Estimating the Usefulness of Search Engines*

Weiyi Meng[1], King-Lup Liu[2], Clement Yu[2], Wensheng Wu[2], Naphtali Rishe[3]
[1] Dept. of Computer Science, SUNY – Binghamton, Binghamton, NY 13902
[2] Dept. of EECS, University of Illinois at Chicago, Chicago, IL 60607
[3] School of Computer Science, Florida International University, Miami, FL 33199

Abstract

In this paper, we present a statistical method to estimate the usefulness of a search engine for any given query. The estimates can be used by a metasearch engine to choose local search engines to invoke. 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 the proposed estimation method is quite accurate.

1 Introduction

Many *search engines* have been created on the Internet to help ordinary users find desired data. 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 to speed up query processing.

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 [13]) and is increasing rapidly. Employing a single search engine for all data in the Internet is unrealistic because its processing power and storage capability may not scale to the virtually unlimited amount of data. In addition, gathering all data in the Internet and keeping them reasonably up-to-date are extremely difficult if not impossible.

A more practical approach to provide search services to the entire Internet is the following two-level approach (the approach can be generalized to more than two levels). At the bottom level are the local search engines. A *metasearch engine* is built on top of them. A metasearch engine is just an interface and it does not maintain its own index on documents. However, a sophisticated metasearch engine may maintain certain characteristic information about each local search engine (called the *representative* of the search engine) to provide better service. When a metasearch engine receives a user query, it first passes the query to the appropriate local search engines, and then collects (sometimes, reorganizes) the results from the search engines used. The advantages of this approach are (a) user queries can 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 the 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.

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. If every search engine is blindly invoked for each query, then 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 the identified search engines. A challenge with this approach is how to identify potentially useful search engines.

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 whose *similarities* with the query as measured by a certain global similarity function are higher than a specified threshold and *AvgSim* is the average similarity of these high-similarity documents. (The global similarity function may or may not be the same as the local similarity function employed by a local search engine.) *NoDoc* and *AvgSim* are defined precisely below:

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

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

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

where $|X|$ denotes the number elements in set X, T is a threshold, D is the database of a search engine and $sim(q, d)$ is the similarity 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 [17]. A document is also transformed into a vector with weights. The similarity between a query and a document can be measured by the *dot product* of the two vectors. Often, the dot product is divided by the product of the *norms* of the two vectors, where the norm of a vector $(x_1, ..., x_n)$ is $\sqrt{\sum_{i=1}^{n} x_i^2}$. This is to normalize the similarity between 0 and 1. The similarity function with such a normalization is known as the *Cosine* function [17,19].

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. Second, a new statistical 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 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 $\max\_sim_i$. Suppose $\max\_sim_i > \max\_sim_j$ and a threshold of retrieval T is set such that $\max\_sim_i > T > \max\_sim_j$, 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. In other words, our method identifies correct search engines to invoke for single-term queries which constitutes a significant portion of all Internet queries. This is consistent to the ideal situation where documents are examined in descending order of similarity. In addition, the new method is quite robust as it can still yield good result even when approximate statistical data are used.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents our method for estimating the usefulness of search engines. Experimental results will be presented in Section 4. Section 5 concludes the paper.

2 Related Work

Several methods have been proposed to identify potentially useful search engines in a metasearch environment [4,7,10,11,20]. However, the database representatives in most methods cannot be used to estimate the number of globally most similar documents in each search engine [3,10,11,20]. The measures used by these methods to rank the search engines are difficult to understand and a separate method has 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. 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.

The usefulness of a search engine with respect to a given query in gGlOSS [4] 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. The representative of gGlOSS can be used to estimate the number of useful documents in a database [5] 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 [4,5] 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. 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. However, 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.

[18] 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 [14], 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 [15] permits term weights to be non-binary but it utilizes the non-binary information in a way which is very different from the subrange-based method to be described in Section 3.

3 A New Method for Usefulness Estimation

We present our method for estimating the usefulness of a search engine in Section 3.1. A simple technique for improving the scalability of the method will be discussed in Section 3.2.

3.1 The Subrange-based 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, $1 \leq i \leq m$. 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 $sim(q, d) = u_1 * d_1 + \cdots + u_m * d_m$. Similarities are often normalized between 0 and 1. One common normalized similarity function is the *Cosine* function [17].

We first consider the case when the 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 can be 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:

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

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 [15].

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.

