usefulness of D with respect to $q$ and $T = 2$ can be estimated as $estimate(2, q, D) = 5 * (0.048 + 0.192 + 0.104) = 1.72$. It is interesting to note that the true usefulness $usefulness(2, q, D) = 2$ since there are two documents having similarity higher than 2 with $q$ (the first and the fourth, and the similarities are 4 and 3, respectively). ■

### 3.2 The Non-binary and Dependent Case

In this case, in addition to the $p_i$'s and $w_i$'s as in the *Non-binary and Independent* case, the database representative also includes term dependency information. We use co-variances to measure the dependencies among different terms. The co-variance between term $i$ and term $j$ is denoted by $\sigma_{ij}$ and the co-variance

[2] All proofs in this paper can be found in [MLYW98].

among terms $i, j$, and $k$ is denoted by $\sigma_{ijk}$. This notation can be generalized for co-variances among any number of terms.

For query $q = (u_1, \ldots, u_r)$, let $\vec{X}$ be a vector of random variables $(X_1, X_2, \ldots, X_r)$. $\vec{X}$ maps each document $d$ in database $D$ to a binary vector $\vec{X}(d) = (X_1(d), X_2(d), \ldots, X_r(d))$, where $X_i(d) = 1$, if document $d$ has term $t_i$, and 0 otherwise $(i = 1, \ldots, r)$. Denote by $\mu_i$ the expected value of $X_i$ $(i = 1, \ldots, r)$. For terms $t_{j_1}, t_{j_2}, \ldots, t_{j_s}$, we measure their degree of dependency by the co-variance of $X_{j_1}, X_{j_2}, \ldots, X_{j_s}$, which is the expected value of the product $\prod_{i=1}^{s} (X_{j_i} - \mu_{j_i})$.

Let $P(\vec{x})$ be the probability that $\vec{X}$ maps a document in $D$ to a given binary vector $\vec{x} = (x_1, x_2, ..., x_r)$. If the terms are mutually independent, this probability, denoted by $P_0(\vec{x})$, is $\prod_{t=1}^{r} p_t^{x_t} * (1 - p_t)^{1-x_t}$. When terms are not independent, then the Bahadur-Lazarsfeld Expansion [DuHa73] can be used to derive the following expression for $P(\vec{x})$ [LaYu82]:

$$P(\vec{x}) = P_0(\vec{x}) \left( 1 + \sum_{i<j} \sigma_{ij} \frac{(x_i - p_i)(x_j - p_j)}{p_i p_j (1-p_i)(1-p_j)} \right.$$
$$+ \sum_{i<j<k} \sigma_{ijk} \frac{(x_i - p_i)(x_j - p_j)(x_k - p_k)}{p_i p_j p_k (1-p_i)(1-p_j)(1-p_k)}$$
$$\left. + \cdots + \sigma_{12\ldots r} \frac{(x_1 - p_1)\cdots(x_r - p_r)}{p_1 \cdots p_r (1-p_1)\cdots(1-p_r)} \right) \quad (7)$$

where $\sigma_{ij}$ is the co-variance of $X_i$ and $X_j$, $\sigma_{ijk}$ is the co-variance of $X_i$, $X_j$ and $X_k$, etc. Expansion (7) can be interpreted as follows. If the terms in $D$ are more or less independent, then $P(\vec{x})$ can be approximated by $P_0(\vec{x})$. Otherwise, dependencies between terms can be added for a better approximation. Usually, the more dependency information we add, the better the approximation will be. If all possible combinations of term dependencies are taken into consideration, then $P(\vec{x})$ is accurately represented.

In Proposition 1, we have shown that function (2) can be used to find the probability that a document in D has similarity $s$ with $q$ when terms are independent. It can be shown (see Proposition 2 below) that when terms are not independent, the following generating function can be used for the $P(\vec{x})$ as expressed in (7) above.

$$\prod_{t=1}^{r} [p_t * X^{w_t * u_t} + (1 - p_t)] + \sum_{i<j} \sigma_{ij} \prod_{t \neq i, t \neq j} [p_t * X^{w_t * u_t}$$
$$+ (1 - p_t)](X^{w_i * u_i} - 1)(X^{w_j * u_j} - 1)$$
$$+ \cdots + \sigma_{12\ldots r} \prod_{t=1}^{r} (X^{w_t * u_t} - 1) \quad (8)$$

**Proposition 2.** Let $q$ and D be defined as above. Based on expansion (7), if the weight of term $t_i$ whenever present in a document is $w_i$ which is given in the database representative $(1 \leq i \leq r)$, then the coefficient of $X^s$ in function (8) is the probability that a document in D has similarity $s$ with $q$.

Note that the first subexpression in (8) is identical to expression (2). In other words, expression (8) is obtained by first assuming that all terms are independent and then incorporating the dependencies among terms into the expression. If term $i$ and term $j$ are independent, then $\sigma_{ij}$ is zero. Similarly, if a set of terms are independent, then the corresponding co-variance is zero. Thus, in practice, it is usually sufficient to incorporate the $O(r)$ most significant co-variances between all term pairs for a query with $r$ distinct terms; other less significant ones can be ignored.

By expanding (8) and combining terms with the same $X^s$, a function similar to (4) can be obtained. After this, the process for deriving the formula for estimating the usefulness of a database is identical to that discussed in section 3.1.

### 3.3 Incorporating More Accurate Weight Information Dynamically

In the database representative for database D, $w_i$ is the average weight of the weights of term $t_i$ among the documents that contain $t_i$. Based on Propositions 1 and 2, generating functions (2) and (8) can accurately estimate the usefulness of D under their respective assumptions on term dependence if the weight of term $t_i$ whenever present in a document is $w_i$. This assumption about the weight of a term being uniform among documents containing the term may not be realistic. As a result, the estimated usefulness may be inaccurate.

