Discovering the Representative of a Search Engine

King-Lup Liu[1] Clement Yu[2] Weiyi Meng[3] Adrian Santoso[1]
Chonghua Zhang[1]
[1] School of CTI, DePaul University
Chicago, IL 60604
Tel : (312)362-5089
FAX : (312)362-6116
kliu@cs.depaul.edu
[2] Dept. of EECS, University of Illinois at Chicago
Chicago, IL 60607
[3] Dept. of Computer Science, SUNY – Binghamton
Binghamton, NY 13902

Paper number: 397
Correspondence author: King-Lup Liu

Abstract

Given a large number of search engines on the Internet, it is difficult for a person to determine which search engines could serve his/her information needs. A common solution is to construct a metasearch engine on top of the search engines. Upon receiving a user query, the metasearch engine sends it to those underlying search engines which are likely to return the desired documents for the query. The selection algorithm used by a metasearch engine to determine whether a search engine should be sent the query typically makes the decision based on the search-engine *representative*, which contains characteristic information about the database of a search engine. However, an underlying search engine may not be willing to provide the needed information to the metasearch engine. This paper shows that the needed information can be estimated from an uncooperative search engine with good accuracy. Two pieces of information which permit accurate search engine selection are the number of documents indexed by the search engine and the maximum weight of each term. In this paper, we present techniques for the estimation of these two pieces of information.

1 Introduction

The Internet has become a vast information resource in recent years. To help ordinary users find desired data in this environment, many *search engines* have been created. Each search engine has a *text database* that is defined by the set of documents that can be searched by the search engine. Usually, an inverted file index for all documents in the database is created and stored in the search engine. For each *term* which can represent a significant word or a combination of several (usually adjacent) significant words, this index can identify the documents that contain the term quickly.

Frequently, the information needed by a user is stored in the databases of multiple search engines. As an example, consider the case when a user wants to find research papers in some subject area. It is likely that the desired papers are scattered in a number of publishers' and/or universities' databases. Substantial effort would be needed for the user to search each database and identify useful papers from the retrieved papers. A solution to this problem is to implement a *metasearch engine* on top of many local search engines. A metasearch engine is just an interface. It does not maintain its own index on documents. However, a sophisticated metasearch engine may maintain information about the contents of its underlying search engines 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 its local search engines. With such a metasearch engine, only one query is needed from the above user to invoke multiple search engines.

If the number of local search engines in a metasearch engine is large, then it is likely that for a given query, only a small percentage of all search engines may contain sufficiently useful documents to the query. In order to avoid or reduce the possibility of invoking useless search engines for a query, we should first identify those search engines that are most likely to provide useful results to the query and then pass the query to only the identified search engines. Examples of systems that employ this approach include WAIS [8], ALIWEB [9], CORI net [3], D-WISE [21], gGlOSS [6], ProFusion [4] and SavvySearch [7]. The problem of identifying potentially useful databases to search is known as the *database selection problem* and has received much research interest [3, 5, 6, 12, 25].

Typically, the metasearch engine keeps some characteristic information of each underlying search engine. We call such characteristic information of a search engine the *representative*. For a given query, the metasearch engine determines whether an underlying local search engine should be searched or not from the representative of the local search engine. Various approaches to the database selection problem differ in the kind of information kept in the representative of a search engine[16]. The more accurate approaches are those which keep detailed statistical information about the contents of each local search engine. In gGlOSS [6], for example, for each term t in the vocabulary of a local search engine a pair (df_t, W_t) is kept, where df_t is the number of documents having term t and W_t is the sum of weights of term t over all documents in the database.

To find an effective and efficient method to select text databases to search has been one of our research goals. We provided several solutions to this problem [12, 14, 15, 22, 24, 25].

Our experimental results showed that our solutions achieved near optimal results. As with most approaches to the database selection problem, we implicitly assume that the underlying local search engines are cooperative and are willing to provide the information needed by the metasearch engine. Under this assumption, for each local search engine, the representative acquired by the metasearch engine would faithfully reflect its contents. However, in the Internet environment, each search engine is usually autonomous and managed with its own interest in mind. The contents of each search engine may be viewed as proprietary. Thus, a local search engine may not be willing to provide all the information requested by the metasearch engine. It may even provide information that leads to an incorrect/inaccurate representation of its contents. Such scenario is also described in [2].

In this paper, we show how the information needed to construct a search-engine representative may be obtained or estimated. Our method is based on sampling documents from a local search engine. No special cooperation is needed from the local search engine. Similar to the approach in [2], we submit queries to a search engine to obtain samples of documents and draw statistical information concerning the search engine based on the samples.

