Server Selection on the World Wide Web*

Nick Craswell and Peter Bailey

Department of Computer Science
The Australian National University, Canberra Australia
{Nick.Craswell,Peter.Bailey}@cs.anu.edu.au

David Hawking

CSIRO Mathematical and Information Sciences
Canberra Australia
David.Hawking@cmis.csiro.au

ABSTRACT

We evaluate server selection methods in a Web environment, modeling a digital library which makes use of existing Web search servers rather than building its own index. The evaluation framework portrays the Web realistically in several ways. Its search servers index real Web documents, are of various sizes, cover different topic areas and employ different retrieval methods. Selection is based on statistics extracted from the results of probe queries submitted to each server. We evaluate published selection methods and a new method for enhancing selection based on expected search server effectiveness.

Results show CORI to be the most effective of three published selection methods. CORI selection steadily degrades with fewer probe queries, causing a drop in early precision of as much as 0.05 (one relevant document out of 20). Modifying CORI selection based on an estimation of expected effectiveness disappointingly yields no significant improvement in effectiveness. However, modifying CORI based on known effectiveness does yield small but significant improvements. Other results are that a very effective selection of ten servers outperforms both a selection of all servers and a centralised index covering all documents. Finally, acknowledging coverage limitations of real Web indexes we model centralised indexes of half and a quarter of the document servers, resulting in a sharp drop in effectiveness.

Keywords: Distributed information retrieval, server selection, World Wide Web, effectiveness evaluation.

* The authors wish to acknowledge that this work was carried out within the Cooperative Research Centre for Advanced Computational Systems established under the Australian Government's Cooperative Research Centres Program.

INTRODUCTION

Inclusion of material in a physical library is a process of bringing in outside information — books, periodicals, videos, CD-ROMs and so on — according to some policy. Librarians choose materials carefully, to meet people's expectations of information quality and quantity. The same quality-control, including only hand-picked information, is desirable in digital libraries. However, since Web documents can contain misleading information or be otherwise out of step with the goals of the library, and may change without notice, there is an issue of maintaining information quality if Web pages are to be included.

A simple quality control method is to include Web *sites* selectively, according to similar policies to those used in building non-digital collections. For example, a digital librarian might link to or otherwise include the Encyclopædia Britannica site (`http://www.eb.com`) in their digital library. Such a decision would depend on the site's documents and the likelihood of the document provider, in this case Encyclopædia Britannica Inc, continuing to provide such documents.

Including a chosen target site in a digital library might be as simple as providing links to the target's home page or to its search interface. A tighter integration can be achieved, with permission from the document provider, by mirroring the target documents, by assigning meta-data to target documents in the library's schema or by cross referencing the library's pages with pages in the target site.

This paper considers a different approach, where a digital library incorporates a distributed information retrieval broker. The broker is capable of selecting, querying and presenting the results of a number of chosen Web search servers. The decision on whether the broker should address a Web search server in this case is equivalent to the decision to link to a Web site: based on the continuing quality of the documents in question.

A large number of Web search servers exist, as do a few manually-generated search server listings. InvisibleWeb (`http://www.invisibleweb.com`) lists 10 000 search servers over a broad range of subject areas and CiteLine Professional (`http://www.caredata.com`) lists

2 000 in the health and medical area. Each claims its listed search servers cover documents “invisible” to or “hidden” from outside indexers — so building a centralised index of such documents would be impossible. InvisibleWeb claims their list contains only a fraction of the true number of Web search servers. Considering these *manual* selection efforts already cover over 10 000 Web search servers, performing automatic selection over 956 servers as examined later in this paper does not seem unreasonable.

Brokers which concurrently query multiple servers and merge their results already exist on the Web. Examples include Inquirus [15], MetaCrawler [18], SavvySearch [6] and ProFusion [7]. For this reason, this paper concentrates on the problem of server selection in such an environment, assuming retrieval and merging are carried out using methods already implemented in Inquirus.

This paper has three goals. The first is to evaluate the CORI [1], vGlOSS [11] and CVV [23] selection methods based on probe queries (probe queries are described below) in an environment of heterogeneous Web search servers. The second is to test the hypothesis that server effectiveness is important in such an environment, by introducing a method for estimating differences in server effectiveness and modifying selection based on this information. The third goal is to compare the effectiveness of distributed and centralised retrieval in the same environment.

BACKGROUND

This section provides background to distributed information retrieval and server selection on the Web.

Distributed Information Retrieval

A distributed information retrieval system normally comprises document servers, search servers and a broker. Document servers, for example Web (HTTP) servers, simply provide documents. Search servers index and search some set of documents, from one or more document servers. They respond to a query q with a results list $R_i = \langle D_i, o_i \rangle$, a document set D_i in some rank order o_i (o_i is an ordering over D_i). Search servers may be large and general (`altavista.com`) or small and specialised (`search.anu.edu.au`), although this paper concentrates on smaller servers, such as those listed by InvisibleWeb and CiteLine Professional, indexing high-quality document collections.