The following impact of using the average weight of each term in the database representative on the estimation accuracy can be observed. Typically, documents whose similarities with a query exceed a large threshold must have large weights for the terms that also appear in the query. However, when the average weights are used, documents having large weights on query terms may fail to be recognized as the average operation brings down weights that are above the average to compensate weights that are below the average. To overcome the bad impact of using the average weights on the estimation accuracy for large thresholds, we also store, for each term $t_i$, the *standard deviation* (denoted $\sigma_i$) of the weights of $t_i$ in the set of documents containing $t_i$. With the deviation added, the representative of each database will be a set of triplets $(w_i, p_i, \sigma_i)$. If dependence information among terms are to be incorporated, then some co-variances among terms will also be in the database representative.

The idea is to use the standard deviation of each term to dynamically adjust (i.e., increase) the average weight of the term when large thresholds are used. For larger thresholds, larger increases should be made. The following formula for obtaining the new weight, $w_i'$, from the original average weight, $w_i$, and the standard deviation, $\sigma_i$, can be used:

$$w_i' = w_i + \frac{c * T}{T_{max}} * \sigma_i \qquad (9)$$

where T is the threshold used and $T_{max}$ is the maximum value that should be used as a threshold for the query, i.e., if a threshold larger than $T_{max}$ is used, then no document in the database can be retrieved. There are several justifications for using formula (9). First, it is a monotonically increasing function of T. As a result, a larger new average weight can be obtained for a larger threshold used. Second, when $T = 0$, $w_i' = w_i$, meaning that the original average weight is used when T = 0. Third, when $\sigma_i = 0$, $w_i'$ is reduced to $w_i$. Fourth, when $T = T_{max}$, $w_i' = w_i + c * \sigma_i$, meaning that the maximum term weight for term $t_i$ is $w_i + c * \sigma_i$. When normalized term weights for a term satisfies the *normal distribution*, then most term weights will fall in the interval $[w_i - 3 * \sigma_i, w_i + 3 * \sigma_i]$. Consequently, $c$ should be chosen to be close to 3. Furthermore, for a given query $q = (u_1, ..., u_r)$ and a threshold T, $T_{max}$ can be set to $(w_1 + 3 * \sigma_1) * u_1 + \cdots + (w_r + 3 * \sigma_r) * u_r$.

Intuitively, using a larger average weight for each term can be considered as using the average of larger term weights (i.e., first discard small weights and then average the remaining larger weights). When more small weights are discarded, the new average will be larger. When some small weights are not used to compute the new average weight $w_i'$, the probability that term $t_i$ appears in those documents whose weights are used to compute $w_i'$ needs to be computed. This probability, $p_i'$, should be used to replace $p_i$, just as $w_i'$ is to replace $w_i$, in the generating functions for usefulness estimation. $p_i'$ can be estimated by $p_i * p_{k_i}$, where $k_i$ is the value such that when the weights for term $t_i$ that are smaller than $k_i$ are not used, the average of the remaining positive weights will yield $w_i'$, and $p_{k_i}$ is the probability that a positive weight of term $t_i$ is greater than or equal to $k_i$. Similarly, for the dependent case, new co-variances should be estimated based on the understanding that term $t_i$ is considered to appear in a document only if the weight of $t_i$ in the document is greater than or equal to $k_i$. The new co-variances are used to replace the original co-variances in generating function (8). See [MLYW98] for details about these estimations.

### 3.4 Experimental Results

Three databases, D1, D2, and D3, and a collection of 6,597 queries are used in the experiment. D1, containing 761 documents, is the largest among the 53 databases that are collected at Stanford University for testing the gGlOSS system. The 53 databases are snapshots of 53 newsgroups at the Stanford CS Department news host and the queries are real queries submitted by users to the SIFT Netnews server [GrGM95a]. D2, containing 1,466 documents, is obtained by merging the two largest databases among the 53 databases. D3, containing 1,014 documents, is obtained by merging the 26 smallest databases among the 53 databases. As a result, the documents in D3 are more diverse than those in D2 and the documents in D2 are more diverse than those in D1.

For all documents and queries, non-content words such as "the", "of", etc. are removed. The similarity function is the normalized dot product function. The normalization guarantees that the similarity between any query and document will be between 0 and 1. As a result, no threshold larger than 1 is needed. When the dependent case is tested, 8,477, 12,482 and 13,658 co-variances are collected for D1, D2 and D3, respectively, for incorporating the dependencies among the terms. More co-variances are used for D2 and D3 because they contain more distinct terms (25,846 for D2 and 29,780 for D3 versus 16,065 for D1).

Consider database D1. For each query and each threshold, five usefulnesses are obtained. The first is the true usefulness obtained by comparing the query with each document in the database. The other four are estimated based on the database representatives and estimation formulas for the following cases: (1) The high-correlation case. (2) The disjoint case. (3) The non-binary and independent case with dynamic adjustment to $w_i$'s and $p_i$'s. (4) The non-binary and dependent case with dynamic adjustment to $w_i$'s, $p_i$'s and co-variances. All estimated usefulnesses, if they are not integers, are rounded to integers. The experimental results for D1 are summarized in Table 1.

