A Statistical Method for Estimating the Usefulness of Text Databases

King-Lup Liu[1], Clement Yu[1], Weiyi Meng[2], Wensheng Wu[1], Naphtali Rishe[3]

[1] Dept. of EECS, University of Illinois at Chicago, Chicago, IL 60607

{kliu, yu, wwu}@eecs.uic.edu

[2] Dept. of Computer Science, SUNY – Binghamton, Binghamton, NY 13902

meng@cs.binghamton.edu

[3] School of Computer Science, Florida International University, Miami, FL 33199

rishe@cs.fiu.edu

June 29, 1999

Abstract

Searching desired data in the Internet is one of the most common ways the Internet is used. No single search engine is capable of searching all data in the Internet. The approach that provides an interface for invoking multiple search engines for each user query has the potential to satisfy more users. When the number of search engines under the interface is large, invoking all search engines for each query is often not cost effective because it creates unnecessary network traffic by sending the query to a large number of useless search engines and searching these useless search engines wastes local resources. The problem can be overcome if the usefulness of every search engine with respect to each query can be predicted. In this paper, we present a statistical method to estimate the usefulness of a search engine for any given query. For a given query, the usefulness of a search engine in this paper is defined to be a combination of the number of documents in the search engine that are sufficiently similar to the query and the average similarity of these documents. Experimental results indicate that our estimation method is much more accurate than existing methods.

Keywords: Metasearch, Information Resource Discovery, Information Retrieval.

1 Introduction

The Internet has become a vast information source in recent years. To help ordinary users find desired data in the Internet, many *search engines* have been created. Each search engine has a corresponding *database* that defines the set of documents that can be searched by the search engine. Usually, an index for all documents in the database is created and stored in the search engine. For each *term* which represents a content word or a combination of several (usually adjacent) content words, this index can identify the documents that contain the term quickly. The pre-existence of this index is critical for the search engine to answer user queries efficiently.

Two types of search engines exist. General-purpose search engines attempt to provide searching capabilities for all documents in the Internet or on the Web. WebCrawler, HotBot, Lycos and Alta Vista are a few of such well-known search engines. Special-purpose search engines, on the other hand, focus on documents in confined domains such as documents in an organization or of a specific interest. Tens of thousands of special-purpose search engines are currently running in the Internet.

The amount of data in the Internet is huge (it is believed that by the end of 1997, there were more than 300 million web pages [15]) and is increasing at a very high rate. Many believe that employing a single general-purpose search engine for all data in the Internet is unrealistic. First, its processing power and storage capability may not scale to the fast increasing and virtually unlimited amount of data. Second, gathering all data in the Internet and keeping them reasonably up-to-date are extremely difficult if not impossible. Programs (i.e., *Robots*) used by search engines to gather data automatically may slow down local servers and are increasingly unpopular.

A more practical approach to providing search services to the entire Internet is the following multi-level approach. At the bottom level are the local search engines. These search engines can be grouped, say based on the relatedness of their databases, to form next level search engines (called *metasearch engines*). Lower level metasearch engines can themselves be grouped to form higher level metasearch engines. This process can be repeated until there is only one metasearch engine at the top. A metasearch engine is essentially an interface and it does not maintain its own index on documents. However, a sophisticated metasearch engine may maintain information about the contents of the (meta)search engines at a lower level to provide better service. When a metasearch engine receives a user query, it first passes the query to the appropriate (meta)search engines at the next level recursively until real search engines are encountered, and then collects

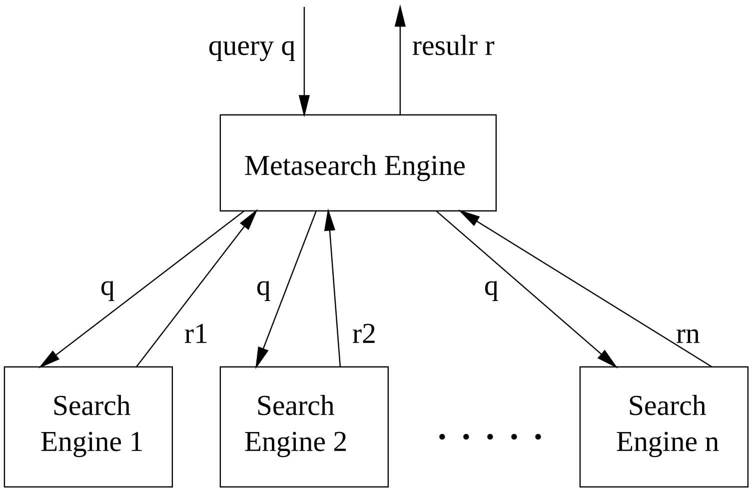


Figure 1: A Two-Level Search Engine Organization

(sometimes, reorganizes) the results from real search engines, possibly going through metasearch engines at lower levels. A two-level search engine organization is illustrated in Figure 1. The advantages of this approach are (a) user queries can (eventually) be evaluated against smaller databases in parallel, resulting in reduced response time; (b) updates to indexes can be localized, i.e., the index of a local search engine is updated only when documents in its database are modified; (Although local updates may need to be propagated to upper level metadata that represent the contents of local databases, the propagation can be done infrequently as the metadata are typically statistical in nature and can tolerate certain degree of inaccuracy.) (c) local information can be gathered more easily and in a more timely manner; and (d) the demand on storage space and processing power at each local search engine is more manageable. In other words, many problems associated with employing a single super search engine can be overcome or greatly alleviated when this multi-level approach is used.

When the number of search engines invokable by a metasearch engine is large, a serious inefficiency may arise. Typically, for a given query, only a small fraction of all search engines may contain useful documents to the query. As a result, if every search engine is blindly invoked for each user query, then substantial unnecessary network traffic will be created when the query is sent to useless search engines. In addition, local resources will be wasted when useless databases are searched. A better approach is to first identify those search engines that are most likely to provide useful results to a given query and then pass the query to only these search engines for desired documents. Examples of systems that employ this approach include WAIS [12], ALIWEB [13], gGlOSS [6], SavvySearch [9] and D-WISE [27]. A challenging problem with this approach is how to identify potentially useful search engines. The current solution to this problem is to rank all underlying databases in decreasing order of usefulness 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. For a given query, the current approach can tell the user, to some degree of accuracy, which search engine is likely to be the most useful, the second most useful, etc. While such a ranking can be helpful, it cannot tell the user *how useful* any particular search engine is.

In this paper, the usefulness of a search engine to a given query is measured by a pair of numbers (NoDoc, AvgSim), where *NoDoc* is the number of documents in the database of the search engine 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 and *AvgSim* is the average similarity of these potentially useful documents. Note that the global similarity function may or may not be the same as the local similarity function employed by a local search engine. While the threshold provides the minimum similarity for a document to be considered potentially useful, *AvgSim* describes more precisely the expected quality of each potentially useful document in a database. The two numbers together characterize the usefulness of each search engine very nicely. *NoDoc* and *AvgSim* can be defined precisely as follows:

$$NoDoc(T, q, D) = cardinality(\{d | d \in D \ and \ sim(q, d) > T\}) \tag{1}$$

$$AvgSim(T, q, D) = \frac{\sum_{d \in D \wedge sim(q,d) > T} sim(q, d)}{NoDoc(T, q, D)} \tag{2}$$

where T is a threshold, D is the database of a search engine and $sim(q, d)$ is the similarity (closeness) between a query q and a document d in D.

A query is simply a set of words submitted by a user. It is transformed into a vector of *terms* with *weights* [22], where a term is essentially a content word and the dimension of the vector is the number of all distinct terms. When a term appears in a query, the component of the query vector corresponding to the term, which is the *term weight*, is positive; if it is absent, the corresponding term weight is zero. The weight of a term usually depends on the number of occurrences of the term in the query (relative to the total number of occurrences of all terms in the query) [22, 26]. It may also depend on the number of documents having the term relative to the total number of documents in the database. A document is similarly transformed into a vector with weights. The similarity between a query and a document can be measured by the dot product of their respective vectors. Often, the dot product is divided by the product of the *norms* of the two vectors, where the norm of a vector $(x_1, x_2, ..., x_n)$ is $\sqrt{\sum_{i=1}^{n} x_i^2}$. This is to normalize similarities into values between 0 and 1. The similarity function with such a normalization is known as the *Cosine* function [22, 26]. Other similarity functions, see for example [21], are also possible.