Note that the information contents of a search-engine representative depends to a large extent on the database selection algorithm employed and can be very detailed. In this paper, we estimate the following quantities:

- *the number of documents indexed by a search engine*
  It was pointed out in [2] that this estimation is an open problem. Our technique can be modified to estimate the document frequency of a term.

- *the maximum weight of each term in the vocabulary of a search engine*
  In our earlier work [12, 15, 22, 24, 25], we showed that the use of the maximum weights of terms permited optimal retrieval results for single-term queries and near optimal results for multiple-term queries.

The quantities which we estimate in this paper are critical pieces of information in search engine selection but are absent from [2].

We estimate the number of documents indexed by a search engine from a random sample of its documents. Three text databases were formed using the text documents from the TREC collection. Our technique was applied to estimate their sizes. In each case, an estimation accuracy of less than 2.5% estimation error of the database size was achieved by sampling no more than 2.5% of the documents in the database.

The *global* weight of a term in a document is the weight of a term in a document determined by a metasearch engine. It is usually different from the weight of the term in the same document determined by an underlying search engine. The maximum value of the global term weights of a query term in the documents indexed by a search engine is a critical piece of information in our database selection algorithm. We estimate its value by sampling the top 30 or 50 documents retrieved when the term is submitted to the search engine as a single-term query. Our estimate is considered to be sufficiently accurate if the ratio of the estimate to the actual maximum value is at least 0.99. Our experimental results show that on average, the number of terms (out of 100) for which sufficiently accurate estimates have been obtained for their maximum global weights was (a) 81.47 when the top 30 documents were sampled, and (b) 84.56 when the top 50 documents were sampled. The corresponding average figures were from 92.95% to 95.5% and 94.7% to 97.5% respectively when the term-weighting formulas used by a metasearch engine and a search engine share similar characteristics.

The rest of this paper is organized as follows. In section 2, we give a brief description of related work. Section 3 discusses our sampling technique to estimate the number of documents indexed by a search engine and the factors that affect the estimation accuracy. We give the experimental results for three text databases. Section 4 describes how we estimate the maximum value of the global weights of a term in all the documents of a local search engine and explains the rationale behind our technique. Experimental results are also provided. We summarize and conclude in section 5.

2 Related Work

The representative of a search engine (or text database) is critical in the solution of the database selection problem. It keeps information for each term in the vocabulary of the search engine (or text database). As discussed in the previous section, the search engine (database) may be unwilling to provide all the information necessary for the construction of its representative. For such an uncooperative search engine, methods are needed to extract the required information without any special cooperation from the search engine. This is a relatively new research area. In [2], a query-based sampling method is provided for the automatic discovery of language models for text databases. It is indicated that in general, a language model consists of the indexing terms that occur in the database and their frequencies of occurrence. However, their method only aims at discovering the set of indexing terms of a text databases. The quantities we estimate in this paper are absent from those estimated in [2]. A statistical technique to estimate the size of the indexable Web is given in [10]. However, this technique requires that the number of pages indexed by a search engine be known.

There are various statistical techniques to estimate the order statistics of a dataset. In a recent paper [13], for instance, random sampling techniques were provided to estimate the order statistics (the k-th largest value, $k = 1, 2, \ldots$) of a large dataset when the size of the dataset is not known in advance. The accuracy of an estimate of an order statistic, as is the case with most techniques, depends on the size of sample. To obtain a sufficiently accurate estimate, a large sample may need to be drawn. As the weights of a term are not known and we need to download the documents containing the term to compute them, a larger sample incurs substantial processing and computational overhead. The number of terms indexed by a search engine is typically in the order of millions. Thus, these techniques are computationally prohibitive and hence, not appropriate for our purposes.

3 Estimating the Number of Documents indexed by a Search Engine

Our search-engine selection method, as well as various other search-engine (database) selection methods, needs to know the number of documents indexed by each underlying search engine. Whether the number of documents indexed by a search engine can be estimated by sampling its documents is an open problem for some time [2]. In this section, we show that the number of documents indexed by a search engine can be estimated with good accuracy by sampling.