A broker facilitates searches over multiple search servers. In general, given a set of search servers S and a user query q, the process of distributed information retrieval is as follows:

$$S + q \xrightarrow{Selection} S' + q \xrightarrow{Retrieval} R_1 \ldots R_{|S'|} + q \xrightarrow{Merging} R_m$$

This is also illustrated in Figure 1. The steps are:

Selection Select a subset S' of servers S ($S' \subseteq S$), to achieve low search cost — in terms of elapsed time, network communication and computation — and high search effectiveness. Effectiveness means maximising the quality of the merged results list R_m.

Retrieval Retrieve search result lists $R_1 \ldots R_{|S'|}$ using appropriate communication protocols, query translation methods and results parsing methods, by applying q at each server in S'.

Merging Generate $R_m = \langle D_m, o_m \rangle$ such that $D_m = D_1 \cup \ldots \cup D_{|S'|}$ and o_m is an effective ranking. An effective ranking is one in which documents that the user finds relevant are listed before those the user finds irrelevant.

Selection and merging methods may be very simple. For example, a simple selection method is to select all servers so $S' = S$. A simple merging method is to present the concatenation of the results lists. However, being concerned with effectiveness, this paper chooses a single highly effective merging method and evaluates a number of selection methods each designed to maximise retrieval effectiveness.

Distributed vs Centralised Web Search

This section compares two prominent ways of providing a full-text search facility over Web documents. One is a centralised solution, which involves building a new search server which covers all documents of interest (assuming they are available for indexing). The other is a distributed solution, which involves using a broker to address existing search servers. A real Web search system may be based on one or the other or a combination of the two. The decision of which to use is based on properties of the search servers and document servers in question.

One drawback of central indexing is the running cost, measured in terms of network traffic costs and required computer power and storage. Network traffic includes both the cost of downloading documents every time the index is updated and that of responding to queries across the network. Total cost increases with the number of documents centrally indexed, the frequency of index updates and query load. It should not be underestimated: even dedicated search engine companies have trouble indexing more than 16% of indexable Web pages, and their indexes may become weeks or months out of date [16].

Another problem with central indexes is that they may miss documents. One common reason for this is that central indexes usually require hyper-links to locate, download and ultimately index documents. Particularly for Web front ends to non-Web document collections, such links may not exist, meaning that the only method for locating documents is through the local site-search server. InvisibleWeb claim that all 10 000 of their listed search servers index documents invisible to central indexing.

Distributed retrieval also has disadvantages. Between receiving a query and providing results, a centralised search server need only perform disk IO and computation. A broker is

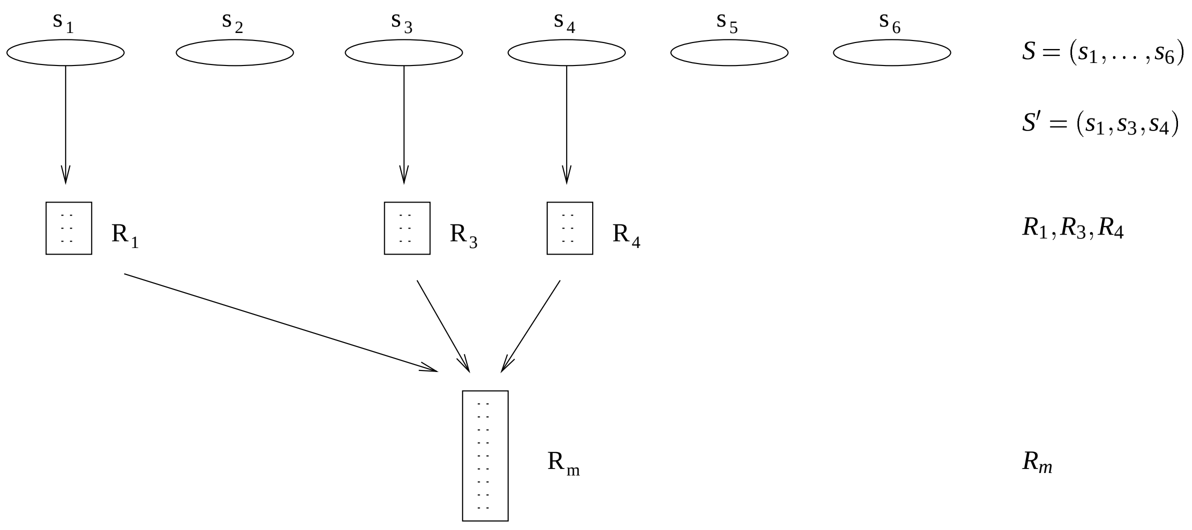

Figure 1: Distributed information retrieval. The broker selects S' a subset of servers S, retrieves results from each server in S' and creates a single merged results list for presentation to the user.

likely to be slower, first performing selection, then retrieval — which is not finished until the last results list arrives over the network — and then merging. Previous studies have also found that distributed retrieval with server selection is less effective than centralised retrieval [22, 1]. Results later in the paper show the same, even when the central index can only index documents from 50% of the document servers.

The decision of which approach to use depends on system goals and resources availability. Fast and effective results can be provided by a centralised solution, at the cost of building, maintaining and searching a large index and missing out on certain documents. Affordable coverage of more documents can be achieved with a distributed system, at the cost of query processing speed and possibly effectiveness.

Selection Motivation

Assuming that a distributed approach has been chosen, there are two reasons to perform server selection, as opposed to selecting $S' = S$. First, selection improves search efficiency. To obtain each results list R_i involves use of limited network bandwidth and monetary charges for network traffic, plus computation at both broker and search server. Selecting and querying fewer servers reduces the total cost of distributed search.