In practice, users may not know how to relate a threshold to the number of documents they like to retrieve. Therefore, users are more likely to tell a metasearch engine directly the number of most similar documents (to their query) they like to retrieve. Such a number can be translated into a threshold by the metasearch engine. For example, suppose we have three databases D1, D2 and D3 such that for a user query q, when T = 0.4, NoDoc(T,q,D1) = 8, NoDoc(T,q,D2) = 3, and NoDoc(T,q,D3) = 4; and when T = 0.5, NoDoc(T,q,D1) = 3, NoDoc(T,q,D2) = 0, and NoDoc(T,q,D3) = 2. In this case, if a user wants 5 documents, then T = 0.5 should be used. As a result, 3 documents will be retrieved from D1 and 2 documents from D3. In general, the appropriate threshold can be determined by estimating the *NoDoc* of each search engine in decreasing thresholds.

Note that knowing *how useful* a search engine is can be very important for a user to determine which search engines to use and how many documents to retrieve from each selected search engine. For example, if a user knows that a highly-ranked search engine with a large database has very few useful documents and searching such a large database is costly, then the user may choose not to use the search engine. Even if the user decides to use the search engine, the cost of the search can still be reduced by limiting the number of documents to be returned to the number of useful documents in the search engine. Such an informed decision is not possible if only ranking information is provided.

This paper has several contributions. First, a new measure is proposed to characterize the usefulness of (the database of) a search engine with respect to a query. The new measure is easy to understand and very informative. As a result, it is likely to be more useful in practice. Second, a new statistical method, a subrange based estimation method, is proposed to identify search engines to use for a given query and to estimate the usefulness of a search engine for the query. We will show that both *NoDoc* and *AvgSim* can be obtained from the same process. Therefore, little additional effort is required to compute both of them in comparison to obtaining any one of them only. The method yields very accurate estimates and is substantially better than existing methods as demonstrated by experimental results. It also guarantees the following property. Let the largest similarity of a document with a query among all documents in search engine i be large_sim$_i$. Suppose large_sim$_i$ > large_sim$_j$ for two search engines i and j, and a threshold of retrieval T is set such that large_sim$_i$ > T > large_sim$_j$. Then based on our method, search engine i will be invoked while search engine j will not if the query is a single term query. This is consistent to the ideal situation where documents are examined in descending order of similarity. Since a large portion of Internet queries are single term queries [10, 11], the above property of our approach means that a large percentage of all Internet queries will be sent to the correct search engines to be processed using our method. In addition, the new method is quite robust as it can still yield good result even when approximate statistical data are used by the method. This method is further improved when adjacent terms in a query are combined. Close to optimal performance is obtained.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents our basic method for estimating the usefulness of search engines. Section 4 discusses several issues on how the proposed method can be applied in practice. Experimental results will be presented in Section 5. Section 6 describes how adjacent query terms can be combined to yield higher performance. Section 7 concludes the paper.

2 Related Work

To be able to identify useful search engines to a query, some characteristic information about the database of each search engine must be stored in the metasearch engine. We call such information the *representative* of a search engine. Different methods for identifying useful search engines can be developed based on the representatives used.

Several metasearch engines have employed various methods to identify potentially useful search engines [6, 9, 12, 13, 17, 27]. However, the database representatives used in most metasearch engines cannot be used to estimate the number of globally most similar documents in each search engine [3, 12, 13, 27]. In addition, the measures that are used by these metasearch engines to rank the search engines are difficult to understand. As a result, separate methods have to be used to convert these measures to the number of documents to retrieve from each search engine. Another shortcoming of these measures is that they are independent of the similarity threshold (or the number of documents desired by the user). As a result, a search engine will always be ranked the same regardless of how many documents are desired, if the databases of these search engines are fixed. This is in conflict with the following situation. For a given query, a search engine may contain many moderately similar documents but very few or zero highly similar documents. In this case, a good measure should rank the search engine high if a large number of moderately similar documents are desired and rank the search engine low if only highly similar documents are desired.

A probabilistic model for distributed information retrieval is proposed in [2]. The method is more suitable in a feedback environment, i.e., documents previously retrieved have been identified to be either relevant or irrelevant.

In gGlOSS [6], a database of m distinct terms is represented by m pairs (f_i, W_i), where f_i is the number of documents in the database that contain the ith term and W_i is the sum of the weights of the ith term over all documents in the database, $i = 1, ..., m$. The usefulness of a search engine with respect to a given query in gGlOSS is defined to be the sum of all document similarities with the query that are greater than a threshold. This usefulness measure is less informative than our measure. For example, from a given sum of similarities of documents in a database, we cannot tell how many documents are involved. On the other hand, our measure can derive the measure used in gGlOSS. The representative of gGlOSS can be used to estimate the number of useful documents in a database [7] and consequently, it can be used to estimate our measure. However, the estimation methods used in gGlOSS are very different from ours. The estimation methods employed in [6, 7] 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 and the set of documents containing term k are disjoint). Due to the restrictiveness of the above assumptions, the estimates produced by these two methods are not accurate. Note that when the measure of similarity sum is used, the estimates produced by the two methods in gGlOSS form lower and upper bounds to the true similarity sum. As a result, the two methods are more useful when used together than when used separately. Unfortunately, when the measure is the number of useful documents, the estimates produced by the two methods in gGlOSS no longer form bounds to the true number of useful documents.

[25] 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 the ith term 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 [16], 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. The estimation method in [18] permits term weights to be non-binary. However, it utilizes the non-binary information in a way that is very different from our subrange-based statistical method to be described in Section 3.2 of this paper.

3 A New Method for Usefulness Estimation

We present our basic method for estimating the usefulness of a search engine in Section 3.1. The basic method allows the values of term weights to be any non-negative real numbers. Two assumptions are used by the basic method: (1) the distributions of the occurrences of the terms in the documents are independent. In other words, the occurrences of term i in the documents have no effect on the occurrences or non-occurrences of another term, say term j, in the documents; (2) for a given database of a search engine, all documents having a term have the same weight for the term. Under the two assumptions, the basic method can accurately estimate the usefulness of a search engine. In Section 3.2, we apply a subrange-based statistical method to remove the second assumption. The first assumption can also be removed by incorporating term dependencies (co-variances) into the basic solution [18]. The problem of incorporating term dependencies will be addressed in Section 6. We will see in Section 5 that very accurate usefulness estimates can be obtained even with the term independence assumption.

3.1 The Basic Method

Consider a database D of a search engine with m distinct terms. Each document d in this database can be represented as a vector $d = (d_1, \ldots, d_m)$, where d_i is the weight (or significance) of the ith term t_i in representing the document, $1 \le i \le m$. Each query can be similarly represented. Consider query $q = (u_1, u_2, ..., u_m)$, where u_i is the weight of t_i in the query, $1 \le i \le m$. If a term does not appear in the query, then its corresponding weight will be zero in the query vector. The similarity between q and document d can be defined as the dot product of their respective vectors, namely $sim(q, d) = u_1 * d_1 + \cdots + u_m * d_m$. Similarities are often normalized between 0 and 1. One common normalized similarity function is the $Cosine$ function [22] although other forms of normalization are possible, see for example [21].

With the basic method, 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 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:

$$(p_1 * X^{w_1 * u_1} + (1 - p_1)) * (p_2 * X^{w_2 * u_2} + (1 - p_2)) * \cdots * (p_r * X^{w_r * u_r} + (1 - p_r)) \tag{3}$$

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. 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 (3) is the probability that a document in D has similarity s with q.