Table 1: Comparison of Different Methods Using D1

| | | high-corr. | disjoint | independent | dependent |
|---|---|---|---|---|---|
| T | U | m/mis/diff | m/mis/diff | m/mis/diff | m/mis/diff |
| .0 | 4370 | 4370/0/7.1 | 4370/0/8.3 | 4370/0/1.6 | 4370/0/0.6 |
| .1 | 1655 | 460/72/20 | 146/0/21.7 | 862/14/11 | 1077/57/10.5 |
| .2 | 516 | 58/6/20.8 | 11/0/22.5 | 215/1/10.8 | 245/4/10.4 |
| .3 | 189 | 11/2/18.7 | 1/0/19.3 | 57/4/9.4 | 68/6/9.0 |
| .4 | 67 | 1/0/15.1 | 0/0/15.1 | 21/1/7.9 | 29/1/7.2 |
| .5 | 32 | 1/0/3.7 | 0/0/3.7 | 11/0/3.5 | 16/2/3.2 |
| .6 | 13 | 0/0/1.5 | 0/0/1.5 | 0/0/1.5 | 2/2/1.3 |

In Table 1, T is the threshold and U is the number of queries that identify D1 as useful (the true usefulness of the database with these queries is at least one). When T = 0.1, for example, 1,655 out of 6,597 queries identify D1 as useful. Now consider the column for the high-correlation case. "m/mis/diff" is a shorthand for “match/mismatch/average difference”. “460/72/20” means that out of the 1,655 queries that identify D1 as useful based on the true usefulness, 460 queries also identify D1 as useful based on the estimated usefulness by the high-correlation approach; there are 72 queries that identify D1 as useful based on the high-correlation approach but in reality, D1 is not useful to these 72 queries; and 20 is the average difference between the true usefulness and estimated usefulness over the 1,655 queries that identify D1 as useful based on the true usefulness. Clearly, a good estimation method should have its “match” close to “U” and its “mismatch” and “average difference” close to zero for any threshold. In other words, a better estimation should yield a larger “match” value and smaller “mismatch” and “average difference” values. In practice, correctly identifying a useful database is often more significant than incorrectly identifying a useless database as a useful database. This is because missing a useful database does more harm than searching a useless database. Therefore, if estimation method A has a larger “match” component than method B while A's “mismatch” component is not too much larger than B's “mismatch” component, then A should be considered better than B.

The experimental results for D2 and D3 are summarized in Tables 2 and 3, respectively.

Table 2: Comparison of Different Methods Using D2

| | | high-corr. | disjoint | independent | dependent |
|---|---|---|---|---|---|
| T | U | m/mis/diff | m/mis/diff | m/mis/diff | m/mis/diff |
| .0 | 4726 | 4726/0/18.2 | 4726/0/24.0 | 4726/0/8.0 | 4726/0/4.7 |
| .1 | 2753 | 995/160/32 | 141/33/36.5 | 1436/146/23 | 1960/211/21 |
| .2 | 1235 | 75/19/19.4 | 6/4/21.5 | 353/43/11.1 | 554/77/10.2 |
| .3 | 542 | 8/5/12.5 | 2/1/12.7 | 112/15/7.4 | 172/27/6.9 |
| .4 | 147 | 1/0/10.4 | 0/0/10.4 | 23/2/6.4 | 31/4/6.0 |
| .5 | 55 | 0/0/5.4 | 0/0/5.4 | 7/1/4.0 | 13/2/3.6 |
| .6 | 14 | 0/0/3.1 | 0/0/3.1 | 4/0/2.2 | 7/2/2.1 |

Table 3: Comparison of Different Methods Using D3

| | | high-corr. | disjoint | independent | dependent |
|---|---|---|---|---|---|
| T | U | m/mis/diff | m/mis/diff | m/mis/diff | m/mis/diff |
| .0 | 4886 | 4886/0/16.1 | 4886/0/16.8 | 4886/0/6.3 | 4886/0/3.7 |
| .1 | 2843 | 983/178/20 | 147/63/22 | 1448/180/16 | 2010/259/15 |
| .2 | 1245 | 85/29/12.7 | 14/15/13.3 | 260/39/8.1 | 458/70/7.6 |
| .3 | 431 | 7/6/10.3 | 4/2/10.5 | 62/11/7.2 | 100/20/6.8 |
| .4 | 140 | 0/1/7.1 | 0/1/7.1 | 16/4/5.7 | 25/11/5.3 |
| .5 | 48 | 0/0/3.8 | 0/0/3.8 | 7/1/3.0 | 11/3/2.8 |
| .6 | 15 | 0/0/2.2 | 0/0/2.2 | 4/0/2.2 | 6/0/2.1 |

The following can be observed from Tables 1, 2 and 3.

1. Our methods are much more accurate in estimating the database usefulness than the methods proposed in [GrGM95b] for all thresholds (see values under the “diff” category in the above tables). The error is typically reduced by a large percentage. Dramatic improvement for the match/mismatch category is also obtained at every threshold.

2. The dependent case is consistently better than the independent case. This indicates the usefulness of using co-variance information in estimating database usefulness. Note that for queries that actually use co-variances, the improvements are typically larger than those shown in the above tables. This is because the averages are computed for all queries that identify the database as useful based on the true usefulness, including those queries that do not use co-variances (i.e., single term queries or queries for which no co-variances are collected for their terms; nearly 30% of the 6,597 queries are single term queries).

3. While in all three databases, the two proposed estimation methods are more accurate than existing methods, the “mismatch” components are smaller for database 1, larger for database 2 and largest for database 3. This is likely due to the increased degrees of inhomogeneity of these three databases by their construction.

### 3.5 Discussion on Applicability

We now discuss several issues concerning the applicability of the two new methods in practice.