Second, selection can also improve effectiveness. Querying a server whose results R_i contain no relevant documents is at best a waste of time, computer and network resources. At worst, the inclusion of R_i's irrelevant documents in R_m can demote relevant documents and reduce effectiveness. This effect is particularly important in situations of large topic skew, where a small proportion of servers contain all relevant documents for a topic, as is the case in this study (Figure 4).

Server Selection Methods

Past studies have looked at selection based on both maximising selection effectiveness and minimising search costs. Search costs include elapsed search time, network traffic charges and possible per-search monetary charges. Fuhr [10] developed a decision theoretic model for selection based on cost and benefit components. ProFusion performs selection based on past search server query processing speed [7]. This paper concentrates on effectiveness, taking a simple view of search cost. It assumes all servers have equivalent search costs and the user is willing to bear the cost of selecting only ten servers per query. This number reflects the number of servers queried by existing Web search brokers. Each of the following methods allows a broker to build a server ranking based on the current query, then we assume the broker selects the top 10 ranked servers.

The CORI [1] method ranks search servers as document surrogates, consisting of the concatenation of the server's documents. The belief $p(r_1,\ldots,r_M|s_i)$ in server s_i due to observing terms $r_1,\ldots,r_M$ is determined by:

$$\begin{aligned}
I &= d_t + (1-d_t)\cdot\frac{\log(DF_{i,k}+0.5)}{\log(DF_i^{max}+1.0)} \\
T &= \frac{\log(\frac{|S|+0.5}{SF_k})}{\log(|S|+1.0)} \\
p(t_k|s_i) &= d_b + (1-d_b)\cdot T\cdot I \\
p(t_1,\ldots,t_M|s_i) &= \frac{\sum_{k=1}^{M} p(t_k|s_i)}{M} \qquad (1)
\end{aligned}$$

$DF_{i,k}$ is the document frequency of term t_k in server s_i. DF_i^{max} is the DF of the most frequent term in s_i. $|S|$ is the total number of servers, and SF_k is a server-frequency statistic, the number of servers containing term t_k. M is the number of unique query terms. This study uses a default term frequency value (d_t) and default belief (d_b) each set to 0.4, as in [1]. Servers are ranked in descending order of belief. The server ranking is based on a modification of an INQUERY document ranking algorithm, using *DF* instead of *TF* (term frequency, the number of times a term occurs in a document) and *SF* instead of *DF*.

The CVV [23] server ranking method is based on the Cue-Validity Variance (*CVV*) of query terms. Terms which can better discriminate between servers have a higher *CVV* and therefore contribute more to the suitability score S_i:

$$\begin{aligned} CV_{i,j} &= \frac{\frac{DF_{i,j}}{SS_i}}{\frac{DF_{i,j}}{SS_i} + \frac{\sum_{k\neq i}^{|S|} DF_{k,j}}{\sum_{k\neq i}^{|S|} SS_k}} \\ CVV_j &= \frac{\sum_{i=1}^{|S|}(CV_{i,j} - \overline{CV_j})^2}{|S|} \\ S_i &= \sum_{j=1}^{M} CVV_j \cdot DF_{i,j} \qquad (2) \end{aligned}$$

where SS_i is the size, in documents, of server s_i. $\overline{CV_j}$ is the population mean of $CV_{i,j}$ over all servers. Servers are ranked in decreasing order of scores S_i.

The vGlOSS [11] server ranking methods $Max(l)$ and $Sum(l)$ rank servers using server document frequency statistics and the vector sum of the server's normalised document vectors. For server s_i and term t_k, the summed weight is:

$$cwt_{i,k} = \sum_{j=0}^{j<SS_i} wt_{j,k}$$

where $wt_{j,k}$ is the vector weight of term t_k in the document vector representing document d_j of server s_i.

$Max(l)$ and $Sum(l)$ estimate the summed scores of documents scoring above l, with the current query and a vector space retrieval model. Also proposed in [11] was a ranking used for evaluation $Ideal(l)$, the actual summed document scores for documents scoring above l. For $l = 0$, all three produce the same ranking, so $Max(0) = Sum(0) = Ideal(0)$. The score S_i achieved by a server s_i when $l = 0$ is:

$$S_i = \sum_{j=1}^{M} cwt_{i,j} \qquad (3)$$

This formulation is used in evaluation experiments below. The same value of l was also used in [9] and [8], although under a different evaluation framework.

Server ranking methods based on user input were excluded from this study. The Web search broker SavvySearch [6] performs server selection. For a query containing term t, a server's future selection score for t is boosted if the user visits a page returned by that server, and is reduced if the server returns no results. The Web search broker ProFusion also performs selection based on user visits to results pages and query categorisation [7]. Voorhees [21] proposed two server selection methods which rely on past relevance judgments of the top 100 merged results for each training query. Since visit events and relevance judgments are not available over WT2g (the test collection used here, see below) for all the training queries used in this paper, none of the above methods are evaluated.