Let N be the number of all documents indexed by a search engine. We first draw a random sample of m documents from the search engine. Suppose we choose a document randomly from the collection of all documents indexed by the search engine. Then, the probability that this document is from the sample of m documents is $\frac{m}{N}$. Now, we perform n times the process of randomly selecting a document and observing whether it is a document from the earlier sample. Suppose Y of these n randomly selected documents are from the sample of m documents chosen earlier. From probability theory, $E(Y)$, the expected value of Y, is $n \times \frac{m}{N}$. By taking the observed value Y_o of Y in the sample as an approximation to the expected value $E(Y)$, i.e., $Y_o \approx n \times \frac{m}{N}$, we then obtain an estimated value $n \times \frac{m}{Y_o}$ of N, the number of documents indexed by the search engine.

Intuitively, if we examine a large number of randomly selected documents (i.e., n is large), then among these n documents, the proportion of documents from the earlier sample of m document will be close to the ratio of number of documents in the sample to the number of all documents indexed by the search engine, (i.e., $\frac{Y_o}{n}$ will be close to $\frac{m}{N}$). That is, $n \times \frac{m}{Y_o}$ will be close to N. (This intuition is justified by the probability law known as the weak law of large numbers [17]). Note that if the number of documents in the sample (i.e., m) is high, our accuracy will also be high. In the extreme case, if the sample includes every document, i.e., $m = N$, then Y_o will be the same as n, the number of individually randomly selected documents, and our estimated value will be exact. In the following, we give a more formal justification of our intuition.

In general, it is unlikely that the observed value of Y_o will be exactly the same as its expected value $E(Y) = n \times \frac{m}{N}$. Let δY be its error. That is, $Y_o = E(Y) + \delta Y$. Then, the error in our estimation is $N - n \times \frac{m}{E(Y)+\delta Y}$, which is the same as $N \times \frac{\delta Y}{E(Y)+\delta Y}$ or $N^2 \times \frac{\delta Y}{m \times n + \delta Y \times N}$. From the last expression, it is clear that for a fixed error in the observation, δY, the error in the estimated value of N , decreases (i.e., the estimated value is more accurate) if we increase the sample size and/or examine a large number of randomly selected documents.

For the documents indexed by a search engine, the only way we can access them are through queries submitted to the search engine. Thus, to obtain a sample of m randomly chosen documents, our approach is this. We first arbitrarily form a query and submit the query to a search engine. Typically, the search engine will return a list of titles and URLs of the documents containing term t. Let df be the number of documents in the query result. Using a random number generator, we generate an integer in the range from 1 to df. If the number generated is k, then we retrieve the k-th document in the query result list and include it in the sample. We repeat this process of document selection until we have retrieved some fraction f of the documents in the query result or m distinct documents have been included in the sample. If we have not sampled enough documents, we then arbitrarily pick another term and do the above process again. Note that if query result list is very long and our document selection process will not be efficient. In that case, instead of sampling from all the documents in the query result, we may sample from a subset of these documents, say the top 1000 documents. To randomly select a document, we use a similar trick. That is, arbitrarily form a query, submit the query to the search engine and randomly pick a document from the result list. Note that this randomly picked document may have been picked previously. However, for the collection of documents sampled earlier, we require all m documents to be distinct.

To test our techniques, we formed three text document databases of different sizes and applied our techniques to estimate their sizes. The documents are from the TREC collection. The first database has 100,000 documents, the second 200,000 and the third 300,000.

For each text database, we drew several samples of varying sizes. For each sample, we performed the experiment a number of times. Each time a different number of documents was examined. The estimation results of the sizes of the three databases are shown in tables 1 to 3. The last row of each table gives the number of documents randomly picked for examination (i.e., value of n). The entries of the first column, except the last entry, specify the sample sizes. The entries of the other columns, except those in the first and the last rows, are the percentage errors of our estimated values of the database size. Consider, for example, the percentage error 1.6% in the entry in the third row and fourth column of table 1. The first entry of the third row is 2000 and the last entry of the fourth column is 3000. This means that for the experiment that resulted in this estimated error of 1.6% of the database size, the number of documents in the sample was 2000, the number of documents randomly picked for inspection was 3000.

As expected, the percentage error generally decreases as the sample size increases and also as the number of documents examined increases. Judging from the results of tables 1 to 3, our technique works reasonably well. For all three text databases, we see that we can obtain a percentage error of less than 2.5% of the database size by checking a total of no more than 2.5% of all the documents.