**Scalability**

If the representative of a database used by an estimation method has a large size relative to that of the database, then this estimation method will have a poor scalability as such a method is difficult to scale to thousands of text databases. Suppose each term occupies four bytes. Suppose each number (probability, average weight, standard deviation and co-variance) also occupies 4 bytes. Consider a database with $k$ different terms. For the independent case, $k$ probabilities, $k$ average weights and $k$ standard derivations are stored in the database representative, resulting in a total storage overhead of $16 * k$ bytes. For the dependent case, we also need to store some co-variances. We intend to use no more than k co-variances as obtaining and using the information is expensive. Thus, for the dependent case, the total storage overhead for the database representative is 20∗k bytes. The following table shows, for several document collections, the percentage of the sizes of the database representatives based on our approach for the independent case relative to the sizes of the original document collections.

| collection | size | # dist. terms | rep. size | % |
|---|---|---|---|---|
| WSJ | 40605 | 156298 | 1250 | 3.08 |
| FR | 33315 | 126258 | 1010 | 3.03 |
| DOE | 25152 | 186225 | 1490 | 5.92 |

In the above table, all sizes are in pages of 2 KB. The statistics of the second and third columns are for three document collections, namely, WSJ (Wall Street Journal), FR (Federal Register) and DOE (Department of Energy), which were collected by ARPA/NIST

[Harm93]. Clearly, the sizes of database representatives based on our approaches are only a very small fraction of those of original databases. Therefore, our approaches are fairly scalable. Also, typically, the percentage of space needed for a database representative relative to the database size will decrease as the database grows. This is because when new documents are added to a large database, the number of distinct terms either remain unchanged or grows slowly. In comparison, if a global inverted file index is built, the size of the index is usually comparable to that of the actual database. As a result, our proposed solution requires much less space.

Comparing to the database representative used in gGlOSS, the size of the database representative for the independent case is 33% larger (due to storing the standard deviation for each term) and the size of database representative for the dependent case is 67% larger. Clearly, there is a tradeoff between space and accuracy.

**Easiness of Obtaining Representative**

The representative of a database for the independent case can be obtained easily and efficiently. This is because the terms' probabilities appearing in a document, the average weights and their standard deviations can all be readily computed from the inverted file entries maintained by the local system. As a result, the local system can provide the information.

In contrast, obtaining the largest co-variances for the dependent case could be time-consuming due to typically a very large number of co-variances. However, the computation can be done off-line. Another possibility is to obtain co-variances adaptively. Specifically, initially, the database representative does not contain any co-variance information. Whenever a user query yields substantially more or fewer similar documents from a database than estimated, the co-variances of the terms in the query are computed and the significant ones are incorporated into the database representative. This method computes only significant co-variances. The global interface may also request those databases that barely missed the cutoff for being considered useful for the query to supply the dependency information. In any case, if a local database is incapable or unwilling to supply the co-variance information, the estimation will be performed based on the independence model as discussed in section 3.1.

**Query Processing Overhead**

It is known that a typical query submitted by a user in the Internet environment contains two to three terms only [ALSF97, Kow97]. The average number of terms in the queries used in our experiments and collected from Stanford University is also slightly less than 3. For such short queries, the computation cost of the estimation process for a query against a database representative is negligible.

# 4 Retrieval of Globally Most Similar Documents

In this section, we focus on the *collection fusion problem*. Its challenge stems from the fact that local systems are often autonomous and heterogeneous units. The problem arises in two forms: (1) The similarity function in a local database is different from that in the global interface. (2) The similarity functions in the local database and the global interface are identical but the weights of terms are different in the local database and the global interface. Both forms of the problem will be tackled in this section. Various approaches to solving this problem have been attempted (e.g., [CLBC95, GrGM97, VGJL95]). However, none of them can guarantee that all globally most similar documents will be retrieved. In a recent paper [GrGM97], an algorithm is provided to retrieve all globally most similar documents from a local database. However, this algorithm has a shortcoming. For a given global threshold of a query $q$, the local threshold of $q$ computed by this algorithm is often lower than necessary. As a result, a large number of documents that are not globally most similar may be retrieved. It is very desirable to get a tight local threshold to reduce communication cost, local processing cost and the cost of merging partial results. Our approach to the collection fusion problem is to form a proper local threshold for each local database for a given global threshold. For each local database, the constructed local threshold guarantees that all globally most similar documents will be retrieved while minimizing the number of non-similar documents retrieved. In this section, we first describe the construction of such a tight local threshold for a given global threshold of a query. Next, we discuss how to retrieve all globally most similar documents in local databases when both the local database and the global interface use the same popular *Cosine* similarity function.

## 4.1 Construction of Tight Local Threshold

Let $sim_{\mathcal{L}}(q, d)$ be a function that computes the local similarity between a query $q$ and a document $d$ in a local database $\mathcal{L}$ and $sim_{G}(q, d)$ be a function that computes the global similarity between $q$ and $d$. Let $T$ be a global threshold. A document $d$ is considered to be *desired* (or globally most similar) with respect to a query $q$ if $sim_{G}(q, d) \geq T$. (In earlier sections, $d$ is desired if $sim_{G}(q, d) > T$. This small change in the meaning of a desired document in this section is made only for the ease of presentation and does not affect the actual results.) Our objective is to determine a local threshold $L(T)$ so that all desired documents in $\mathcal{L}$ will be retrieved locally. Clearly, $L(T)$ is a non-decreasing function of the global threshold $T$.