Example 3.1: Let q be a query with three terms with all weights equal to 1, i.e., $q = (u_1, u_2, u_3) = (1, 1, 1)$ (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 the 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:

$$(0.6 * X^2 + 0.4)(0.2 * X + 0.8)(0.4 * X^2 + 0.6) \quad (4)$$

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 $w_1 * u_1$ (=2). The latter is the probability that a document in D has exactly the last query term and the corresponding similarity is $w_3 * u_3$ (=2). Therefore, the coefficient of X^2 (= 0.416) is the estimated probability that a document in D has similarity 2 with q. ■

After expanding the generating function (3) and merging terms with the same X^s, we obtain

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

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 expected number of documents whose similarity with query q is greater than T is $\sum_{i=1}^{C}(n * a_i)$. That is, the *NoDoc* measure of D for query q based on threshold T (i.e. the number of documents in database D whose similarities with q are greater than T) can be estimated as:

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

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. Then $\sum_{i=1}^{C}(n * a_i * b_i)$ 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 (i.e., the average similarity of those documents in database D whose similarities with q are greater than T) can be estimated as:

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

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 3.2: (Continue Example 3.1). When the generating function (4) is expanded, we have:

$$0.048 * X^5 + 0.192 * X^4 + 0.104 * X^3 + 0.416 * X^2 + 0.048 * X + 0.192$$

From the above estimation formulas, 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 there is one document having similarity higher than 3 with q (the fourth and the similarity is 4) and the actual *AvgSim* is $AvgSim(3, q, D) = 4$. ■

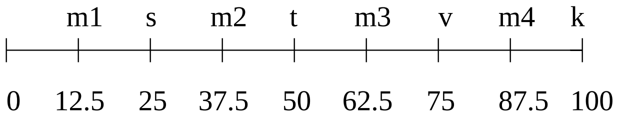
One unrealistic assumption used in Proposition 1 is that all documents having a term have the same weight for the term. We now present a subrange-based statistical method to overcome the problem.

Consider a term t. Let w and σ be the average and 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 above discussion, if term t is specified in a query, then the following polynomial is included in the probability generating function (see Expression (3)):

$$p * X^{u*w} + (1-p) \tag{7}$$

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 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 weight 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 are 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 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 (7) in the generating function can be replaced by the following polynomial:

$$p_1 * X^{u*w_{m1}} + p_2 * X^{u*w_{m2}} + p_3 * X^{u*w_{m3}}$$
$$+ p_4 * X^{u*w_{m4}} + (1-p) \tag{8}$$

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 (8) is obtained from polynomial (7) 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 (8) 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 weight distribution of the term is normal. Then $w_{mi} = w + c_i * \sigma$, where σ is the standard deviation and 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 3.3: 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. In the experiments we report in Section 4, a specific six-subrange is used with a special subrange (i.e., the highest subrange) containing the maximum normalized weight only (see Section 4). 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.

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 $\{(p_i, w_i, \sigma_i)\}$, $i = 1, ..., m$, 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.

We now provide some insight into why the maximum normalized weight is critically important for correctly identifying useful search engines for single-term queries. Consider a query q that contains a single term t. Suppose the similarity function is the widely used *Cosine* function. Then the query has a normalized weight of 1 for t and the similarity of a document d with the query q is w', which is the dot product $1 \cdot w'$, where $w' = w/|d|$ is the normalized weight of t in document d, $|d|$ is the *norm* of d, and w is the weight of the term in d before normalization. Consider a database D_1 that contains documents having term t. Let mw_1 be the maximum normalized weight of term t among all documents in database D_1. Then the component of the database representative concerning term t will contain mw_1. Suppose 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. Then the generating function for the query q for the database D_1 is:

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

where p_1 is set to be $1/n$ 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 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 $mw_2 < T$, 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 can 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 useful. This identification is consistent with the ideal situation where these databases contain documents with similarities greater than T and other databases do not have the desired documents (with similarities greater than T). Thus, when the highest subrange contains the maximum normalized weight only, our method guarantees the correct identification of useful search engines for single term queries. The same argument applies to other similarity functions such as [16]. It is reported in [8,9] that the percentage of single-term queries in the Internet is quite high (30% or much higher). Thus, for a large percentage of all Internet queries, our method guarantees optimal identification when the maximum normalized weight of each term is utilized.

3.2 Improving the 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. Suppose each term occupies four bytes. Suppose each number (probability, average weight, standard deviation and maximum normalized weight) also occupies 4 bytes. Then based on our estimation method, $20 * k$ bytes of space is needed to store the database representative for a database with k different terms. 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. All sizes are in pages of 2 KB.

collection	size	#dist. terms	rep. size	%
WSJ	40605	156298	1563	3.85
FR	33315	126258	1263	3.79
DOE	25152	186225	1862	7.40

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 [6]. The above table shows that for the three databases, the sizes of the representatives range from 3.79% 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.