Sample size m	% error	% error	% error	% error
1000	3.0%	2.5%	2.2%	1.6%
2000	2.5%	2.2%	1.6%	1.2 %
3000	2.3%	1.7%	1.3%	0.8%
4000	1.7%	1.4%	0.9%	0.5%
No. of docs examined n	1000	2000	3000	4000

Database size = 100,000

Table 1: Estimation Results for the first text database

Sample size m	% error	% error	% error	% error
1000	3.8%	3.4%	2.9%	2.5%
2000	3.5%	2.9%	2.4%	2.2%
3000	3.0%	2.5%	2.1%	1.7%
4000	2.5%	2.2%	1.6%	1.2%
No. of docs examined n	1000	2000	3000	4000

Database size = 200,000

Table 2: Estimation Results for the second text database

Sample size m	% error	% error	% error	% error
1000	4.8 %	4.3%	3.7%	3.3%
2000	4.4%	3.7%	3.4%	2.8%
3000	3.9%	3.3%	2.9%	2.4%
4000	3.4%	3.0%	2.5%	2.0%
No. of docs examined n	1000	2000	3000	4000

Database size = 300,000

Table 3: Estimation Results for third text database

We can easily modify our technique to estimate the document frequency of a term. The document frequency of a term is the number of documents indexed by the search engine having the term. We simply approximate the proportion of documents having a term by the proportion of documents in the sample having the term.

4 Estimation of Maximum Global Term Weight in a Local Search Engine

The weight of a term in a document is a measure of the significance of the term in representing the document. Each search engine determines the weight of a term using its own term-weighting formula. The weight of a term in a document computed using the term-weighting formula of a local search engine shall be referred to as the *local weight* of the term and that determined by the term-weighting formula of the metasearch engine as the *global weight* of the term.

When a query is submitted to a metasearch engine, in determining whether a local search engine should be searched or not, we have shown that the maximum values of the global weights of the query terms in all the documents indexed by the local search engine are critical information [12, 15, 22]. We shall refer to the maximum value of the global weights of a term in all the documents indexed by a local search engine as the *maximum global weight* of the term in the local search engine. Note that a local search engine is usually autonomous. It may not be willing to expend extra resources to compute the maximum global weight of each term in the local search engine using the term-weighting formula of the metasearch engine. In this secton, we discuss how the maximum global weight of a term in a local search engine may be estimated.

The term-weighting formula used by a local search engine is likely to be different from that employed by the metasearch engine. As a result, for a term in a document, its local weight in the document is likely to be different from its global weight in the document. The term "information", for example, may have a weight of 0.16 in a document according to the metasearch engine but have a weight of 0.35 in the same document according to a local search engine. Even if a local search engine and the metasearch engine use the same term-weighting formula, the local weight of a term in a document may still be different from the global weight of the term in the same document. This is because the weight of a term may be affected by the document frequency of the term (i.e., the number of documents having the term). As the metasearch engine conceptually contains all the documents of the underlying local search engines, the collection of documents having a term in the (conceptual) text database of the metasearch engine is likely to be larger than that in the text database of any of its underlying local search engines. That is, the global document frequency of a term is likely to be different from the document frequency of the term in a local search engine. Thus, the local weight and global weight of a term in a document may not be the same even though the local and global term-weighting formulas are identical.

When we submit a query to a local search engine, local weights of the query terms are used to determine the similarities of the documents indexed by the search engine to the query. For a single-term query containing the term t, as the same query term weight is used to compute the similarity of each document, the rank order of the retreived documents determined by the document similarities to the query will be the same as that determined by the local weights of the query term t in the documents. Then, the first retrieved document in the query result will have the maximum local weight for the term t. However, from the discussion given above, for a local search engine, a document with the maximum local weight for a term may not be the document with the maximum global weight of the term in the local search engine.

The problem of finding the maximum value of a dataset is trivial if the values of the dataset are known. A single pass of the values is sufficient to find accurately the maximum value of the dataset. However, the global weights of a term t in the documents indexed by a search engine are not known. To determine accurately the maximum global weight of a term t in a search engine, we need to download all the documents containing the term t and compute the global weight of term t for each downloaded document. As there are many such documents and there are many terms in the vocabulary of the search engine, repeating this process for each term is computationally not feasible. Note that the maximum global weights are used for estimating the usefulness of a search engine. Thus, for our purposes, 100% accuracy for the maximum global weights is preferable but not critical. It is sufficient that for a term t, we have a sufficiently accurate estimate of the maximum global weight of the term t (say, its ratio to the actual maximum global weight of term t in the local search engine is at least 0.99). In the following, we present a method for finding an accurate estimate of the maximum global weight of each term t. The method is based on the observation that for many combinations of global term-weighting formula and local term-weighting formula, documents that have relatively large local weights for a term tend to have relatively large global weights for the term. (That is, if a term in document d1 has higher local weight than in document d2, then the same term is also likely to have a higher global weight in document d1 than that in document d2.)