Consider a Cartesian plane. Let the $x$-axis and $y$-axis of this plane represent the global and local similarities, respectively. Then, the global similarity $\alpha$ and the local similarity $\beta$ of a document can be represented by a point $(\alpha, \beta)$. Suppose the global and local similarities of all documents in local database $\mathcal{L}$ are distributed as depicted in Figure 1. Each point representing a document is marked with a '+' symbol. Then, all the documents in $\mathcal{L}$ with global similarities greater than or equal to $T$ are those represented in the figure by points lying on or to the right of the line $x = T$. Let $A(T)$ be the set of the $y$-coordinates of these points (or the local similarities of all these globally desired documents in $\mathcal{L}$). That is,

$$A(T) = \{sim_{\mathcal{L}}(q, d) | T \leq sim_G(q, d),\ d \in \mathcal{L}\} \quad (10)$$

In order not to miss any of the desired documents in $\mathcal{L}$, the local threshold $L(T)$ must not be larger than any similarity value in $A(T)$. At the same time, we want the local threshold to be as large as possible so that as few documents as possible in $\mathcal{L}$ with global similarities less than $T$ will be retrieved (those represented by points to the left of $x = T$). Hence, we choose the minimum local similarity value in $A(T)$ to be the local threshold for $\mathcal{L}$. That is,

$$L(T) = \min\{sim_{\mathcal{L}}(q, d) | T \leq sim_G(q, d), d \in \mathcal{L}\} \quad (11)$$

As can be seen in Figure 1, the local threshold $L(T)$ is the minimum of the $y$-coordinates of those points lying on or to the right of the line $x = T$.

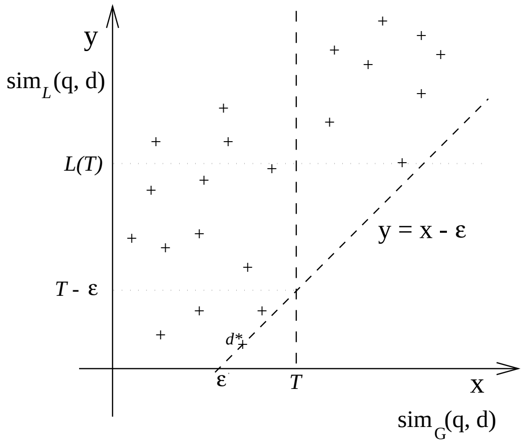


Figure 1: Local and Global Similarities of Documents in a Local Database $\mathcal{L}$

From (11) (and Figure 1), it is clear that all documents in $\mathcal{L}$ with global similarities greater than or equal to $T$ also have local similarities greater than or equal to $L(T)$. Thus, using $L(T)$, all desired documents in $\mathcal{L}$ will be retrieved.

**Proposition 3.** For a given global threshold $T$ of a query $q$, let local threshold $L(T)$ be defined as in (11) for a local database $\mathcal{L}$. Let $H(T)$ be any local threshold that will retrieve all documents with global similarities greater than or equal to $T$ from $\mathcal{L}$, i.e., for any documents $d$ in $\mathcal{L}$, whenever the global similarity $sim_G(q, d) \geq T$, the local similarity $sim_{\mathcal{L}}(q, d) \geq H(T)$. Then $H(T) \leq L(T)$. That is, the local threshold $L(T)$ is the tightest.

We now explain the difference between our approach and that used in [GrGM97] to obtaining a local threshold by means of Figure 1. In [GrGM97], a constant $\epsilon$ is determined such that the following inequality

$$sim_{\mathcal{L}}(q, d) \geq sim_G(q, d) - \epsilon \quad (12)$$

holds for every document $d$ in a local system $\mathcal{L}$. Referring to Figure 1, this is equivalent to finding an $\epsilon$ such that all the points are either on or above the line $y = x - \epsilon$. For a global threshold $T$, in [GrGM97], the local threshold is computed as $T - \epsilon$. In the situation shown in the figure, $d^*$ is a document represented by a point lying on the line $y = x - \epsilon$ and no points are under $y = x - \epsilon$. Thus, the value of $\epsilon$ shown is admissible. However, the $\epsilon$ cannot be made smaller, otherwise the point representing $d^*$ will fall below $y = x - \epsilon$. That is, the local threshold $T - \epsilon$, shown in Figure 1, is the best (highest) that can be determined using the method in [GrGM97]. As can be seen in the figure, the difference between the two thresholds $L(T)$ and $T - \epsilon$ can be quite large; and the number of undesirable documents retrieved from $\mathcal{L}$ is three using $L(T)$ versus ten using $T - \epsilon$.

For a given query $q$ and a given global threshold $T$, the optimal local threshold is the minimum of $A(T)$ (see (10)). In a local system, we do not know beforehand which documents have global similarities greater than or equal to $T$. As a result, it is not possible to determine $A(T)$. Hence, instead of finding the minimum of $A(T)$, we seek as the local threshold $L(T)$ the minimum possible local similarity that can be attained by a document $d$ with global similarity greater than or equal to $T$. In effect, our attempt to find the local threshold $L(T)$ becomes that of solving the following problem.

**(*)** *For a given query $q$, minimize, over all possible documents $d$ in $\mathcal{L}$, the function $sim_{\mathcal{L}}(q, d)$ subject to $sim_G(q, d) \geq T$.*

Various techniques can be employed to solve problem (*). In the following, we give two methods that can be used for a great variety of similarity functions.