A simple method to reduce the size of a database representative is as follows. Instead of using 4 bytes for each number, a one-byte number can be used to approximate it. For example, all probabilities are in interval [0, 1]. Using one byte, 256 different values can be represented. Based on this, interval [0, 1] can be 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. Experimental results show (see Section 4) 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 k terms drops to $8 * k$ bytes from $20 * k$ 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.

4 Experimental Results

Three databases, D1, D2 and D3, and a collection of 6,234 queries were used in the experiment. D1, containing 761 documents, is the largest among the 53 databases that were 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, was obtained by merging the two largest databases among the 53 databases. D3, containing 1,014 documents, was 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 [4]. Since most user queries in the Internet environment are short [1,12], 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 which 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 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 [15]. (The method in [15] is similar to the basic method described in Section 3.1 of this paper except that it also 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. No experimental results for the method based on the disjoint case [4] will be reported here as we have shown that this method underperforms the method based on the high-correlation case [15].) We then present the results when the database representative is still represented by a set of quadruplets but each number is approximated by a one-byte number. This is to investigate whether our 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. All other medians are determined in the same manner as before. The experimental results show the importance of maximum normalized weights in the estimation process.

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; 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 Tables 1 and 2.

		high-correlation	our prev method	subrange method
T	U	match/mismatch	match/mismatch	match/mismatch
0.1	1475	296/35	767/14	1423/13
0.2	440	24/3	180/0	421/2
0.3	162	5/1	49/2	153/3
0.4	56	1/0	20/1	52/0
0.5	30	0/0	11/0	24/0
0.6	12	0/0	0/0	6/0

Table 1: Comparison of Match/Mismatch Using D1

		high-correlation		our prev method		subrange method	
T	U	d-N	d-S	d-N	d-S	d-N	d-S
0.1	1475	16.87	0.121	9.29	0.078	7.05	0.017
0.2	440	17.61	0.242	8.91	0.159	7.34	0.029
0.3	162	20.28	0.354	9.79	0.261	7.69	0.042
0.4	56	17.14	0.470	8.57	0.325	9.48	0.054
0.5	30	3.87	0.586	3.70	0.401	3.77	0.130
0.6	12	1.50	0.692	1.50	0.692	0.92	0.323

Table 2: Comparison of d-N and d-S Using D1

In Tables 1 and 2, T is 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 at least 1). When T = 0.1, for instance, 1,475 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 1,475 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 1 shows that the subrange-based approach is substantially more accurate than our previous method which in turn is substantially more accurate than the high-correlation approach under the "match/mismatch" criteria.

d-N: For each threshold T, the "d-N" (for "difference in *NoDoc*") measure 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 1,475 queries. The smaller the number in "d-N" is, the better the corresponding estimation method is. 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*") measure 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*. 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 database D2 are summarized in Tables 3 and 4. The experimental results for database D3 are summarized in Tables 5 and 6. From Tables 1 to 6, 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 "mismatch" components are smaller for database D1, larger for database D2 and largest for database D3. This is probably due to the increased degrees of inhomogeneity of these three databases by their construction.

		high-correlation	our prev method	subrange method
T	U	match/mismatch	match/mismatch	match/mismatch
0.1	2506	779/102	1299/148	2352/215
0.2	1110	30/7	321/41	1002/80
0.3	500	4/2	104/14	401/28
0.4	135	1/0	27/1	97/1
0.5	54	0/0	9/1	38/1
0.6	14	0/0	4/0	8/0

Table 3: Comparison of Match/Mismatch Using D2

		high-correlation		our prev method		subrange method	
T	U	d-N	d-S	d-N	d-S	d-N	d-S
0.1	2506	26.96	0.112	20.31	0.082	12.04	0.026
0.2	1110	19.56	0.252	9.80	0.191	8.35	0.047
0.3	500	13.00	0.347	7.64	0.282	7.02	0.088
0.4	135	11.13	0.458	6.49	0.374	4.58	0.152
0.5	54	5.43	0.550	3.67	0.463	4.61	0.187
0.6	14	3.07	0.664	2.21	0.492	2.50	0.291

Table 4: Comparison of d-N and d-S Using D2

		high-correlation	our prev method	subrange method
T	U	match/mismatch	match/mismatch	match/mismatch
0.1	2582	760/135	1379/192	2410/276
0.2	1125	46/23	277/55	966/76
0.3	393	6/5	76/12	310/21
0.4	133	0/1	17/6	93/7
0.5	48	0/0	8/0	30/0
0.6	15	0/0	3/0	6/0