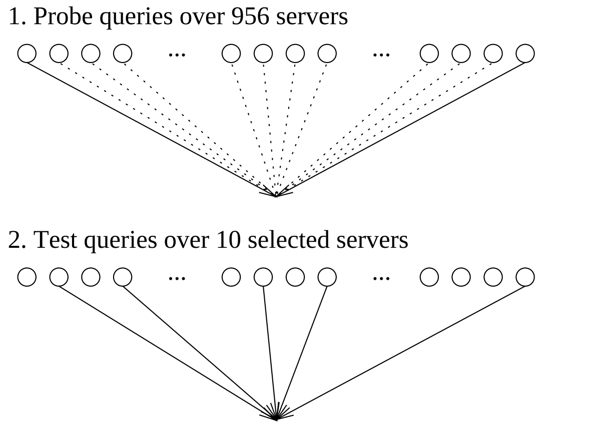
1. Probe queries over 956 servers
2. Test queries over 10 selected servers

Figure 2: Probe queries and test queries. 1) Before query time, the broker sends each probe query to all search servers downloading their top results. The broker can then extract statistics from the documents for use in server ranking, and can also use the results to estimate server effectiveness. 2) Given a new query, apply server selection methods to select the top ten servers.

Server selection methods based on server categorisation are also not evaluated here. Methods relying on manual classification such as used by CiteLine Professional and InvisibleWeb would be impractical due to the lack of such classification over WT2g documents. Due to time constraints even methods based on automatic classification such as Pharos [5] are not evaluated here.

PROBE QUERIES AND ESTIMATING SERVER EFFECTIVENESS

Web search servers normally do not export any information about the documents they index, such as term occurrence statistics, for use in selection. For this reason, information needs to be extracted using the lowest common denominator: the ability of a search server to return search results. Callan, Connell and Du [2] suggested query based sampling of documents, by sending a series of probe queries to all servers (see Figure 2) before query time, downloading the documents returned and extracting term occurrence statistics from those documents. Although this provides a non-random sample of a server's documents, the study [2] found extracted statistics to be representative of full server statistics. The approach is also appealing because: 1) it can be applied to any search server whose returned documents are available for download, and 2) a broker can address a new server at any time simply by sending it the probe queries.

Hawking and Thistlewaite [13] also proposed a selection method based on probe queries, called light-weight probes. However, light-weight probes were performed at query time, not before, and were designed to obtain up-to-date term frequency statistics from cooperating servers rather than sampling documents from non-cooperating servers.

This study takes the document sampling approach of Callan, Connell and Du a step further. Statistics extracted from sampled documents may be indicative of those over a server's full document set. However, such statistics give no indication of a server's ability to return any relevant documents in that set. Consider two servers indexing the same documents, one running a highly effective retrieval system and the other performing boolean unranked retrieval. Based on statistics alone these servers are identical, but the former is almost certainly a better selection than the latter in practice. In an environment such as the Web where many different retrieval systems are used and retrieval effectiveness varies [12], we hypothesize that estimating server effectiveness is important.

Our initial method for estimating server retrieval effectiveness is unrefined (results later in the paper indicate it requires further study and tuning). We choose a broker-specific point in the ranking — ten documents, since ten per server are included in the merged list — and estimate effectiveness across all probe queries. By contrast, a selection model by Fuhr [10] would be based on query-specific expected precision at each point in the results ranking. However, our method is a first empirical step in estimating server effectiveness and using that estimation in selection.

The broker sends probe queries to each server as suggested by Callan, Connell and Du. However, queries are multi-term, as opposed to the single term queries used in [2]. For each probe query the broker builds a highly effective merged list (potentially containing documents from all servers in S) based on downloaded document contents and merging methods suggested in [3]. Use of multi-term queries makes the merging problem more interesting.

Under the assumption that the top documents in the merged list are of high quality, and in the absence of real human relevance judgments, the broker marks the top 20 documents in the merged list as "relevant". This technique is similar to one used in automatic relevance feedback where top ranked documents are used for query expansion. Then the broker can calculate which servers tend to return "relevant" documents. Specifically, E_i is the expected (mean) number of relevant documents returned by a server s_i. For example, in the first ten probe queries simulated here, the server $s_n =$ `www.cpac.org` has eight documents in the ten merged top 20s, giving an expected number of $E_n = \frac{8}{10} = 0.8$.

Since E_i is query independent, it is used in conjunction with a content based server ranking method. Since CORI was the best unmodified server ranking method, we modify it with E_i as follows. We modify the CORI belief value p by adding E_i times some constant, and base the new ranking on the modified belief values. The constant was calculated by taking the ideal expected effectiveness value E'_i, which is based on real relevance judgments over the test queries, and tuning the constant to maximise selection effectiveness. This constant (0.03) was then used to combine p and E in evaluation.

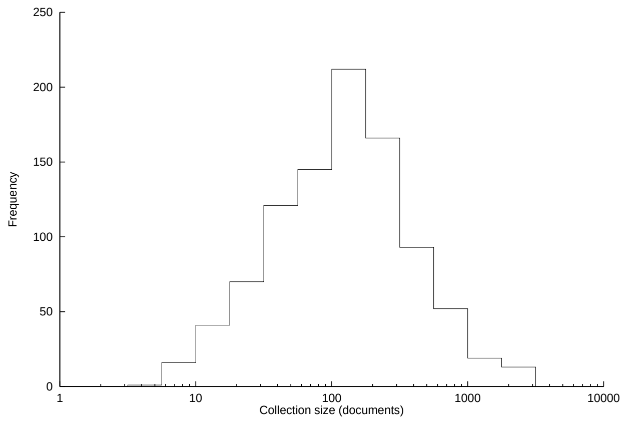

Figure 3: Server sizes. The evaluation here uses a range of server sizes.