**(a). Linear Programming Techniques**

Consider a common situation in which both the local and global similarity functions are the dot product function. Let $r$ be the number of terms in a given query $q$. Let the local query vector $l$ be $(l_1, \ldots, l_r)$ and the global query vector $g$ be $(g_1, \ldots, g_r)$, where $l_i$ and $g_i$ are the local weight and global weight of the $i$-th term in $q$, respectively. Let document $d$ be represented by the vector $(w_1, w_2, \ldots, w_r)$, where $w_i$ is the weight of the $i$-th term and $\alpha_i \leq w_i \leq \beta_i$ for some constant $\alpha_i$ and $\beta_i$. Then, $sim_{\mathcal{L}}(q,d) = l \bullet d = \sum_{i=1}^{r} l_i w_i$ and $sim_G(q,d) = g \bullet d = \sum_{i=1}^{r} g_i w_i$, where $\bullet$ denotes dot product. Our problem to find the local threshold $L(T)$ becomes the following minimization problem:

$$minimize \sum_{i=1}^{r} l_i w_i \ subject\ to \sum_{i=1}^{r} g_i w_i \geq T \ and\ \alpha_i \leq w_i \leq \beta_i,\ i = 1, \ldots, r.$$

This is a standard linear programming problem [Gass69]. Note that the set of inequalities $\alpha_i \leq w_i \leq \beta_i$, $i = 1, \ldots, r$, defines the space of all possible $d$ and is problem dependent.

In general, problem $(*)$ will be amenable to linear programming techniques if both the local and global similarity functions are some linear functions of the term weights of a document.

**Example 3** In the real-estate example of [GrGM97], the local similarity function weighs price (0.9) much more than location (0.1), while the global similarity function weighs them equally. For both location and price, it is assumed that a similarity between two houses can be computed. Specifically, the local similarity function is $sim_{\mathcal{L}}(q,d) = 0.1l + 0.9p$ and the global similarity function is $sim_G(q,d) = 0.5l + 0.5p$, where $l$ and $p$ are the similarities due to the location and price of a house, respectively. Given a global threshold $T$, to compute the local threshold $L(T)$ is equivalent to

*minimizing* $0.1l + 0.9p$ *subject to* $0.5l + 0.5p \geq T$ *and* $0 \leq l, p \leq 1$.

Using linear programming techniques, we obtain

$$L(T) = \begin{cases} 1.8T - 0.8 & \text{if } 0.5 \leq T \leq 1 \\ 0.2T & \text{if } T < 0.5 \end{cases}$$

In [GrGM97], the relationship between $sim_{\mathcal{L}}(q,d)$ and $sim_G(q,d)$ is determined to be $sim_{\mathcal{L}}(q,d) \geq sim_G(q,d) - 0.4$. The best local threshold that can be obtained based on this inequality is $T - 0.4$. For a global threshold $T = 0.8$, the local threshold is $0.8 - 0.4 = 0.4$, whereas $L(T)$, the local threshold according to our computation, is $1.8 \times 0.8 - 0.8 = 0.64$. It can be easily shown that $L(T) > T - 0.4$ except for $T = 0.5$ (when $T = 0.5$, $L(T) = T - 0.4$). ■

**(b). Lagrange's Multipliers**

The computation of $L(T)$ can be reformulated as the following two-step process.

1. Find the function $f(t)$, the minimum of the local similarity function $sim_{\mathcal{L}}(q,d)$, over all documents $d$ in $\mathcal{L}$, subject to $t = sim_G(q,d)$.
2. Minimize $f(t)$ in the range $t \geq T$.

Note that in step 1, $t$ is fixed and $d$ varies over all possible documents in $\mathcal{L}$, whereas in step 2, $t$ varies in the range $t \geq T$. It can be easily checked that the minimum of $f(t)$ obtained in step 2 is the desired threshold $L(T)$. Let $\{t_i\}$ be the set of terms specified in the query $q$. If both $sim_{\mathcal{L}}(q,d)$ and $sim_G(q,d)$ are differentiable functions with respect to the weight $w_i$ of each term $t_i$ of document $d$, then step 1 to find $f(t)$ can generally be achieved using the method of Lagrange in calculus [Widd89]. Once $f(t)$ is found, its minimum value in the range $t \geq T$ can be computed using calculus method or other algebraic techniques. If $f(t)$ is non-decreasing, $L(T)$ is simply $f(T)$. Since many similarity functions are differentiable, the above technique can be used to find the local threshold $L(T)$ for many different combinations of local and global similarity functions.

**Example 4** Let $d = (w_1, \ldots, w_r)$ be a document and $q = (u_1, \ldots, u_r)$ be a query. Let the global similarity function $sim_G(q,d) = \sum_{i=1}^{r} u_i w_i$ and a local similarity function $sim_{\mathcal{L}}(q,d) = (\sum_{i=1}^{r} u_i^p w_i^p)^{\frac{1}{p}}$ (known as p-norm in [SaMc83]) $(p \geq 1)$.

Step 1 to find $f(t)$ requires us to minimize $(\sum_{i=1}^{r} u_i^p w_i^p)^{\frac{1}{p}}$ subject to $\sum_{i=1}^{r} u_i w_i = t$.

Using the Lagrange method, $f(t)$ is found to be $t \cdot n^{(\frac{1}{p}-1)}$. As this function is an increasing function of $t$, for a global threshold $T$, the local threshold $L(T)$ is then $T \cdot n^{(\frac{1}{p}-1)}$. ■

## 4.2 Retrieval of Globally Most Similar Documents Using the *Cosine* function

In this subsection, we provide a technique to retrieve all globally most similar documents from a local database when both the local and global similarity functions are the widely used *Cosine* function [SaMc83]. Let $q = (v_1, \ldots, v_n)$ be a query, $v_j$ being the weight of the $j$-th query term. Let $d$ be a document having weight $w_j$ for the $j$-th query term. The similarity between $q$ and document $d$, computed using the *Cosine* function, is $(\sum_{j=1}^{n} v_j w_j)/(\mathbf{qd})$, where $\mathbf{q}$ and $\mathbf{d}$ are the *norms* of $q$ and $d$, respectively.