Table 5: Comparison of Match/Mismatch Using D3

		high-correlation		our prev method		subrange method	
T	U	d-N	d-S	d-N	d-S	d-N	d-S
0.1	2582	17.44	0.114	13.96	0.081	8.02	0.026
0.2	1125	12.47	0.245	7.16	0.198	5.72	0.054
0.3	393	10.92	0.354	6.76	0.297	5.55	0.095
0.4	133	7.18	0.460	4.89	0.405	3.85	0.158
0.5	48	3.77	0.558	2.81	0.472	2.50	0.226
0.6	15	2.20	0.659	3.20	0.534	1.80	0.409

Table 6: Comparison of d-N and d-S Using D3

Using Quadruplets and Approximate Numbers

In Section 3.2, 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. Essentially no difference in performance (compare Table 7 with Tables 1 and 2). Similar result can be obtained for databases D2 and D3 (see Tables 8 and 9).

T	m/mis	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 7: Using One Byte for Each Number for D1

T	m/mis	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 8: Using One Byte for Each Number for D2

T	m/mis	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	m/mis	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 10: Result for D1 When Maximum Weights Are Estimated

Using Triplets and Original Numbers

In Section 3.1, 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 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 10 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 Tables 1 and 2 with those in Table 10, it is clear that the use of maximum normalized weights can indeed improve the estimation accuracy substantially. Similar conclusion can be reached for databases D2 (see Table 11) and D3 (see Table 12).

T	m/mis	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 11: Result for D2 When Maximum Weights Are Estimated

T	m/mis	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 12: Result for D3 When Maximum Weights Are Estimated

5 Conclusions

In this paper, we introduced a search engine usefulness measure that is intuitive and easily understandable. 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 methods which rank search engines independent of 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.

3. Experimental results indicate that our estimation method 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 perform extensive experiments involving much larger and much more databases.

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 [4].

References

[1] G. Abdulla, B. Liu, R. Saad, and E. Fox. *Characterizing World Wide Web Queries.* TR-97-04, Virginia Tech., 1997.
[2] C. Baumgarten. *A Probabilistic Model for Distributed Information Retrieval.* ACM SIGIR, 1997.
[3] J. Callan, Z. Lu, and W. Bruce Croft. *Searching Distributed Collections with Inference Networks.* ACM SIGIR, 1995.
[4] L. Gravano, and H. Garcia-Molina. *Generalizing GlOSS to Vector-Space databases and Broker Hierarchies.* VLDB Conf., 1995.
[5] L. Gravano, and H. Garcia-Molina. *Generalizing GlOSS to Vector-Space databases and Broker Hierarchies.* Technical Report, CS Dept., Stanford U., 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 [4].)
[6] D. Harman. *Overview of the First Text Retrieval Conference.* Edited by D. Harman, Computer Systems Technology, NIST, 1993.
[7] A. Howe, and D. Dreilinger. *SavvySearch: A Meta-Search Engine that Learns Which Search Engines to Query.* AI Magazine, 18(2), 1997.
[8] IDM: Research Agenda for the 21st Century. IDM Program, NSF, March 1998.
[9] B. Jansen, A. Spink, J. Bateman, and T. Saracevic. *Real Life Information Retrieval: A Study of User Queries on the Web.* ACM SIGIR Forum, 32:1, 1998.
[10] B. Kahle, and A. Medlar. *An Information System for Corporate Users: Wide Area information Servers.* Technical Report TMC199, Thinking Machine Corp., April 1991.
[11] M. Koster. *ALIWEB: Archie-Like Indexing in the Web.* Comp. Networks and ISDN Syst., 27:2, 1994.
[12] G. Kowalski. *Information Retrieval Systems, Theory and Implementation.* Kluwer Academic Publishers, 1997.
[13] S. Lawrence, and C. Lee Giles. *Searching the World Wide Web.* Science, 280, April 1998.
[14] K. Lam, and C. Yu. *A Clustered Search Algorithm Incorporating Arbitrary Term Dependencies.* ACM TODS, September 1982.
[15] W. Meng, K-L. Liu, C. Yu, X. Wang, Y. Chang, and N. Rishe. *Determining Text Databases to Search in the Internet.* VLDB Conf., 1998.
[16] A. Singhal, C. Buckley. and M. Mitra. *Pivoted Document Length Normalization.* ACM SIGIR, 1996.
[17] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* New York: McCraw-Hill, 1983.
[18] 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.
[19] C. Yu, and W. Meng. *Principles of Database Query Processing for Advanced Applications.* Morgan Kaufmann, San Francisco, 1998.
[20] B. Yuwono, and D. Lee. *Server Ranking for Distributed Text Resource Systems on the Internet.* 5th Int'l Conf. on Database Systems for Adv. Appli. (DASFAA'97), April 1997.