EVALUATION FRAMEWORK

This experiment evaluates the effectiveness of a broker addressing a large number of Web search servers, using various server selection methods. Real Web site-search servers have the following properties. There are many of them. They index and search Web documents. Different servers use different retrieval methods, cover widely disparate numbers of documents and often concentrate on documents in a specific topic area. Each server indexes an offering of Web documents from a single document provider. This section describes how these elements are modeled using a test collection.

The test collection is from the official TREC-8 Small Web Track [14], comprising the WT2g document collection, TREC queries 401-450 and official NIST relevance judgments. The two gigabytes of Web documents in the collection come from 956 document servers, and are partitioned along document server boundaries into 956 simulated search servers. These search servers have a number of good properties. First, each covers an offering of Web documents from a single document provider, for example there are search servers for `greenpeace.org` and `newsnet.com`. Second, the servers have a wide range of sizes, from a handful of documents to several thousand (see Figure 3). Unfortunately, although the sizes vary they are too small. A real Web search broker might face a higher proportion of large servers. However, this limitation of the test collection is considered ac-

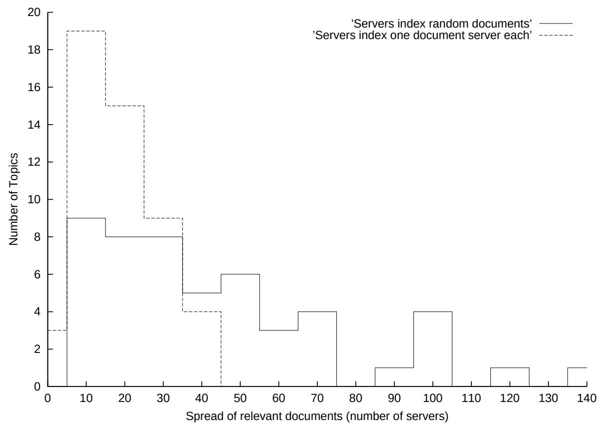

Figure 4: Topic skew. Instead of documents being spread randomly across servers, the evaluation here has a high degree of topic skew. That is, relevant documents for a given topic tend to be concentrated in a few servers. This sort of skew probably occurs often on the Web. Here it is a result of the partitioning of WT2g along document server boundaries.

ceptable. Third, the servers have a realistic topic skew, with relevant documents for a given topic tending to be concentrated at a few servers rather than spread across many servers (see Figure 4).

Retrieval by each server is simulated using one of three retrieval algorithms. The first is highly effective, the Cornell variant of the Okapi BM25 probabilistic ranking function [19, 17]. The second is less effective, ranking servers based on their summed query term frequencies, simulating a Web search server which gives documents a bonus for having more term occurrences but does not use more sophisticated models or statistics. The third retrieval method simply assumes a boolean conjunction of the query terms, listing matching documents in arbitrary order. The BM25 ranking function was modified slightly for use in retrieval here. In the usual BM25 function, presence of a term which occurs in more than half the server's documents will reduce a document's relevance weight. Because such occurrence statistics are much more likely with the small server sizes and high topic skew, any negative values due to high document frequency proportions were eliminated, allowing each term to have only non-negative effects on the RSV.

Retrieval algorithms were assigned to search servers as follows. The servers were listed from those with the least to those with the most documents. Then the three retrieval algorithms were assigned in alternating order. This ensured that roughly an even number of servers used each retrieval algorithm, and that some very small and some very large servers used each. It was not possible to consider more than one assignment within the resources available for the experiment.

Merging is based on downloading documents D_m and applying Okapi BM25 document ranking. This method, which was found to be highly effective in [3], requires the broker to use reference collection statistics. These statistics were taken from a 10% sample of the 100 gigabyte TREC VLC2 Web-data collection [14], and are used in place of accurate collection statistics which would not be available to a Web search broker. Reference statistics are described further in [3]. Merging is carried out over up to 100 documents, because the broker selects ten servers per query and each returns up to ten documents. Servers only return less than ten documents if they do not have enough which match the query.

In evaluation probe queries are first sent to all servers. Then information from probe queries is used to select ten servers for each test query. Probe queries are used as described in the previous section, for both extracting collection statistics and estimating server effectiveness. For each test query, results are merged then evaluated according to precision at 20. Precision at 20 was chosen because Web users do not often look beyond the first page of search results [20] and most Web brokers return 20–50 results on their first page [15, 18, 7]. Precision at 20 has also been used in official TREC Web-data experiments [14].

EXPERIMENTS

Since probe queries themselves are expensive to carry out, requiring the broker to download thousands of documents, several levels of probing were evaluated. The 200 probe queries were the titles of TREC topics 151–200 and 251–400 (201–250 have no titles). Levels of probing were at 10, 25, 50, 100, 150 and 200 probe queries, preferring newer queries (so the 10 were 391–400). Test queries were also title-only, from TREC topics 401–450.