A common term weighting scheme is employed. In this scheme [BuSA93, VGJL95], for the $j$-th query term $t_j$, its weight in the query, $v_j$, is computed as $u_j \times I_j$, where $u_j$ is the weight of $t_j$ specified by the user (if the user does not specify the weight, then the weight is the number of times that term occurs in the query) and $I_j$ is the inverse document frequency weight (IDF) of the term. Recall that the IDF of a term $t$ in a database of $N$ documents is defined as $\log \frac{N}{n_t}$, where $n_t$ is the number of documents in the database containing the term $t$. The IDF of a term in a local database $\mathcal{L}$ depends on all the documents in $\mathcal{L}$ whereas the global IDF of the same term depends on all the documents in all databases. Thus, for a query term, its local and global IDFs, and hence its local and global query weights, are usually different. As for a term in a document, the weight of the term is determined using only document-dependent information; thus, a document has the same representation both locally and globally.

Let $q_u = (u_1, \ldots, u_n)$ be a query, where $u_j$ is the user-specified weight of the $j$-th query term. For the $j$-th query term, let $l_j$ and $l'_j$ be its IDF in a local database $\mathcal{L}$ and its global IDF, respectively. Thus, for query $q_u$, the local query vector for $\mathcal{L}$, $q_{\mathcal{L}}$, is $(u_1 l_1, \ldots, u_n l_n)$; and the global query vector $q_G$ is $(u_1 l'_1, \ldots, u_n l'_n)$. Let $\mathbf{q}_{\mathcal{L}}$ and $\mathbf{q}_G$ be the norms of $q_{\mathcal{L}}$ and $q_G$, respectively. Using the *Cosine* function, for query $q_u$, the local similarity for a document $d = (w_1, \ldots, w_n)$ in $\mathcal{L}$ is $sim_{\mathcal{L}}(q_{\mathcal{L}}, d) = (\sum_{j=1}^{n} u_j l_j w_j)/(\mathbf{q}_{\mathcal{L}}\mathbf{d})$; while the global similarity for $d$ is $sim_G(q_G, d) = (\sum_{j=1}^{n} u_j l'_j w_j)/(\mathbf{q}_G \mathbf{d})$.

Below, we present two methods to retrieve documents from a local database $\mathcal{L}$. The first computes the exact global similarity for each document in $\mathcal{L}$ through query modification. The second determines a local threshold to obtain all the globally desired documents in $\mathcal{L}$.

### (a). Local Document Retrieval Using Query Modification

In this approach, the global interface, upon receiving the user query $q_u = (u_1, \ldots, u_n)$, modifies the query as follows. The weight of the $j$-th query term is first multiplied by $(l'_j/l_j)$. Let $m_j = u_j \times (l'_j/l_j)$ be the product obtained for the $j$-th query term. Then, the modified query $q'_u = (m_1, \ldots, m_n)$ is submitted to $\mathcal{L}$ instead of $q_u$.

Upon receiving the modified user query $q'_u$, local database $\mathcal{L}$ computes the local query weight for the $j$-th term in $q'_u$ as the product $m_j \times l_j$, which equals to $u_j \times (l'_j/l_j) \times l_j$ or $u_j l'_j$. The resulting local query vector is $q'_{\mathcal{L}} = (u_1 l'_1, \ldots, u_n l'_r)$, which is the same as the global query vector $q_G$ for the original user query $q_u$. As mentioned above, in the term weighting scheme we are using, the local and global weight of a term in a document $d$ are identical. In effect, the local similarity computed between $q'_{\mathcal{L}}$ and a document $d$, $sim_{\mathcal{L}}(q'_{\mathcal{L}}, d)$, is the same as $sim_G(q_G, d)$, the global similarity between $q_G$ and $d$. Thus, all the globally desired documents in $\mathcal{L}$ can be determined and only these need to be retrieved.

### (b). Local Document Retrieval by Determining a Local Threshold

An alternative approach is to construct a local threshold for local database $\mathcal{L}$ to retrieve all the globally desired documents. As described in the previous subsection, for a given global threshold $T$, to find the local threshold $L(T)$, we

**(**)** *minimize the local similarity function* $\dfrac{\sum_{j=1}^{n} u_j l_j w_j}{\mathbf{q}_{\mathcal{L}}\mathbf{d}}$, *over all possible* $d$, *subject to* $\dfrac{\sum_{j=1}^{n} u_j l'_j w_j}{\mathbf{q}_G \mathbf{d}} \geq T$.

The above minimization problem $(**)$ can be solved using the method of Lagrange. In [MLYW98], we solve this problem with no restriction on the document term weights. (The usual situation is that all document term weights are non-negative.) The local threshold $L(T)$ obtained is $CT - \sqrt{(1-T^2)(1-C^2)}$, where $C = \dfrac{q_G \bullet q_{\mathcal{L}}}{\mathbf{q}_G \mathbf{q}_{\mathcal{L}}}$, and $q_G \bullet q_{\mathcal{L}} = \sum_{j=1}^{n} u_j^2 l_j l'_j$ is the dot product between $q_G$ and $q_{\mathcal{L}}$. This threshold is optimal if the values of $w_j$, $j = 1, \ldots, n$, at which the minimum is attained are non-negative.