Proof: Clearly, s must be the sum of zero or more $w_i * u_i$'s with each $w_i * u_i$ being used at most once. Different combinations of $w_i * u_i$'s may add up to s. Without loss of generality, let us assume that there are two such combinations. Suppose $s = w_{i_1} * u_{i_1} + \cdots + w_{i_k} * u_{i_k} = w_{j_1} * u_{j_1} + \cdots + w_{j_l} * u_{j_l}$. Then the probability that q has similarity s with a document d in D is the probability that d has either exactly the terms in $\{t_{i_1}, ..., t_{i_k}\}$ or exactly the terms in $\{t_{j_1}, ..., t_{j_l}\}$. With the independence assumption about terms, the probability that d has exactly terms in $\{t_{i_1}, ..., t_{i_k}\}$ is P $= \prod p_v * \prod(1 - p_y)$, where the first product is over all v in $\{i_1, ..., i_k\}$ and the second product is over all y in {1, 2, ..., r} $-$ $\{i_1, ..., i_k\}$. Similarly, the probability that d has exactly terms in $\{t_{j_1}, ..., t_{j_l}\}$ is Q $= \prod p_v * \prod(1 - p_y)$, where the first product is over all v in $\{j_1, ..., j_l\}$ and the second product is over all y in {1, 2, ..., r} $-$ $\{j_1, ..., j_l\}$. Therefore, the probability that d has either exactly the terms in $\{t_{i_1}, ..., t_{i_k}\}$ or exactly the terms in $\{t_{j_1}, ..., t_{j_l}\}$ is the sum of P and Q which is the same as the coefficient of X^s in function (3). ■

Example 1 Let q be a query with three terms with all weights equal to 1, i.e., q = (1, 1, 1) (for ease of understanding, the weights of terms in the query and documents are not normalized). Suppose database D has five documents and their vector representations are (only components corresponding to query terms are given): (3, 0, 0), (1, 1, 0), (0, 0, 2), (2, 0, 2) and (0, 0, 0). Namely, the first document has query term 1 and the corresponding weight is 3. Other document vectors can be interpreted similarly. From the five documents in D, we have (p_1, w_1) = (0.6, 2) as 3 out of 5 documents have term 1 and the average weight of term 1 in such documents is 2. Similarly, (p_2, w_2) = (0.2, 1) and (p_3, w_3) = (0.4, 2). Therefore, the corresponding generating function is:

$$(0.6 * X^2 + 0.4)(0.2 * X + 0.8)(0.4 * X^2 + 0.6) \tag{4}$$

Consider the coefficient of X^2 in the function. Clearly, it is the sum of $p_1 * (1 - p_2) * (1 - p_3)$ and $(1 - p_1) * (1 - p_2) * p_3$. The former is the probability that a document in D has exactly the first query term and the corresponding similarity with q is w_1 (=2). The latter is the probability that a document in D has exactly the last query term and the corresponding similarity is w_3 (=2). Therefore, the coefficient of X^2, namely, $p_1 * (1 - p_2) * (1 - p_3) + (1 - p_1) * (1 - p_2) * p_3 = 0.416$, is the estimated probability that a document in D has similarity 2 with q. ■

After generating function (3) has been expanded and the terms with the same X^s have been combined, we obtain

$$a_1 * X^{b_1} + a_2 * X^{b_2} + \cdots + a_c * X^{b_c} \tag{5}$$

We assume that the terms in (5) are listed in descending order of the exponents, i.e., $b_1 > b_2 > ... > b_c$. By Proposition 1, a_i is the probability that a document in D has similarity b_i with q. In other words, if database D contains n documents, then $n * a_i$ is the expected number of documents that have similarity b_i with query q. For a given similarity threshold T, let C be the largest integer to satisfy $b_C > T$. Then, the *NoDoc* measure of D for query q based on threshold T, namely, the number of documents whose similarities with query q are greater than T, can be estimated as:

$$est\_NoDoc(T, q, D) = \sum_{i=1}^{C} n * a_i = n \sum_{i=1}^{C} a_i \tag{6}$$

Note that $n * a_i * b_i$ is the expected sum of all similarities of those documents whose similarities with the query are b_i. Thus, $\sum_{i=1}^{C}(n * a_i * b_i)$ is the expected sum of all similarities of those documents whose similarities with the query are greater than T. Therefore, the *AvgSim* measure of D for query q based on threshold T, namely, the average similarity of those documents in database D whose similarities with q are greater than T, can be estimated as:

$$est\_AvgSim(T, q, D) = \frac{n \sum_{i=1}^{C} a_i * b_i}{n \sum_{i=1}^{C} a_i} = \frac{\sum_{i=1}^{C} a_i * b_i}{\sum_{i=1}^{C} a_i} \tag{7}$$

Since both *NoDoc* and *AvgSim* can be estimated from the same expanded expression (5), estimating both of them requires little additional effort in comparison to estimating only one of them.

Example 2 (Continue Example 1). When the generating function (4) is expanded, we have:

$$0.048 * X^5 + 0.192 * X^4 + 0.104 * X^3 + 0.416 * X^2 + 0.048 * X + 0.192 \tag{8}$$

From formula (6), we have $est\_NoDoc(3, q, D) = 5 * (0.048 + 0.192) = 1.2$ and $est\_AvgSim(3, q, D) = (0.048*5+0.192*4)/(0.048+0.192) = 4.2$. It is interesting to note that the actual *NoDoc* is $NoDoc(3, q, D) = 1$ since only the fourth document in D has a similarity (the similarity is 4) higher than 3 with q and the actual *AvgSim* is $AvgSim(3, q, D) = 4$. The second and the third columns of Table 1 list the true usefulness of D with respect to q and different T's. Note that in Table 1, the *NoDoc* and *AvgSim* values are obtained when the estimated similarities are strictly greater than the threshold T. When $NoDoc = 0$, no value for *AvgSim* is available. In this case, the corresponding entry under *AvgSim* will be left blank. The remaining columns list the estimated usefulness based on different methods. The fourth and the fifth columns are for our basic method. The sixth and the seventh columns are for the estimation method based on the high-correlation case, and the eighth and ninth columns are for the estimation method for the disjoint case, which are proposed in [6, 7]. It can be observed that the estimates produced by the new method approximate the true values better than those given by the methods based on the high-correlation and the disjoint assumptions. ■

	True		Basic Method		High-Correlation		Disjoint	
T	NoDoc	AvgSim	NoDoc	AvgSim	NoDoc	AvgSim	NoDoc	AvgSim
0	4	2.75	4.04	2.84	3	3.67	6	1.83
1	4	2.75	3.80	2.96	3	3.67	5	2.00
2	2	3.50	1.72	3.87	2	4.50	0	
3	1	4.00	1.20	4.20	2	4.50	0	
4	0		0.24	5.00	1	5.00	0	

Table 1: Estimated Usefulness versus True Usefulness

Furthermore, the distribution of the exact similarities between q and documents in D can be expressed by the following function (analogous to expression (8)):

$$0 * X^5 + 0.2 * X^4 + 0.2 * X^3 + 0.4 * X^2 + 0 * X + 0.2 \tag{9}$$

That is, the probability that a document having similarity 5 with q is zero and the probability that a document having similarity 4 with q is 0.2 since no document in D has similarity 5 with q and exactly one document in D has similarity 4 with q. Other terms can be interpreted similarly. Notice the good match between the corresponding coefficients in (8) and (9).

3.2 Subrange-based Estimation for Non-uniform Term Weights

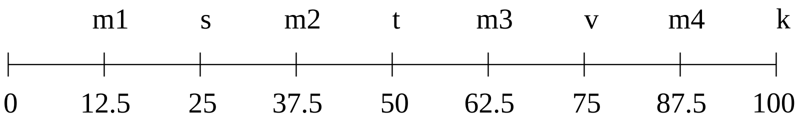
One assumption used in the above basic solution is that all documents having a term have the same weight for the term. This is not realistic. In this subsection, we present a subrange-based statistical method to overcome the problem.

Consider a term t. Let w and σ be the average and the standard deviation of the weights of t in the set of documents containing t, respectively. Let p be the probability that term t appears in a document in the database. Based on the basic solution in Section 3.1, if term t is specified in a query, then the following polynomial is included in the probability generating function (see Expression (3)):

$$p * X^{u*w} + (1 - p) \tag{10}$$

where u is the weight of the term in the user query. This expression essentially assumes that the term t has an uniform weight of w for all documents containing the term. In reality, the term weights may have a non-uniform distribution among the documents having the term. Let these weights in *non-ascending* order of magnitude be $w_1, w_2, ..., w_k$, where $k = p * n$ is the number of documents having the term and n is the total number of documents in the database. Suppose we partition the weight range of t into 4 subranges, each containing 25% of the term weights, as follows. The first subrange contains the weights from w_1 to w_s, where $s = 25\% * k$; the second subrange contains the weights from w_{s+1} to w_t, where $t = 50\% * k$; the third subrange contains the weights from w_{t+1} to w_v, where $v = 75\% * k$ and the last subrange contains weights from w_{v+1} to w_k. In the first subrange, the median is the $(25\% * k/2)$-th weight of the term weights in the subrange and is w_{m1}, where $m1 = 12.5\% * k$; similarly, the median weights in the second, the third and the fourth subranges have median weights w_{m2}, w_{m3} and w_{m4}, respectively, where $m2 = 37.5\% * k, m3 = 62.5\% * k$ and

$m4 = 87.5\% * k$. This can be illustrated by the following figure.