Probe queries have some interesting implications in implementation. For example, it is possible for probe queries to retrieve no documents from a server. For the two term query "*A B*", a BM25 server will return no documents if no document contains *A* or *B*. A boolean server will return no documents if none contain both *A* and *B*. Ten probe queries found some information on 702 servers, 25 probe queries on 829, 50 on 915 servers, 100 on 933, 150 on 948 and 200 also on 948. The 8 missing servers for the 150 and 200 probe levels indexed from 9 to 217 documents (on average 63) and all were boolean servers. Amongst the implications of having no information on a server are that it will never be selected, and that SS_i can be 0. When computing server rankings in such cases, for example with Yuwono selection, division by zero was handled by making the result $0/0 = 0$.

To compare distributed vs centralised is difficult under this Web scenario. On the real Web only some servers allow their documents to be indexed. Because it is not clear what proportion of servers allow what proportions of their documents to be indexed, central indexing under three levels is considered: all servers, 50% of servers and 25% of servers. As with

		Precision @20	
Selection method	Proportion selected	Mean	Standard Deviation
Ten most relevant	10/956	0.365	0.222
Centralised 100% index	N/A	0.343	0.239
Select all servers	956/956	0.247	0.266
Centralised 50% index	N/A	0.227	0.222
CORI plus E_i'	10/956	0.208	0.205
CORI plus E_i *	10/956	0.200	0.208
CORI *	10/956	0.190	0.202
vGlOSS Max(0) *	10/956	0.155	0.213
Centralised 25% index	N/A	0.135	0.170
CVV *	10/956	0.131	0.168
Random selection	10/956	0.011	0.043

Table 1: Distributed vs centralised. This table presents results over test topics 401-450. It lists results for four realistic selection scenarios, each selecting ten servers based on 50 probe queries (marked with *). It also presents results for centralised BM25 retrieval over 100%, 50% and 25% of the servers (using title only queries and no relevance feedback as used in the distributed case). Finally it presents results based on selecting all 956 servers, selecting the ten with the most judged-relevant documents and selecting ten random servers.

the assignment of retrieval systems, the set of servers which are unavailable for indexing is selected by ordering servers in size order and indexing every second or every fourth. For all levels of centralised indexing, the retrieval method is BM25, with the same title-only queries used in distributed retrieval and no relevance feedback.

RESULTS

DISCUSSION

It is interesting that selection based on fewer than 50 000 documents has comparable effectiveness to that based on all 250 000 documents. This indicates that probe queries are a good way of extracting information from Web search servers. CORI selection degrades most with fewer probe queries, although when modified by E_i' the effect is reduced. Surprisingly the rightmost point for CORI plus E_i' is lower than the unmodified CORI point, although the difference is not statistically significant.

Modified CORI selection based on E_i disappointingly shows no significant improvement on CORI selection. This may be due to the method of estimation or its tuning. The possibility of some improvement using E_i', however, confirms the intuition that selection can be improved if it takes into account server effectiveness. Even the results based on E_i' might be improved, perhaps using a different method for combining CORI and E_i'. One reason for the small scope for improvement, even in the case of E_i', might be the small server sizes in the current experiment. With small servers the same documents are likely to be returned regardless of the quality of the ranking method, because only a few documents per server contain query terms. With larger servers, and hundreds or thousands of documents per server containing query terms, variation in retrieval effectiveness might become more apparent and its estimation more useful.

The initial upward slope of CVV and vGlOSS from the full information to the 200 probe query case may indicate some problem with server size normalisation. Without size normalisation, the broker might be confused by size variation in servers tending to select larger servers. Probe queries mask server size variation, perhaps explaining the improvement of CVV and vGlOSS. This fits with the observation by French et al [8] that vGlOSS correlates well with a size-based server ranking.

A notable feature of CVV is that it gives a bonus to query terms with high document frequency variance, assuming they will be good discriminators between servers. However, this may not be a good assumption, at least in the current collection. For example, the test queries included the terms *fluids* and *mirjana*. In document ranking the latter term would be considered a more important than the former, because it is rare. Instead, in the CVV method, *fluids* has the highest CVV of all query terms and *mirjana* has the lowest. This is because the document frequency of *mirjana* is usually 0 and sometimes 2 or 3. By contrast, *fluids* appears in 297 servers in between 1–232 documents in each. The intuition that terms with varying document frequencies are good discriminators may not be correct, particularly for terms which only appear a few times in a few servers such as *mirjana*.

The fact that a relevance based selection of ten servers can be more effective than 956 shows that there is potential for server selection to enhance a search broker's effectiveness (as opposed to selection $S' = S$). However, current algorithms do not provide such an improvement in effectiveness when selecting ten servers. It would be interesting to experiment with selecting more than ten servers, to see what level is required to improve on the effectiveness of a all-server selection. Note that the current results do not mean that selection is unnecessary. Compared to a selecting all 956 servers, a selection of ten requires only $\frac{10}{956} \approx 1\%$ of the retrieval effort and while loosing roughly 0.05 on precision at twenty.