## 5 Conclusions

In this paper, we proposed two new methods for estimating the number of potentially useful documents in a database. Our estimation methods are based upon established statistical theory and general database representation framework. Our experimental results indicate that these methods can yield substantial improvements over existing techniques. We also provided solutions to the collection fusion problem. Specifically, we reformulated the problem so that optimal local thresholds can be determined. Two techniques, the first involving linear programming and the second using Lagrange's method, are suggested to yield optimal local thresholds. By applying the techniques to three examples (the real-estate example, the p-norm, and the popular *Cosine* function), optimal solutions are obtained in each case. When both the global and local databases use the *Cosine* function, we also gave a query modification technique to compute the global similarity for a document in the local database.

**Acknowledgment:** We are grateful to Luis Gravano and Hector Garcia-Molina of Stanford University for providing us with the database and query collections used in [GrGM95a]. We also like to thank Yonghe Zhang and Xiaolan Liu for writing part of the code used for the experiment. This research is supported by the following organizations: NSF (IRI-9509253, CDA-9711582, HRD-9707076), Air Force (AFOSR 93-1-0059), NASA (NAGW-4080, NAG5-5095) and ARO (NAAH04-96-1-0049, DAAH04-96-1-0278).

## References

[ALSF97] G. Abdulla, B. Liu, R. Saad, and E. Fox. *Characterizing World Wide Web Queries.* TR-97-04, Virginia Polytechnic Institute, 1997.

[BuSA93] C. Buckley, G. Salton, and J. Allan. *Automatic Retrieval with Locality Information Using SMART.* First Text REtrieval Conference (TREC-1), pp. 59-72. NIST Special Publication 500-207, March 1993.

[CLBC95] J. Callan, Z. Lu, and W. Bruce Croft. *Searching Distributed Collections with Inference Networks.* ACM SIGIR, 1995.

[DuHa73] R. Duda, and P. Hart. *Pattern Classification and Scene Analysis, Chapter 4.* Wiley, New York, 1973.

[Gass69] S. I. Gass. *Linear Programming, Methods and Applications.* McGraw-Hill, New York, 1969.

[GrGM95a] L. Gravano, and H. Garcia-Molina. *Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies.* VLDB, 1995.

[GrGM95b] L. Gravano, and H. Garcia-Molina. *Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies.* Technical Report, Computer Science Dept., Stanford University, 1995. (This report discussed how to estimate the database usefulness defined in this paper for the high-correlation and disjoint scenarios. Such discussion did not appear in [GrGM95a].)

[GrGM97] L. Gravano, and H. Garcia-Molina. *Merging Ranks from Heterogeneous Internet Sources.* VLDB, 1997.

[Harm93] D. Harman. *Overview of the First Text Retrieval Conference.* Computer Systems Technology, U.S. Department of Commerce, NIST, 1993.

[HoDr97] A. Howe, and D. Dreilinger. *SavvySearch: A Meta-Search Engine that Learns Which Search Engines to Query.* AI Magazine, 18(2), 1997.

[KaMe91] B. Kahle, and A. Medlar. *An Information System for Corporate Users: Wide Area information Servers.* Technical Report TMC199, Thinking Machine Corporation, April 1991.

[Kost94] M. Koster. *ALIWEB: Archie-Like Indexing in the Web.* Computer Networks and ISDN Systems, 27:2, 1994, pp. 175-182 (http://www.cs.indiana.edu/aliweb/form.html).

[Kow97] G. Kowalski. *Information Retrieval Systems, Theory and Implementation.* Kluwer Academic Publishers, 1997.

[LaYu82] K. Lam, and C. Yu. *A Clustered Search Algorithm Incorporating Arbitrary Term Dependencies.* ACM TODS, September 1982.

[MaBi97] U. Manber, and P. Bigot. *The Search Broker.* USENIX Symposium on Internet Technologies and Systems (NSITS'97), Monterey, 1997.

[MLYW98] W. Meng, K. Liu, C. Yu, X. Wang, Y. Chang, and N. Rishe. *Determine Text Databases to Search in the Internet.* Technical Report, Dept. of CS, SUNY at Binghamton, 1998 (http://panda.cs.binghamton.edu/~meng/pub.d/vldb98s.ps).

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

[SaMc83] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McCraw-Hill, 1983.

[Salt89] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison Wesley, 1989.

[SeEt95] E. Selberg, and O. Etzioni. *Multi-Service Search and Comparison Using the MetaCrawler.* 4th Int'l World Wide Web Conference, Dec. 1995.

[SeEt97] E. Selberg, and O. Etzioni. *The MetaCrawler Architecture for Resource Aggregation on the Web.* IEEE Expert, 1997.

[TVGJ95] G. Towell, E. Voorhees, N. Gupta, and B. Johnson-Laird. *Learning Collection Fusion Strategies for Information Retrieval.* 12th Int'l Conf. on Machine Learning, 1995.

[VGJL95] E. Voorhees, N. Gupta, and B. Johnson-Laird. *Learning Collection Fusion Strategies.* ACM SIGIR Conference, 1995.

[Widd89] D. V. Widder. *Advanced Calculus.* 2nd Edition, Dover Publications, Inc., New York, 1989.

[YaGM95] T. W. Yan, and H. Garcia-Molina. *SIFT – A Tool for Wide-Area Information Dissemination.* USENIX 1995 Technical Conference, 1995.

[YuLS78] C. Yu, W. Luk and M. Siu. *On the Estimation of the Number of Desired Records with respect to a Given Query.* ACM TODS, March 1978.

[YuLe97] B. Yuwono, and D. Lee. *Server Ranking for Distributed Text Resource Systems on the Internet.* 5th Int'l Conf. On Database Systems For Advanced Applications (DASFAA'97), Melbourne, April 1997.