Then, the distribution of the term weights of t may be approximated by the following distribution: The term has a uniform weight of w_{m1} for the first 25% of the k documents having the term, another uniform weight of w_{m2} for the next 25% of the k documents, another uniform weight of w_{m3} for the next 25% of documents and another uniform weight of w_{m4} for the last 25% of documents.

With the above weight approximation, for a query containing term t, polynomial (10) in the generating function can be replaced by the following polynomial:

$$p_1 * X^{u*w_{m1}} + p_2 * X^{u*w_{m2}} + p_3 * X^{u*w_{m3}} + p_4 * X^{u*w_{m4}} + (1 - p) \tag{11}$$

where p_j is the probability that term t occurs in a document and has a weight of $w_{mj}, j = 1, 2, 3, 4$. Since 25% of those documents having term t are assumed to have a weight of w_{mj} for t and for each j, $p_j = p/4$. Essentially, polynomial (11) is obtained from polynomial (10) by decomposing the probability p that a document has the term into 4 probabilities, p_1, p_2, p_3 and p_4, corresponding to the 4 subranges. A weight of term t in the first subrange, for instance, is assumed to be w_{m1} and the corresponding exponent of X in polynomial (11) is the similarity due to this term t, which equals $u * w_{m1}$, taking into consideration the query term weight u.

Since it is expensive to find and to store w_{m1}, w_{m2}, w_{m3} and w_{m4}, they are approximated by assuming that the weights of the term are normally distributed with mean w and standard deviation σ. Then $w_{mi} = w + c_i * \sigma$, where c_i is a constant that can be looked up from a table for the standard normal distribution. It should be noted that these constants are independent of individual terms and therefore one set of such constants is sufficient for all terms.

Example 3 Suppose the average weight of a term t is w = 2.8 (to ease presentation, assume that term weights are not normalized) and the standard deviation of the weights of the term is 1.3. From a table of the standard normal distribution, $c_1 = 1.15, c_2 = 0.318, c_3 = -0.318$ and $c_4 = -1.15$. Note that these constants are independent of the term. Thus, $w_{m1} = w + c_1 * 1.3 = 4.295; w_{m2} = w + c_2 * 1.3 = 3.2134; w_{m3} = w + c_3 * 1.3 = 2.3866$ and $w_{m4} = w + c_4 * 1.3 = 1.305$.

Suppose the probability that a document in the database has the term t is 0.32. Then $p_i = 0.08$, for i = 1, 2, 3 and 4. Suppose the weight of the term t in the query is 2. Then, the polynomial for the term t in the generating function is

$$0.08 * X^{8.59} + 0.08 * X^{6.4268} + 0.08 * X^{4.7732} + 0.08 * X^{2.61} + 0.68$$

■

In general, it is not necessary to divide the weights of the term into 4 equal subranges. For example, we can divide the weights into 5 subranges of different sizes, yielding a polynomial of the form:

$$p_1 * X^{u*w_{m1}} + p_2 * X^{u*w_{m2}} + p_3 * X^{u*w_{m3}} + p_4 * X^{u*w_{m4}} + p_5 * X^{u*w_{m5}} + (1-p)$$

where $p_1 + p_2 + p_3 + p_4 + p_5 = p$, p_i represents the probability that the term has weight in the ith subrange, and w_{mi} is the median weight of the term in the ith subrange.

In the experiments we report in Section 5, a specific six-subrange is used with a special subrange (the highest subrange) containing the maximum normalized weight only (see Section 5). The normalized weight of a term t in a document is the weight of the term in the document divided by the norm of the document. The maximum normalized weight of t in the database is the largest normalized weight among all documents containing t in the database. The probability for the highest subrange is set to be 1 divided by the number of documents in the database. This probability may be an underestimate. However, since different documents usually have different norms and therefore there is usually only one document having the largest normalized weight, the estimated probability is reasonable.

Example 4 (Continuing Example 3: Term weights are not normalized to facilitate ease of reading.) Suppose that the number of documents in the database is $n = 100$ and the maximum weight of the term t is $mw = 5.8$. Since the probability that the term occurs in the documents is 0.32, the number of documents having the term $t = 0.32 * 100 = 32$.

We use 5 subranges, which are obtained by splitting the first subrange in Example 3 into two subranges, and the first new subrange is covered by the maximum weight. Since we assume that there is only one document having the largest term weight, the probability that a document has the largest term weight is $p_1 = 1/100 = 0.01$. Among the documents having the term, the percentage of documents having the largest term weight is $1/32 * 100\% = 3.125\%$. This document occupies the first new subrange. The second new subrange contains term weights from the 75 percentile to the $100 - 3.125 = 96.875$ percentile. The probability associated with the second new subrange is $(25/100 - 1/32) * 0.32 = 0.21875 * 0.32 = 0.07$. The median for the second new subrange is $(75 + 96.875)/2 = 85.9375$ percentile. By looking up a standard normal distribution table, $c_2 = 1.08$. Therefore, $w_{m2} = 2.8 + c_2 * 1.3 = 4.204$.

The next 3 subranges are identical to the last 3 subranges given in Example 3, that is, $c_3 = 0.318$, $c_4 = -0.318$, $c_5 = -1.15$, $w_{m3} = 3.2134$, $w_{m4} = 2.3866$, $w_{m5} = 1.305$, and $p_3 = p_4 = p_5 = 25\% * 0.32 = 0.08$. Assume that the query term weight is 2. The polynomial for the term t in the generating function is

$$0.01 * X^{11.6} + 0.07 * X^{8.408} + 0.08 * X^{6.4268} + 0.08 * X^{4.7732} + 0.08 * X^{2.61} + 0.68$$

■

Note that the subrange-based method needs to know the standard deviation of the weights for each term. As a result, a database with m terms is now represented as m triplets $\{(p_i, w_i, \sigma_i)\}$, $i = 1, \ldots, m$, where p_i is the probability that term t_i appears in a document in the database, w_i is the average weight of term t_i in all documents containing the term and σ_i is the standard deviation of the weights of t_i in all documents containing t_i. Furthermore, if the maximum normalized weight of each term is used by the highest subrange, then the database representative will contain m quadruplets $\{(p_i, w_i, \sigma_i, mw_i)\}$, with mw_i being the maximum normalized weight for term t_i. Our experimental results indicate that the maximum normalized weight is a critical parameter that can drastically improve the estimation accuracy of search engine usefulness. In the following subsection, we elaborate why the maximum normalized weight is a critically important piece of information for correctly identifying useful search engines.

3.2.1 Single-term Query

Consider a query q that contains a single term t. Suppose the similarity function is the widely used *Cosine* function. Then the normalized query has a weight of 1 for the term t and the similarity of a document d with the query q using the *Cosine* function is w', which is the dot product $(1 \cdot w')$, where $w' = w/|d|$ is the normalized weight of the term t in the document d, $|d|$ is the *norm* of d, and w is the weight of the term in the document before normalization. Consider a database D_1 that contains documents having term t. The component of the database representative concerning term t will contain the maximum normalized weight mw_1, if mw_1 is the largest normalized weight of term t among all documents in database D_1. By our discussion just before this subsection, the highest subrange contains the maximum normalized weight only and its probability is set to be 1 divided by the number of documents in the database. The generating function for the query q for the database D_1 is:

$$p_1 * X^{mw_1} + ...$$

where p_1 is $1/n$ and n is the number of documents in this database. For a different database D_i, $i \neq 1$, having maximum normalized term weight mw_i for term t, the generating function for the same query for database D_i is obtained by replacing mw_1 by mw_i in the above expression (with p_1 being modified accordingly). Suppose mw_1 is the largest maximum normalized term weight of term t among all databases and mw_2 is the second largest with $mw_1 > mw_2$. Suppose the threshold of retrieval T is set such that $mw_1 > T > mw_2$. Then, the estimated number of documents with similarities greater than T in database D_1 is at least $p_1 * n = 1$ because $mw_1 > T$. Since $T > mw_2 > mw_j, (j \neq 1, 2)$, the estimated numbers of documents with similarities greater than T in database D_2 and other databases are zero. Thus, database D_1 is the only database which can be identified by our estimation method as having documents with similarities greater than T for the single term query. This identification is correct, because documents with normalized term weight mw_1 only appear in database D_1 and documents in other databases have similarities less than or equal to mw_2. In general, if the maximum normalized weights of term t in the databases are arranged in descending order $mw_1 > mw_2 > ... > mw_s > ... > mw_v$, where v is the number of databases, and the threshold T is set such that $mw_{s-1} > T > mw_s$, then databases $D_1, D_2, ..., D_{s-1}$ will be identified by our estimation method to be searched. This identification is consistent with the ideal situation, where these selected databases contain documents with similarities greater than T and other databases do not have the desired documents (with similarities greater than T). Thus, our method guarantees the correct identification of useful search engines for single term queries. The same argument applies to other similarity functions such as [21]. Several recent studies indicate that 30% or higher percentages of all Internet queries are single-term queries [10, 11]. Thus, for a large percentage of all Internet queries, our method guarantees optimal identification when the maximum normalized weight of each term is utilized.