Another interesting result is that a highly effective selection of only ten servers, based on knowledge of relevance judgments, outperforms the 100% centralised retrieval using BM25 (Table 1). Centralised BM25 results could be improved using relevance feedback and other techniques. However, so could the result in the distributed case. Further, the distributed result is based on a fraction of the WT2g documents (from only ten servers) and retrieval using heterogeneous servers. A likely explanation for ten servers beating a centralised index stems from topic skew. The ten servers with the most relevant documents on average contained 83% of a topic's the relevant documents, and for 15 topics contained all relevant documents. Particularly in the latter cases

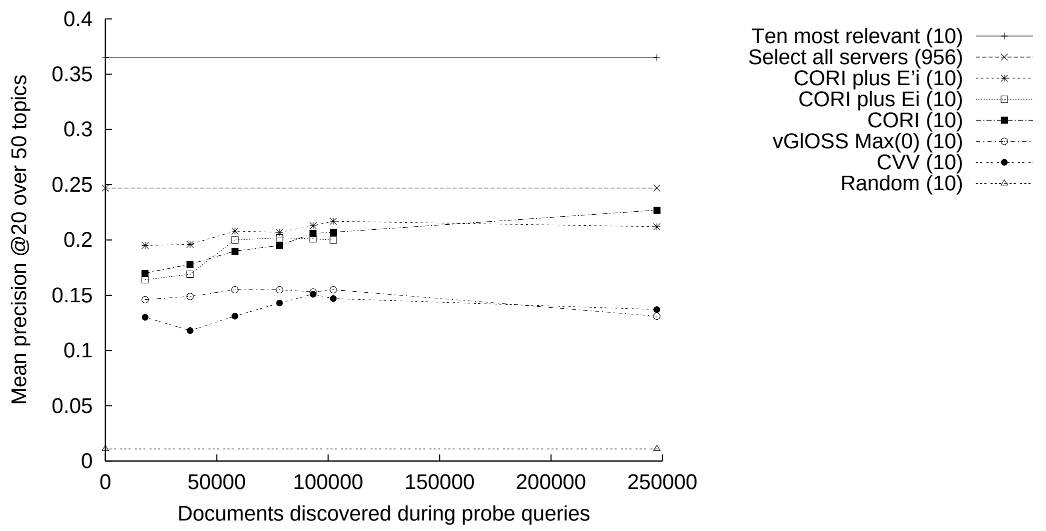

Figure 5: Selection effectiveness with probe queries. This figure presents evaluation results over various levels of probe queries for test topics 401–450. Modified CORIs, CORI, vGlOSS and CVV have data points for (from left to right) 10, 25, 50, 100, 150 and 200 probe queries. The rightmost points for CORI plus E_i', CORI, vGlOSS and CVV are based on full collection information, not probe queries. There is no equivalent point for CORI plus E_i because E_i is based on probe information. For reference, the figure also includes three results from Table 1, based on selecting ten servers with the most relevant documents, all 956 servers and ten random servers. Numbers in brackets are the number of servers selected per query.

retrieval over the subset containing all relevant documents is likely to be effective. Centralised 100% retrieval adds a huge proportion of irrelevant documents, and in the case of this experiment leads to worse early precision.

The results presented here are for a specific type of user in a specific configuration of distributed environment. The user is most interested in the top 20 search results, and is only prepared to wait for the broker to select and search ten servers, merging at most 100 results. The configuration of the distributed environment includes a specific assignment of retrieval algorithms to search servers and centralised indexes of 50% and 25% of servers are based on a single server subset each. Therefore the results describe what happens in one realistic Web search configuration — more realistic than any previous experiments in terms of document type, topic skew and server heterogeneity — but it is not known whether the results would generalise to a general distributed Web search scenario. Only more experiments of this type will test the generality of these results.

CONCLUSION

This paper evaluated selection methods applicable over Web search servers. Such methods could be implemented in a digital library's distributed information retrieval broker, allowing it to address a large number of servers and send each query to a selected subset. The evaluation was designed to realistically model present-day Web search servers, such as those listed by InvisibleWeb and CiteLine, by using real Web documents, heterogeneous retrieval algorithms and real server divisions. As far as we are aware it is also the only study presently in publication to evaluate selection based on probe queries. In this environment, the paper evaluates the effectiveness of the CORI, vGlOSS and CVV selection methods. It also introduces a method for estimating server effectiveness when conducting probe queries.

Findings were:

- CORI is best for selecting a set of 10 servers, outperforming vGlOSS and CVV.
- All selection methods maintain a reasonable level of effectiveness based on probe queries, even on as few as 10 or 25. With low numbers of probe queries, CORI effectiveness degrades more than that of vGlOSS or CVV.
- Modifying CORI according to estimated server effectiveness E_i provided no significant improvement to overall effectiveness.
- Although the modification of CORI is crude, modifying by E_i' (effectiveness based on relevance judgments) yielded a statistically significant improvement for ten and 25 probe queries.
- Distributed retrieval over ten servers per query is not as effective as retrieval over a centralised index with 100% coverage. However, in the Web case such an index is unrealistic, and for 50% and 25% coverage the distributed case fares better.

This means a digital library could incorporate server selection based on only a few probe queries and expect performance comparable to selection based on full server information. In addition, selection may be improved by taking into account both a server's documents and its retrieval effectiveness, although our initial attempts at estimating such effectiveness were not successful. Further study and tuning of estimation methods and methods for incorporating estimates into selection might yield better results.