4.1 Sampling the Top Local Term Weights

As discussed above, the weight of a term in a document determined by the metasearch engine is usually different from that determined by an underlying local search engine. In the text database of a local search engine, the document with the maximum global weight for a term may not be the same document with the maximum local weight for the term. However, the document with the maximum global weight of a term is often (though not necessarily) found among the set of documents with the highest local weights for the term. The size in this set of documents depends on the combination of global and local term-weighting formulas. In our experiments, we found that in many cases, this number is no more than 50. To understand this phenomenon, below we give a discussion of the formulas generally used to determine the weight of a term.

4.1.1 Weight of a Term in a Document

Although the weight of a term can be computed in many different ways, it is generally influenced by the following components: (a) its term frequency in the document, (b) the inverse document frequency weight and (c) the document normalization [19].

Term frequency The term frequency, tf, of a term in a document is the number of occurrences of the term in the document. Evidently, the higher the term frequency of a term is in a document, the more significant the term is in the document. (Note that stop words (words that occur frequently in a large number of documents, such as "is", "the" and etc.) are usually removed.) However, the significance of a term that occurs, say, ten times in a document may not be considered to be ten times more significant than a term that occurs only once in the document. To reduce the impact of terms with high occurrences, the term frequency component in the weight of a term is often computed as some function of the term frequency. We shall call this term frequency component the **term frequency weight** of the term.

Inverse document frequency The document frequency of a term in a search engine is the number of documents (web pages) indexed by the search engine having the term. Usually, the more documents having a term t, the less useful the term t is in discriminating those documents having t from those not having t. The *inverse document frequency* (*idf*) method assigns a weight, called the **inverse document frequency weight**, to a term t that is usually defined as $\log \frac{N}{df_t}$, where N is the number of documents indexed by the search engine and df_t is the number of documents indexed by the search engine having term t. Variations of this *idf* definition are also common.

Document normalization A longer document tends to have more terms and/or terms with higher term frequencies. This means that document retrieval would be biased in favor of long documents. To compensate for the document length, term weights are frequently normalized or divided by a factor involving the document length. Some common document normalization factors include the common Euclidean norm, p-norm, the total number of terms in the document, the maximum term frequency in the document or some combinations of these factors.

A search engine can choose any expression to determine the significance of each of the above components in the computation of term weights. From the above discussion, in general, an expression for the term-frequency weight of a term in a document increases with the term frequency of the term in the document, an expression for the inverse document frequency weight (*idf*) of a term decreases with the document frequency of the term and a document normalization expression increases with the document length. Thus, a term with the highest weight in a document computed using one term-weighting formula also tends to have high weight in the same document computed using a different term-weighting formula. This is the intuition behind why for a term t, we can often obtain the document with the maximum global term weight from the sample of documents with the highest local weights for the term t.

Each combination of the expressions of the above components determines a different term-weighting formula. Potentially, there are infinitely many term-weighting formulas. In our experiments, we used expressions that are commonly encountered in the literature or are being used by existing IR systems.

4.1.2 Our Experiments

We formed four text databases for four hypothetical local search engines. The documents are from the TREC collection. The total number of documents in these text databases is 153,425. That is, the hypothetical metasearch engine has 153,425 documents. The local text database we chose for our experiments has 45,815 documents. The global document frequencies of terms are determined from all the documents in the four text databases.

There are two common term-weighting schemes that make use of the inverse document frequency weight (*idf*):

Scheme (a) The *idf* weight is incorporated into the **query term weight** (and not the document term weight). The weight of a term in a document is determined as its term frequency weight divided by a document normalization factor.

Scheme (b) The *idf* weight is incorporated into the **document term weight**. The weight of a term in a document is computed as the product of its term frequency weight and the *idf* weight divided by a document normalization factor [19].

Thus, concerning the term-weighting schemes of the metasearch engine and a local search engine, there are four possible combinations, as each may use either scheme (a) or scheme (b). We performed the experiments for all four possible combinations. However, we reported only the results for the case when both the metasearch engine and the local search engine use scheme (a) (because of space limitation). Note that in practice, scheme (a) is more commonly used. This is because for scheme (b), the *idf* weight is incorporated into the document term weight and, as a result, the weight of a term may need to be updated when the documents indexed by a search engine changes.