4 Discussion on Applicability

We now discuss several issues concerning the applicability of the new method.

4.1 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 maximum normalized weight) also occupies 4 bytes. Consider a database with m distinct terms. For the subrange-based method, m probabilities, m average weights, m standard derivations and m maximum normalized weights are stored in the database representative, resulting in a total storage overhead of $20 * m$ bytes. The following table shows, for several document collections, the percentage of the sizes of the database representatives based on our approach relative to the sizes of the original document collections.

collection name	collection size	#distinct terms	representative size	percentage
WSJ	40605	156298	1563	3.85
FR	33315	126258	1263	3.79
DOE	25152	186225	1862	7.40

In the above table, all sizes are in pages of 2 KB. The statistics of the second and third columns of the three document collections, namely, WSJ (Wall Street Journal), FR (Federal Register) and DOE (Department of Energy), were collected by ARPA/NIST [8]. The above table shows that for the three databases, the sizes of the representatives range from 3.85% to 7.40% of the sizes of the actual databases. Therefore, our approach is 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 remains unchanged or grows slowly.

In comparison to the database representative used in gGlOSS, the size of the database representative for our approach is 67% larger (due to storing the standard deviation and the maximum normalized weight for each term). The following methods can be used to substantially reduce the size of the database representative.

There are several ways to reduce the size of a database representative. Instead of using 4 bytes for each number (probability, average weight, standard deviation, maximum normalized weight), a one-byte number can be used to approximate it as follows. Consider probability first. Clearly, all probabilities are in interval [0, 1]. Using one byte, 256 different values can be represented. Based on this, interval [0, 1] is partitioned into 256 equal-length intervals. Next, the average of the probabilities falling into each small interval can be computed. Finally, we map each original probability to the average of its corresponding interval. The probability 0.15, for example, lies in the 39-th interval ($[0.1484, 0.1523]$). In the database representative, this probability will be represented by the number 38 (using one byte). Suppose the average of all probabilities in the 39-th interval is 0.1511. Then, 0.1511 will be used to approximate the probability 0.15. Similar approximation can also be applied to average weights, maximum normalized weights and standard deviations. Our experimental results show (see Section 5) that the approximation has negligible impact on the estimation accuracy of database usefulness. When the above scheme is used, the size of the representative of a database with m distinct terms drops to $8 * m$ bytes from $20 * m$ bytes. As a result, the sizes of the database representatives for the above databases will be about 1.5% to 3% of the database sizes. Further size reduction is possible by using 4 bits for each weight, maximum normalized weight and standard deviation. Our experimental results show (also in Section 5) that good accuracy can still be obtained with the reduction. When 4 bits are used for each weight, maximum normalized weight and standard deviation while each probability still uses one byte, the size of the representative of a database with m distinct terms drops to $6.5 * k$ bytes, reducing the above percentages further to 1.23% to 2.4%. As mentioned above, for larger databases, the database representatives are likely to occupy even lower percentages of space.

4.2 Hierarchical Organization of Representatives

If the number of search engines is very large, the representatives can be clustered to form a hierarchy of representatives. Each query is first compared against the highest level representatives. Only representatives whose ancestor representatives have been estimated to have a large number of very similar documents will be examined further. As a result, most database representatives will not be compared against the query. Similar idea has also been suggested by others [6].

Suppose $P_1, ..., P_v$ are the representatives of v local databases $D_1, ..., D_v$. A higher level representative P above these representatives in the hierarchy can be considered as a representative of a database D, where D is combined from $D_1, ..., D_v$ by a union. We now discuss how to obtain P from $P_1, ..., P_v$. We assume that databases $D_1, ..., D_v$ are pair-wise disjoint. Let T_i be the set of terms in D_i, $i = 1, ..., v$. For a given term t, let $p_i(t)$ be the probability of a document in D_i that contains t; $w_i(t)$ be the average weight of t in all documents in D_i that contain t; $mw_i(t)$ be the maximum normalized weight of t in all documents in D_i; and $\sigma_i(t)$ be the standard deviation of all positive weights of t in D_i. Let $p(t), w(t), mw(t)$ and $\sigma(t)$ be the probability, average weight, maximum normalized weight, and standard deviation of term t in the new representative P, respectively. We now discuss how to obtain $p(t), w(t), mw(t)$ and $\sigma(t)$. To simplify the notation, assume that $p_i(t) = w_i(t) = mw_i(t) = \sigma_i(t) = 0$ if t is not a term in T_i, $i = 1, ..., v$.

The first three quantities, namely $p(t), w(t)$ and $mw(t)$, can be obtained easily.

$$p(t) = \frac{\sum_{i=1}^{v} p_i(t) * n_i}{\sum_{i=1}^{v} n_i}$$

where n_i is the number of documents in D_i.

$$w(t) = \frac{\sum_{i=1}^{v} p_i(t) * n_i * w_i(t)}{\sum_{i=1}^{v} p_i(t) * n_i}$$

$$mw(t) = \max\{mw_1(t), ..., mw_v(t)\}$$

We now compute $\sigma(t)$. Let $k_i(t)$ be the number of documents containing t in D_i. Note that $k_i(t) = p_i(t) * n_i$. Let $w_{ij}(t)$ denote the weight of t in the jth document in D_i.

¿From $\sigma_i(t) = \sqrt{\frac{\sum_j w_{ij}^2(t)}{k_i(t)} - w_i^2(t)}$, we have

$$\frac{\sum_j w_{ij}^2(t)}{k_i(t)} = \sigma_i^2(t) + w_i^2(t) \tag{12}$$

Based on the definition of standard deviation, we have

$$\sigma(t) = \sqrt{\frac{\sum_i \sum_j w_{ij}^2(t)}{\sum_i k_i(t)} - \left(\frac{\sum_i \sum_j w_{ij}(t)}{\sum_i k_i(t)}\right)^2} = \sqrt{\frac{\sum_j w_{1j}^2(t)}{\sum_i k_i(t)} + \cdots + \frac{\sum_j w_{vj}^2(t)}{\sum_i k_i(t)} - w^2(t)}$$

$$= \sqrt{\frac{k_1(t)}{\sum_i k_i(t)} * \frac{\sum_j w_{1j}^2(t)}{k_1(t)} + \cdots + \frac{k_v(t)}{\sum_i k_i(t)} * \frac{\sum_j w_{vj}^2(t)}{k_v(t)} - w^2(t)}$$

¿From Equation (12), we have

$$\sigma(t) = \sqrt{\frac{k_1(t)}{\sum_i k_i(t)} * (\sigma_1^2(t) + w_1^2(t)) + \cdots + \frac{k_v(t)}{\sum_i k_i(t)} * (\sigma_v^2(t) + w_v^2(t)) - w^2(t)}$$

The above derivations show that each quantity in the representative P can be computed from the quantities in the representatives at the next lower level.

4.3 Obtaining Database Representatives

To obtain the accurate representative of a database used by our method, we need to know the following information: (1) the number of documents in the database; (2) the *document frequency* of each term in the database (i.e., the number of documents in the database that contain the term); and (3) the weight of each term in each document in the database. (1) and (2) are needed to compute the probabilities and (3) is needed to compute average weights, maximum normalized weights and standard deviations. (1) and (2) can usually be obtained with ease. For example, when a query containing a single term is submitted to a search engine, the number of hits returned is the document frequency of the term. Many other proposed approaches for ranking text databases also use the document frequency information [3, 6, 27]. Most recently, the STARTS proposal for Internet metasearching [5] suggests that each database (source) should provide document frequency for each term.