In future experiments it would be interesting to vary the user and system model presented here. For example, instead of modeling a typical Web user interested only in early precision, future experiments could model a user performing research, selecting from more servers and attempting to maximise recall. In addition, construction of a new Web-data test collection is currently underway by the authors, which should contain more large servers, allowing for even more realistic evaluation.

REFERENCES

1. James P. Callan, Zihong Lu, and W. Bruce Croft. Searching distributed collections with inference networks. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of ACM SIGIR '95*, pages 12–20, Seattle, Washington, July 1995. ACM Press.

2. Jamie Callan, Margaret Connell, and Aiqun Du. Automatic discovery of language models for text databases. In *Proceedings of the 1999 ACM International Conference on Management of Data (SIGMOD 99)*, New York, 1999. ACM.

3. Nick Craswell, David Hawking, and Paul Thistlewaite. Merging results from isolated search engines. In John Roddick, editor, *Proceedings of the 10th Australasian Database Conference, Auckland, NZ*, pages 189–200. Springer-Verlag, January 1999. `http://pastime.anu.edu.au/nick/pubs/adc99.ps.gz`.

4. W. Bruce Croft, Alistair Moffat, C.J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors. *ACM SIGIR 98*, Melbourne, Australia, August 1998. ACM Press, New York.

5. R Dolin, D Agrawal, and A El Abbadi. Scalable collection summarization and selection. In Edward A Fox and Neil Rowe, editors, *ACM DL'99*, pages 49–58, August 1999.

6. Daniel Dreilinger and Adele E. Howe. Experiences with selecting search engines using metasearch. *ACM Transactions on Information Systems*, 15(3):195–222, July 1997.

7. Yizhong Fan and Susan Gauch. Adaptive agents for information gathering from multiple, distributed information sources. In *1999 AAAI Symposium on Intelligent Agents in Cyberspace*. Stanford University, March 1999.

8. James French, Allison Powell, Jamie Callan, Charles Viles, Travis Emmitt, Kevin Prey, and Yun Mou. Comparing database selection algorithms. In Marti Hearst, Fredric Gey, and Richard Tong, editors, *ACM SIGIR 99*, pages 238–245, Berkeley, CA, August 1999. ACM Press, New York.

9. James C. French, Allison L. Powell, Charles L. Viles, Travis Emmitt, and Kevin J. Prey. Evaluating database selection techniques: A testbed and experiment. In Croft et al. [4], pages 121–129.

10. Norbert Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 17(3):229–249, July 1999.

11. Luis Gravano and Hector Garcia-Molina. Generalising GlOSS to vector-space databases and broker hierarchies. In *Proceedings of the 21st VLDB Conference*, Zurich, Switzerland, 1996. Morgan Kaufmann, San Francisco CA. Also Stanford technical report CS-TN-95-21.

12. David Hawking, Nick Craswell, Paul Thistlewaite, and Donna Harman. Results and challenges in Web search evaluation. In *Proceedings of WWW8, Toronto*, pages 243–252. Elsevier, 1999. `http://pastime.anu.edu.au/nick/pubs/www8.pdf`.

13. David Hawking and Paul Thistlewaite. Methods for information server selection. *ACM Transactions on Information Systems.*, 17(1):40–76, 1999.

14. David Hawking, Ellen Voorhees, Nick Craswell, and Peter Bailey. Overview of the TREC-8 Web Track. To appear TREC-8., 1999.

15. Steve Lawrence and C. Lee Giles. Inquirus, the NECI meta search engine. In *Proceedings of the Seventh International World Wide Web Conference*, 1998. `http://www7.scu.edu.au/programme/fullpapers/1906/com1906.htm`.

16. Steve Lawrence and C. Lee Giles. Accessibility of information on the web. *Nature*, 400:107–109, July 1999.

17. S. E. Robertson, S. Walker, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In D. K. Harman, editor, *Proceedings of the Third Text Retrieval Conference (TREC-3)*, Gaithersburg, MD, November 1994. U.S. National Institute of Standards and Technology. NIST special publication 500-225.

18. E. Selberg and O. Etzioni. The MetaCrawler architecture for resource aggregation on the Web. *IEEE Expert*, V(N):11–14, January–February 1997.

19. Amit Singhal, Gerard Salton, Mandar Mitra, and Chris Buckley. Document length normalization. Technical Report TR95-1529, Department of Computer Science, Cornell University, Ithaca NY, 1995.

20. Amanda Spink, Judy Bateman, and Bernard J. Jansen. Searching the web: a survey of excite users. *Internet Research: Electronic Networking Applications and Policy*, 9(2):117–128, 1999.

21. Ellen M. Voorhees. Siemens TREC-4 report: Further experiments with database merging. In D. K. Harman, editor, *Proc. Fourth Text Retrieval Conference (TREC-4)*, pages 121–130, Gaithersburg, MD, November 1995. U.S. National Institute of Standards and Technology.

22. Jinxi Xu and Jamie Callan. Effective retrieval with distributed collections. In Croft et al. [4], pages 112–120.

23. Budi Yuwono and Dik L. Lee. Server ranking for distributed text retrieval systems on the internet. In Rodney Topor and Katsumi Tanaka, editors, *DASFAA '97*, pages 41–49, Melbourne, April 1997. World Scientific, Singapore.