We formed ten different term-weighting formulas. Thus, the number of formula combinations is 100. The formulas are shown in table 4. The following notations are used in the table.

- tf_t : the term frequency of a term t in a document
- avg_tf : the average of the term frequencies of all terms in a document
- max_tf : the maximum of the term frequencies of all terms in a document
- tot_tms : total number of terms
- $unique\_tms$: number of unique terms in a document
- w_j : weight of the j-th term in a document before normalization
- $Euclidean\_norm$: defined as $\sqrt{w_1^2 + w_2^2 + \ldots + w_j^2 + \ldots}$

In table 4, under the column *tf* **weight** are the term-frequency weight formulas and under the column *norm* are the formulas for the document normalization.

	tf **weight**	*norm*
f_1	$\frac{1+\log tf}{1+\log avg\_tf}$ [20]	$10 + unique\_tms$ [20]
f_2	$1 + \frac{tf}{max\_tf}$ [18]	$3 + \left(\sum w_j^5\right)^{\frac{1}{5}}$
f_3	$1 + \frac{tf}{max\_tf}$	$2 + Euclidean\_norm$
f_4	$1 + \frac{100 \times tf_t}{tf_t + tot_t ms}$ [20]	$Euclidean\_norm$ [18]
f_5	$tf_t^{0.3}$	$0.3 \times unique\_tms + 0.7 \times Euclidean\_norm$ [20]
f_6	$tf_t^{0.2}$	$\left(\sum w_j^7\right)^{\frac{1}{7}}$
f_7	tf_t	$5 + \left(\sum w_j^3\right)^{\frac{1}{3}}$
f_8	tf_t	$max\_tf + 0.6 \times unique\_tms$ [20]
f_9	tf_t	$0.5 + 0.5 \times Euclidean\_norm$
f_{10}	tf_t	$Euclidean\_norm$

Table 4: Term-weighting formulas

In our experiments, the ten term-weighting formulas are formed from the formulas for the term-frequency weight and the document normalization in the **rows** f_1 **to** f_{10}. For row f_2, for example, the term-weighting formula is $\left(1 + \frac{tf}{max\_tf}\right) / \left(3 + \left(\sum w_j^5\right)^{\frac{1}{5}}\right)$.

For each combination of local term-weighting formula and global term-weighting formula, we perform the following steps:

1. chose randomly 100 terms in the local database;
2. for each term chosen in step 1,

(a) computed the local weight and global weight of the term in each document in the local database having the term; and determined the actual maximum global weight of the term in the local database;

(b) obtained 50 documents with highest local weights of the term;

(c) determined the maximum value of the global weights of the term in (i) the 30 documents with highest local term weights and (ii) the 50 documents with highest local term weights;

(d) for the maximum value of the global term weight obtained in each of the two cases (i) and (ii) of the previous step, computed the ratio of the maximum value to the actual maximum global weight of the term in the local database (obtained in step 2.(b)); and if the computed ratio is at least 0.99, we recorded that a sufficiently accurate estimate has been obtained.

As terms with low document frequencies would favor the experimental results, the terms obtained in step 1 are all randomly chosen from terms with document frequencies higher than 2000. The average document frequency of these 100 terms is 3035. An estimate of the maximum global weight of a term in the local database is obtained by examining (a) 30 documents and (b) 50 documents with the highest local term weights (step 2.(b)). The estimate is considered to be *sufficiently accurate* if *the ratio of the estimate to the actual maximum global weight of the term in the local database is at least 0.99.* For each combination of local and global term-weighting formulas, we recorded the *number N_{SAE} of terms for each of which a sufficiently accurate estimate of the actual maximum global term weights in the local database has been obtained.* A local and global term-weighting formula combination will be considered to be

- *perfect* if this number N_{SAE} is 100 (since we then have obtained sufficiently accurate estimates for all 100 terms);
- *very good* if $96 \leq N_{SAE} \leq 99$;
- *good* if $91 \leq N_{SAE} \leq 95$;
- *satisfactory* if $86 \leq N_{SAE} \leq 90$;
- *fair* if $81 \leq N_{SAE} \leq 85$; and
- *unsatisfactory* if $N_{SAE} \leq 81$.

The local and global term-weighting formula combinations were partitioned into six groups according to whether they are perfect, very good, good, satisfactory, fair or unsatisfactory. For each group, we determined the number of formula combinations in the group and its percentage of all formula combinations.