We now discuss how to obtain information (3). In an Internet environment, it may not be practical to expect a search engine to provide the weight of each term in each document in the search engine. We propose the following techniques for obtaining the average term weights, their standard deviations and the maximum normalized term weights.

1. Use sampling techniques in statistics to estimate the average weight and the standard deviation for each term. When a query is submitted to a search engine, a set S of documents will be returned as the result of the search. For each term t in S and each document d in S, the term frequency of t in d (i.e., the number of times t appears in d) can be computed (the STARTS proposal even suggests that each search engine provides the term frequency and weight information for each term in each returned document [5]). As a result, the weight of t in d can be computed. If the weights of t in a reasonably large number of documents can be computed (note that more than one query may be needed), then an approximate average weight and an approximate standard deviation for term t can be obtained. Since the returned documents for each query may contain many different terms, the above estimation can be carried out for many terms at the same time.

2. Obtain the maximum normalized weight (with respect to the global similarity function used in the metasearch engine) for each term t directly as follows. Submit t as a single term query to the local search engine which retrieves documents according to a local similarity function. Two cases are considered:

   Case 1: The global similarity function is known to be the same as the similarity function in the search engine. In this case, if the search engine returns a similarity for each retrieved document, then the similarity returned for the first retrieved document is the maximum normalized weight for the term; if the search engine does not return similarity explicitly, then the first retrieved document is downloaded to compute its similarity with the one-term query and this similarity will be the maximum normalized weight for the term.

Case 2: The global similarity function and the local similarity function are different or the local similarity function is unknown. In this case, the first few retrieved documents are downloaded to compute the global similarities of these documents with the one-term query. The largest similarity is then tentatively used as the maximum normalized weight, which may need to be adjusted when another document from the same search engine is found to have a higher weight for the term (with respect to the global similarity function).

Although using sampling techniques can introduce inaccuracy to the statistical data (e.g., average weight and standard deviation), our usefulness estimation method is quite robust with respect to the inaccuracy as a 4-bit approximation of each value can still produce reasonably accurate usefulness estimation. Furthermore, a recent study indicates that using sampling queries is capable of generating decent statistical information for terms [4].

5 Experimental Results

Three databases, D1, D2 and D3, and a collection of 6,234 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. 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. The queries are real queries submitted by users to the SIFT Netnews server [23, 6]. Since most user queries in the Internet environment are short [1, 14], only queries with no more than 6 terms are used in our experiments. Approximately 30% of the 6,234 queries in our experiments are single-term queries.

For all documents and queries, non-content words such as "the", "of", etc. are removed. The similarity function is the *Cosine* function. This function guarantees that the similarity between any query and document with non-negative term weights will be between 0 and 1. As a result, no threshold larger than 1 is needed.

We first present the experimental results when the database representative is represented by a set of quadruplets $(w_i, p_i, \sigma_i, mw_i)$ (average normalized weight, probability, standard deviation, maximum normalized weight) and each number is the original number (i.e., no approximation is used). The results will be compared against the estimates generated by the method for the high-correlation case and our previous method proposed in [18]. (The method in [18] is similar to the basic method described in Section 3.1 of this paper except that it utilizes the standard deviation of the weights of each term in all documents to dynamically adjust the average weight and probability of each query term according to the threshold used for the query. Please see [18] for details. No experimental results for the method for the disjoint case [6] will be reported here as we have shown that the method for the high-correlation case performs better than that for the disjoint case [18].) We then present the results when the database representative is still represented by a set of quadruplets but each original number is approximated either by a one-byte number or a 4-bit number. This is to investigate whether our estimation method can tolerate certain degree of inaccuracy on the numbers used in the database representative. These experiments use six subranges for our subrange-based method. The first subrange contains only the maximum normalized term weight; the other subranges have medians at 98 percentile, 93.1 percentile, 70 percentile, 37.5 percentile and 12.5 percentile, respectively. Note that narrower subranges are used for weights that are large because those weights are often more important for estimating database usefulness, especially when the threshold is large. Finally, we present the results when the database representative is represented by a set of triplets (w_i, p_i, σ_i) and each number is the original number. In other words, the maximum normalized weight is not directly obtained but is estimated to be the 99.9 percentile from the average weight and the standard deviation. The experimental results show the importance of maximum normalized weights in the estimation process. All other medians are the same.

Using Quadruplets and Original Numbers

Consider database D1. For each query and each threshold, four usefulnesses are obtained. The first is the true usefulness obtained by comparing the query with each document in the database. The other three are estimated based on the database representatives and estimation formulas of the following methods: (1) the method for the high-correlation case; (2) our previous method [18]; and (3) our subrange-based method with the database representative represented by a set of quadruplets and each number being the original number. All estimated usefulnesses are rounded to integers. The experimental results for D1 are summarized in Table 2.

		high-correlation			our previous method			subrange-based method		
T	U	match/mismatch	d-N	d-S	match/mismatch	d-N	d-S	match/mismatch	d-N	d-S
0.1	1474	296/35	16.87	0.121	767/14	9.29	0.078	1421/1	6.74	0.017
0.2	433	24/3	17.61	0.242	180/0	8.91	0.159	413/1	7.89	0.030
0.3	162	5/1	20.28	0.354	49/2	9.79	0.261	153/0	9.76	0.042
0.4	56	1/0	17.14	0.470	20/1	8.57	0.325	51/0	9.54	0.062
0.5	30	0/0	3.87	0.586	11/0	3.70	0.401	24/0	3.83	0.130
0.6	12	0/0	1.50	0.692	0/0	1.50	0.692	6/0	0.92	0.323

Table 2: Comparison of Different Estimation Methods Using D1

In Table 2, T is the threshold and U is the number of queries that identify D1 as useful (D1 is useful to a query if there is at least one document in D1 which has similarity greater than T with the query, i.e., the actual *NoDoc* is greater than 1). When T = 0.1, 1,474 out of 6,234 queries identify D1 as useful. The comparison of different approaches are based on the following three different criteria.

match/mismatch: For a given threshold, "match" reports among the queries that identify D1 as useful based on the true *NoDoc*, the number of queries that also identify D1 as useful based on the estimated *NoDoc*; "mismatch" reports the number of queries that identify D1 as useful based on the estimated *NoDoc* but in reality D1 is not useful to these queries based on the true *NoDoc*. For example, consider the "match/mismatch" column using the method for the high-correlation case. When T = 0.1, "296/35" means that out of the 1,474 queries that identify D1 as useful based on the true *NoDoc*, 296 queries also identify D1 as useful based on the estimated *NoDoc* by the high-correlation approach; and there are also 35 queries that identify D1 as useful based on the high-correlation approach but in reality, D1 is not useful to these 35 queries. Clearly, a good estimation method should have its "match" close to "U" and its "mismatch" close to zero for any threshold. Note that in practice, correctly identifying a useful database is 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 much larger "match" component than method B while A's "mismatch" component is not significantly larger than B's "mismatch" component, then A should be considered to be better than B.

Table 2 shows that the subrange-based approach is substantially more accurate than our previous method [18] which in turn is substantially more accurate than the high-correlation approach under the "match/mismatch" criteria. In fact, for thresholds between 0.1 and 0.4, the accuracy of the subrange-based method is 91% or higher for the "match" category.

d-N: For each threshold T, the "d-N" (for "difference in *NoDoc*") column for a given estimation method indicates the average difference between the true *NoDoc* and the estimated *NoDoc* over the queries that identify D1 as useful based on the true *NoDoc*. For example, for T = 0.1, the average difference is over the 1,474 queries. The smaller the number in "d-N" is, the better the corresponding estimation method is. Again, Table 2 shows that the subrange-based approach is better than our previous method for most thresholds which in turn is much better than the high-correlation approach under the "d-N" criteria.

d-S: For each threshold T, the "d-S" (for "difference in *AvgSim*") column for a given estimation method indicates the average difference between the true *AvgSim* and the estimated *AvgSim* over the queries that identify D1 as useful based on the true *NoDoc*. Again, the smaller the number in "d-S" is, the better the corresponding estimation method is. Table 2 shows that the subrange-based approach is substantially more accurate than the other two approaches for all thresholds.

The experimental results for databases D2 and D3 are summarized in Table 3 and Table 4, respectively. From Tables 2 to 4, the following observations can be made. First, the subrange-based estimation method significantly outperformed the other two methods for each database under each criteria. Second, the "match" components are best for database D1, and not as good for database D2 and for database D3. This is probably due to the inhomogeneity of data in databases D2 and D3.

		high-correlation			our previous method			subrange-based method		
T	U	match/mismatch	d-N	d-S	match/mismatch	d-N	d-S	match/mismatch	d-N	d-S
0.1	2592	779/102	26.96	0.112	1299/148	20.31	0.082	2552/0	11.74	0.020
0.2	1149	30/7	19.56	0.252	321/41	9.80	0.191	1067/2	8.35	0.040
0.3	525	4/2	13.00	0.347	104/14	7.64	0.282	424/0	6.66	0.084
0.4	134	1/0	11.13	0.458	27/1	6.49	0.374	98/0	4.54	0.145
0.5	55	0/0	5.43	0.550	9/1	3.67	0.463	38/1	4.53	0.196
0.6	15	0/0	3.07	0.664	4/0	2.21	0.492	8/0	2.40	0.317

Table 3: Comparison of Different Estimation Methods Using D2

		high-correlation			our previous method			subrange-based method		
T	U	match/mismatch	d-N	d-S	match/mismatch	d-N	d-S	match/mismatch	d-N	d-S
0.1	2707	760/135	17.44	0.114	1379/192	13.96	0.081	2638/1	7.56	0.020
0.2	1146	46/23	12.47	0.245	277/55	7.16	0.198	1019/3	5.71	0.046
0.3	420	6/5	10.92	0.354	76/12	6.76	0.297	334/4	5.39	0.092
0.4	144	0/1	7.18	0.460	17/6	4.89	0.405	103/1	3.64	0.148
0.5	46	0/0	3.77	0.558	8/0	2.81	0.472	30/0	2.56	0.213
0.6	15	0/0	2.20	0.659	3/0	3.20	0.534	6/0	1.93	0.406

Table 4: Comparison of Different Estimation Methods Using D3

Using Quadruplets and Approximate Numbers

In Section 4.1, we proposed a simple method to reduce the size of a database representative by approximating each needed number (such as average weight) using one byte or 4 bits. When the accuracies of the parameter values are reduced from 4 bytes to 1 byte each, there is essentially no difference in performance (see Table 5 relative to Table 2). Table 6 lists the experimental results when all numbers are represented by 4 bits except that each probability continues to use 1 byte, again for database D1. The results (compare Tables 5 and 6 with Table 2) show that the drop in accuracy of estimation due to approximation is small for each criteria.

T	match/mismatch	d-N	d-S
0.1	1423/13	6.79	0.017
0.2	421/2	7.64	0.030
0.3	153/3	7.69	0.042
0.4	52/0	9.50	0.055
0.5	24/0	3.77	0.130
0.6	6/0	0.92	0.323

Table 5: Using One Byte for Each Number for D1

T	match/mismatch	d-N	d-S
0.1	1371/12	7.07	0.026
0.2	413/2	8.37	0.037
0.3	144/3	9.07	0.062
0.4	51/12	9.54	0.069
0.5	26/0	3.60	0.102
0.6	6/0	0.92	0.332

Table 6: Using 4 Bits for Each Number (Each Probability Uses One Byte) for D1

Similar but slightly weaker results can be obtained for databases D2 and D3 (see Tables 7 and 8 relative to Table 3 for D2 and Tables 9 and 10 relative to Table 4 for D3).

Using Triplets and Original Numbers

In Section 3.2.1, we discussed the importance of the maximum normalized weights for correctly identifying useful databases, especially when single term queries are used. Since single term queries represent a large fraction of all queries in the Internet environment, it is expected that the use of maximum normalized weights

T	match/mismatch	d-N	d-S
0.1	2353/214	12.19	0.026
0.2	1002/79	8.35	0.047
0.3	401/29	7.03	0.088
0.4	97/1	4.59	0.152
0.5	38/1	4.59	0.187
0.6	8/0	2.50	0.291

Table 7: Using One Byte for Each Number for D2

T	match/mismatch	d-N	d-S
0.1	2287/215	13.20	0.033
0.2	975/78	8.88	0.055
0.3	390/23	7.83	0.098
0.4	102/17	4.63	0.141
0.5	38/1	4.65	0.192
0.6	7/0	2.57	0.345

Table 8: Using 4 Bits for Each Number (Each Probability Uses One Byte) for D2

T	match/mismatch	d-N	d-S
0.1	2411/280	8.03	0.027
0.2	966/76	5.74	0.054
0.3	310/21	5.56	0.095
0.4	93/7	3.85	0.158
0.5	30/0	2.52	0.225
0.6	6/0	1.80	0.409

Table 9: Using One Byte for Each Number for D3

T	match/mismatch	d-N	d-S
0.1	2332/259	8.15	0.032
0.2	935/77	6.09	0.061
0.3	311/24	5.56	0.096
0.4	92/14	3.86	0.162
0.5	30/3	2.46	0.229
0.6	8/0	1.67	0.333

Table 10: Using 4 Bits for Each Number (Each Probability Uses One Byte) for D3

will significantly improve the overall estimation accuracy for all queries. Among 6,234 queries used in our experiments, 1,941 are single term queries. Table 11 shows the experimental results for database D1 when the maximum normalized weights are not explicitly obtained. (Instead, it is assumed that for each term, the normalized weights of the term in the set of documents containing the term satisfy a normal distribution and therefore the maximum normalized weight is estimated to be the 99.9 percentile based on its average weight and its standard deviation.) Comparing the results in Table 2 and those in Table 11, it is clear that the use of maximum normalized weights can indeed improve the estimation accuracy substantially. Nevertheless, even when estimated maximum normalized weights are used, the results based on the subrange-based approach are still much better than those based on the high-correlation assumption and those obtained by our previous method [18] . Similar conclusion can be reached when the results in Table 3 and Table 12 are compared and when the results in Table 4 and Table 13 are compared.

T	match/mismatch	d-N	d-S
0.1	891/12	7.38	0.067
0.2	189/0	7.97	0.154
0.3	60/1	8.33	0.239
0.4	24/0	9.98	0.293
0.5	12/1	4.23	0.390
0.6	1/2	1.23	0.641

Table 11: Result for D1 When Maximum Weights Are Estimated.

T	match/mismatch	d-N	d-S
0.1	1691/175	12.55	0.062
0.2	442/47	8.96	0.165
0.3	117/10	7.56	0.272
0.4	34/1	4.85	0.353
0.5	12/3	4.91	0.439
0.6	5/1	2.29	0.440

Table 12: Result for D2 When Maximum Weights Are Estimated.

T	match/mismatch	d-N	d-S
0.1	1851/205	8.50	0.058
0.2	291/50	6.43	0.194
0.3	76/15	6.19	0.294
0.4	30/3	4.23	0.365
0.5	10/0	2.85	0.446
0.6	3/0	2.00	0.536

Table 13: Result for D3 When Maximum Weights Are Estimated.

6 Combining Terms

In the subrange-based estimation method presented earlier, terms are assumed to be independently distributed in the documents of the database. Although the overall experimental results reported in Section 5 are very good, there are some room for improvement at high retrieval thresholds (see Tables 2, 3 and 4 with threshold values 0.5 and 0.6.) Thus, we propose the following scheme to incorporate the dependencies of terms in the estimation process.

There are quite a few term dependency models in the information retrieval literature (see, for example, the tree-dependency model, the Bahadur-Lazarsfeld model and the generalized dependency model in [26].) In [18], we employ the Bahadur-Lazarsfeld model to incorporate the dependencies into the estimation process. That model is somewhat complicated. In addition, it does not make use of the maximum normalized term weight. As the experimental results in the last section indicate, the maximum normalized term weight is a critical parameter. Thus, the following approach is used instead.