The experimental results are given in table 5. The first column of the table shows the number of documents with highest local weights that were examined. The last column with the heading "Average estimation" shows the average number of terms (out of a total of 100 terms) for which sufficiently accurate estimates of their maximum global weights in the local database have been obtained for a local and global term-weighting formula combination. The second entry 81.47 of the last column, for example, means that when the top 30 local documents were examined, for a local and global formula combination, on average, sufficiently accurate estimates were obtained for 81.47 of the 100 terms. A percentage value in the columns 2 to 7 is the percentage of combinations of local and global term-weighting formulas in the group specified by the first entry of the column containing the percentage value. In table 5, the entry of the second row under column "good", for example, has a value of 8.89%. It shows that if we examined only the top 30 local documents, of all combinations of local and global term-weighting formulas 8.89% were found to be good. (That is, for each of these combinations, the number of terms for which sufficiently accurate estimates of their maximum global weights in the local database have been obtained by examining the top 30 documents is between 91 and 95).

Note that there are 10 formula combinations in which both local and global document term-weighting formulas are identical. Since both local and global document term-weighting formulas do not incorporate the *idf* weight, for each of these 10 formulas, for a term in a document, its local term weight is exactly the same as its global term weight. That is, its maximum local weight will also be the maximum global weight of the term in the local database. Thus, we do not include these 10 formula combinations in our calculation for the results of table 5.

Documents examined	(below 81) unsatisfactory	(81 to 85) fair	(86 to 90) satisfactory	(91 to 95) good	(96 to 99) very good	(100) perfect	Average estimation
Top 30	26.67%	6.67%	6.67%	8.89%	28.89%	22.22%	81.47
Top 50	22.22%	3.33%	7.78%	8.89%	31.11%	26.67%	84.56

Number of formula combinations = 90

Table 5: Maximum global term weight estimation

Last column of table 5 shows that for a local and global formula combination, on average, sufficiently accurate estimates of the maximum global weights in the local database were obtained for (a) 81.47 terms when the top 30 local documents were sampled, and (b) 84.56 terms when the top 50 local documents were sampled. That is, we get decent but not spectacular results. If we examine closely the collection of formulas of the term-weighting components of table 4, we found that for two given term-weighting formulas, the formulas of the term-frequency weight component are important in determining whether the results are good or not (i.e., whether the maximum global term weight in a local database can be determined by examining a small number of local documents with the highest local weights of the term). The formulas for the document normalization do have some effects but not as critical in comparison. The term-frequency weight formulas used in our experiments can be broadly divided into three categories: (a) the formulas in rows f_1 to f_4 of table 4, each being transformed in some way that involves the term frequencies of a document, (b) the formulas in rows f_5 and f_6, each having an exponent (other than 1), and (c) the formulas in rows f_7 to f_{10}, each having the same formula. Our experiments showed that good results were obtained if the local and global term-weighting formulas were both from (i) the group of formulas formed from the formulas of term-frequency weight and document normalization of rows f_1 to f_5, or (ii) the group of formulas formed from those formulas of rows f_6 to f_{10}. The results are shown in tables 6 and 7. Despite the term frequency weight formulas of rows f_5 and f_6 are of the same form, the term-weighting formulas formed from the component formulas of these two rows belong to different groups. This may be due to the fact that the two term-weighting formulas use different formulas of document normalization.

Documents examined	(below 81) unsatisfactory	(81 to 85) fair	(86 to 90) satisfactory	(91 to 95) good	(96 to 99) very good	(100) perfect	Average estimation
Top 30	0%	15%	0%	5%	45%	35%	95.55
Top 50	0%	0%	10%	5%	40%	45%	97.5

Local and global term-weighting formulas: rows f_1 to f_5 of table 4
Total number of combinations = 20

Table 6: Maximum global term weight estimation

Documents examined	(below 81) unsatisfactory	(81 to 85) fair	(86 to 90) satisfactory	(91 to 95) good	(96 to 99) very good	(100) perfect	Average estimation
Top 30	10%	0%	15%	5%	40%	30%	92.95
Top 50	5%	5%	0%	15%	40%	35%	94.7

Local and global term-weighting formulas: rows f_6 to f_{10} of table 4
Total number of formula combinations = 20

Table 7: Maximum global term weight estimation

Our technique of sampling the top local documents generally give good results when the local and global term-weighting formulas exhibit similar characteristics. Thus, to gauge whether our technique would work well for a particular local search engine or not, one method is to find out the term-weighting formula of the local search engine. In our earlier work [11], we have shown how to estimate the term-weighting formula from an uncooperative local search engine.