Consider the distributions of terms t_i and t_j in a database of documents. Within the set of documents having both terms, there is a document having the largest sum of the normalized term weight of t_i and the normalized term weight of t_j. Let the largest sum be called the *maximum normalized weight* of the combined term and be denoted by mnw_{ij}. If terms t_i and t_j are combined into a single term, then the probability that a document in the database has the maximum normalized weight of the combined term, mnw_{ij}, can be assumed to be $1/n$, where n is the number of documents in the database. As pointed out earlier, it is unlikely that another document in the database has the same maximum normalized weight under the combined term. If the two terms were independently distributed in the documents of the database, then the probability that a document in the database has the normalized sum of term weights mnw_{ij} under the two terms t_i and t_j can be estimated using the subrange-based estimation method. Specifically, a polynomial representing the probability that a document has the maximum normalized term weight for t_i, followed by the probabilities that a document has certain percentiles of weights for term t_i can be written (see Example 4). Similarly, another such polynomial can be written for term t_j. By multiplying these two polynomials together, the desired probability can be estimated. The criteria that the two terms t_i and t_j should be combined into a single term t_{ij} is that the estimated probability under the term independence assumption is very different from $1/n$ and the maximum normalized weight of the combined term is higher than the maximum normalized weight of each of the two individual terms. Since our aim is to estimate the similarities of the most similar documents, the latter condition is to ensure that if the combined term is used, it will not lead to smaller similarities. The former condition is implemented by computing the difference in absolute value between $1/n$ and the estimated probability and then comparing to a pre-set threshold. If the difference exceeds the threshold, then the two terms should be combined. The difference for the term pair t_i and t_j is denoted by d_{ij} and is stored together with the combined term t_{ij}.

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

Example 5 Suppose that the user's query q is "computer algorithm", and that normalized term weight is used in this example.

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

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

and

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

Suppose that the maximum normalized weight of the combined term "computer algorithm" $mnw_{12} = 0.825$ which is greater than mw_1 and mw_2. By multiplying the above polynomials, the probability that a document has a total normalized weight (associated with these two terms) of mnw_{12} or higher is $3.878 * 10^{-5}$. This probability is based on the assumption that the two terms were independent. The actual probability is $1/n = 0.0013$, where $n = 761$ in this example. Since the estimated probability and the actual probability differ substantially, the two terms should be combined. The combined term occurs in 53 documents out of a total of 761 documents. Its average normalized weight is 0.352; and the standard deviation of the normalized weights is 0.203. Using the subrange-based method on the combined term, the first subrange contains the maximum normalized sum of 0.825 and the probability associated with this subrange is 0.0013. The second subrange has its median at the 98-th percentile. From the standard normal distribution table, the constant c for the 98-th percentile is 2.055. Thus, the median weight is $0.352 + 2.055 * 0.203 = 0.769$. Note that the larger end point of the second subrange corresponds to the $(1 - \frac{1}{53}) * 100$-th percentile. Since the median is at the 98th percentile, the width of the second subrange is $2 * \left((1 - \frac{1}{53}) * 100 - 98\right)/100 = (\frac{4}{100} - \frac{2}{53})$. Thus, the probability that the normalized weight of the combined term in a document lies in the second subrange (having median at the 98-th percentile) is $\left(\frac{4}{100} - \frac{2}{53}\right) * \frac{53}{761} = 0.000158$. The median weights and probabilities for the other subranges can be determined similarly. Thus, the polynomial for the combined term is

$$0.0013 * X^{0.825} + 0.000158 * X^{0.769} + 0.0055 * X^{0.667} + 0.0279 * X^{0.458} + 0.0174 * X^{0.287} + 0.0174 * X^{0.118} + 0.93$$

■

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

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

Using this strategy to combine terms, we perform experiments on the same set of queries and the same three databases D1, D2 and D3. The results are reported in Tables 14, 15 and 16. It is clear that the combined-term method is better than the subrange-based method in the match/mismatch measure, especially when the thresholds (0.5 and 0.6) are large. Close to optimal results are obtained.

Theoretically, it is possible to get better results by (1) combining three or more terms together and (2) modify the polynomial representing two terms when they are combined. It should be noted that the polynomial for the combined term does not take into consideration the situations that exactly one of the two terms occurs. It is possible to include those situations. However, that would require storing more information. Similarly, the process of combining 3 or more terms into one is feasible but would introduce complications. Since the simple combined-term method yields close to optimal results, it is not clear whether it is worthwhile to complicate the estimation process.

		subrange-based method			combined-term method		
T	U	match/mismatch	d-N	d-S	match/mismatch	d-N	d-S
0.1	1474	1421/1	6.74	0.017	1455/0	8.57	0.015
0.2	433	413/1	7.89	0.030	429/1	9.65	0.023
0.3	162	153/0	9.76	0.042	160/0	9.70	0.028
0.4	56	51/0	9.54	0.062	56/0	9.89	0.027
0.5	30	24/0	3.83	0.130	29/0	3.63	0.039
0.6	12	6/0	0.92	0.323	10/0	0.50	0.112

Table 14: Comparison of Different Estimation Methods Using D1

		subrange-based method			combined-term method		
T	U	match/mismatch	d-N	d-S	match/mismatch	d-N	d-S
0.1	2592	2552/0	11.74	0.020	2586/0	11.49	0.018
0.2	1149	1067/2	8.35	0.040	1122/1	9.64	0.028
0.3	525	424/0	6.66	0.084	487/0	7.34	0.044
0.4	134	98/0	4.54	0.145	123/0	4.85	0.067
0.5	55	38/1	4.53	0.196	46/1	4.02	0.114
0.6	15	8/0	2.40	0.317	13/0	1.73	0.103

Table 15: Comparison of Different Estimation Methods Using D2

		subrange-based method			combined-term method		
T	U	match/mismatch	d-N	d-S	match/mismatch	d-N	d-S
0.1	2707	2638/1	7.56	0.020	2692/1	9.43	0.017
0.2	1146	1019/3	5.71	0.046	1113/0	6.69	0.026
0.3	420	334/4	5.39	0.092	393/4	6.10	0.042
0.4	144	103/1	3.64	0.148	127/1	3.62	0.073
0.5	46	30/0	2.56	0.213	44/0	1.89	0.041
0.6	15	6/0	1.93	0.406	14/0	1.53	0.063

Table 16: Comparison of Different Estimation Methods Using D3

7 Conclusions

In this paper, we introduced a search engine usefulness measure which is intuitive and easily understood by users. We proposed a statistical method to estimate the usefulness of a given search engine with respect to each query. Accurate estimation of the usefulness measure allows a metasearch engine to send queries to only the appropriate local search engines to be processed. This will save both the communication cost and the local processing cost substantially. Our estimation method has the following properties:

1. The estimation makes use of the number of documents desired by the user (or the threshold of retrieval), unlike some other estimation methods which rank search engines without using the above information.

2. It guarantees that those search engines containing the most similar documents are correctly identified, when the submitted queries are single-term queries. Since Internet users submit a high percentage of such short queries, they can be sent to the correct search engines to be processed.

3. Experimental results indicate that our estimation methods are much more accurate than existing methods in identifying the correct search engines to use, in estimating the number of potentially useful documents in each database, and in estimating the average similarity of the most similar documents.

We intend to fine-tune our algorithm to yield even better results and to perform experiments involving much larger and many more databases. Our current experiments are based on the assumption that term weights satisfy the normal distribution. However, the Zipfian distribution [20] may model the weights more accurately. We will also examine ways to further reduce the storage requirement for database representatives.

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

We are grateful to Luis Gravano and Hector Garcia-Molina of Stanford University for providing us with the database and query collections used in [6].

References

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

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

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

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

[5] L. Gravano, C. Chang, H. Garcia-Molina, and A. Paepcke. *STARTS: Stanford Proposal for Internet Meta-Searching.* ACM SIGMOD Conference, Tucson, May 1997, pp. 207-218.

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

[7] L. Gravano, and H. Garcia-Molina. *Generalizing GlOSS to Vector-Space databases and Broker Hierarchies.* Technical Report, Computer Science Dept., Stanford University, 1995. (This report discussed how to estimate the database usefulness used defined in this paper for the high-correlation and disjoint scenarios. Such discussion did not appear in [6].)

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

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

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

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

[13] 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).

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

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

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

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

[18] W. Meng, K. Liu, C. Yu, X. Wang, Y. Chang, and N. Rishe. *Determining Text Databases to Search in the Internet.* International Conferences on Very Large Data Bases, New York City, 1998.

[19] W. Meng, K. Liu, C. Yu, W. Wu, and N. Rishe. *Estimating the Usefulness of Search Engines.* IEEE International Conference on Data Engineering, pp. 146-153, Sydney, Australia, 1999.

[20] C.J. Van Rijsgergen. Information Retrieval. Hyper-text book (http://www.dei.unipd.it/~melo/bible/bible_home_page.html).

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

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

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

[24] C. Yu, K. Liu, W. Wu, W. Meng, and N. Rishe. *Finding the Most Similar Documents across Multiple Text Databases*. IEEE Conference on Advances in Digital Libraries, pp. 150-162, Baltimore, Maryland, 1999.

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

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

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