5 Conclusions

The database selection algorithm is an important component of a metasearch engine, especially when the metasearch engine has a large number of underlying search engines. For a given query, it determines whether an underlying local search engine should be sent the query or not. Typically, the decision is made based on the search engine representative, which contains characteristic information about the database of the search engine. Two pieces of information which permit accurate search engine selection are the number of documents indexed by the search engine and the maximum weight of each term. In this paper, we presented techniques for the estimation of these two pieces of information.

In [2], it was pointed out that the estimation of the number of documents indexed by a search engine is an open problem. We developed a technique that makes this estimation possible. The number of documents indexed by a search engine is estimated by drawing a random sample of its documents. Three text databases were formed using the text documents from the TREC collection. Our technique was applied to estimate their sizes. In each case, an estimation accuracy of less than 2.5% estimation error of the database size was achieved by sampling no more than 2.5% of the documents in the database.

The *global* weight of a term in a document is the weight of a term in a document determined by a metasearch engine and is usually different from the weight of the term in the same document determined by a local search engine. We are interested in estimating the maximum value of the global weights of a term in all documents indexed by a search engine. Our approach is to sample the top 30 or 50 documents retrieved when the term is submitted to the search engine as a single-term query. Our estimate is considered to be sufficiently accurate if the ratio of the estimate to the actual maximum value is at least 0.99. Our experimental results show that on average, the number of terms (out of 100) for which sufficiently accurate estimates have been obtained for their maximum global weights was (a) 81.47 when the top 30 documents were sampled, and (b) 84.56 when the top 50 documents were sampled. The corresponding average figures were from 92.95% to 95.5% and 94.7% to 97.5% respectively when the term-weighting formulas used by a metasearch engine and a search engine share similar characteristics.

References

[1] J. Broglio, J.P. Callan, W.B. Croft and D.W. Nachbar. *Document retrieval and routing using the INQUERY system.* Proceedings of the Third Text REtrieval Conference (TREC-3), pp.29-38. NIST Special Publication 500-225, April 1994.

[2] J. Callan, M. Connell and A. Du. *Automatic Discovery of Language Models for Text Databases.* ACM SIGMOD, 1999, pp. 479-490.

[3] J. Callan, Z. Lu, and. W. Croft. *Searching Distributed Collections with Inference Networks.* ACM SIGIR, 1995, pp. 21-28.

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

[5] J. French, A. Powell, J. Callan, C. Viles, T. Emmitt, K. Prey and Y. Mou. *Comparing the Performance of Database Selection Algorithms.* ACM SIGIR, 1999.

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

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

[9] 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).

[10] S. Lawrence and C.L. Giles. *Searching the World Wide Web.* Science, Vol. 280, Number 5360, pp.98-100, 1998.

[11] K.L. Liu, W. Meng, C. Yu, C. Zhang and N. Rishe. *Discovery of Similarity Computations of Search Engines.* ACM CIKM'00, November, 2000.

[12] K.L. Liu, C. Yu, W. Meng, W. Wu, and N. Rishe. *A Statistical Method for Estimating the Usefulness of Text Databases.* IEEE Transactions on Knowledge and Data Engineering (to appear).

[13] G. S. Manku, S. Rajagopalan and B. G. Lindsay. *Random Sampling Techniques for Space Efficient Online Computation of Order Statistics of Large Datasets.* ACM SIGMOD, 1999.

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

[15] W. Meng, K.L. Liu, C. Yu, W. Wu, and N. Rishe. *Estimating the Usefulness of Search Engines.* 15th International Conference on Data Engineering, March 1999.

[16] W. Meng, C. Yu and K.L. Liu. *Building Efficient and Effective Metasearch Engines.* ACM Computing Surveys (to appear).

[17] A.M. Mood, F.A. Graybill and D.C. Boes. *Introduction to the Theory of Statistics.* McGraw-Hill, 1974.

[18] G. Salton and C. Buckley. *Term-weighting approaches in automatic text retrieval.* Information Processing and Management, 24(5):513-523, 1988.

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

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

[21] 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.

[22] C. Yu, K.L. Liu, W. Wu, W. Meng and N. Rishe. *Finding the Most Similar Documents across Multiple Text Databases.* Proc. of the IEEE Conference on Advances in Digital Libraries (ADL'99), Baltimore, Maryland, May 1999.

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

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

[25] C. Yu, W. Meng, W. Wu and K.L. Liu. *Efficient and Effective Metasearch for Text Databases Incorporating Linkages among Documents.* ACM SIGMOD, 2001, pp. 187-198.