Noname manuscript No.
(will be inserted by the editor)

Server selection methods in personal metasearch: a comparative empirical study

Paul Thomas · **David Hawking**

the date of receipt and acceptance should be inserted later

Abstract Server selection is an important subproblem in distributed information retrieval (DIR) but has commonly been studied with collections of more or less uniform size and with more or less homogeneous content. In contrast, realistic DIR applications may feature much more varied collections. In particular, personal metasearch—a novel application of DIR which includes all of a user's online resources—may involve collections which vary in size by several orders of magnitude, and which have highly varied data.

We describe a number of algorithms for server selection, and consider their effectiveness when collections vary widely in size and are represented by imperfect samples. We compare the algorithms on a personal metasearch testbed comprising calendar, email, mailing list and web collections, where collection sizes differ by three orders of magnitude. We then explore the effect of collection size variations using four partitionings of the TREC ad hoc data used in many other DIR experiments. Kullback-Leibler divergence, previously considered poorly effective, performs better than expected in this application; other techniques thought to be effective perform poorly and are not appropriate for this problem. A strong correlation with size-based rankings for many techniques may be responsible.

Keywords server selection, distributed information retrieval

1 Introduction

Distributed information retrieval (DIR) systems, or metasearchers, present an alternative to centralised indexes when searches cover several independent collections. A *broker* forwards users' queries to each collection independently, collates the results, and presents a single result set.

Personal metasearch is a novel application of DIR techniques which aims to incorporate all of a user's online resources including, for example: email, calendars, local databases,

P Thomas
CSIRO ICT Centre, GPO Box 664, Canberra ACT 2601, Australia
E-mail: paul.thomas@csiro.au

D Hawking
Funnelback Pty Ltd, PO Box 1441, Dickson ACT 2601, Australia
E-mail: david.hawking@acm.org

and private or public Web sites. A personal metasearch tool would have several advantages: it could provide much greater coverage than any single system, could allow per-collection optimisations, and could significantly reduce user effort.

Unified search of the information sources routinely consulted by a knowledge worker is a particularly important DIR application, because there appears to be no feasible alternative to a metasearch approach. It is infeasible for individuals to crawl and index the whole Web or the Dialog database from their personal computer; it is equally infeasible for external search engines like Google, Yahoo! and Live to crawl and include in their index, the documents and email from everyone's personal computers. In contrast to earlier DIR applications, the types of collections used in personal metasearch will be highly heterogenous, varying from small private collections such as calendars to large public collections such as the public Web. This contrasts with systems such as FedLemur (Avrahami et al 2006), AllInOneNews (Liu et al 2007), or "vertical" selection by web search engines, where data types are more constrained or characteristics of each collection are well-known.

Brokers in personal metasearch or other DIR applications may include *server selection* as a normal part of query processing. It may not be feasible to forward every query to every server, as costs will be incurred in network traffic, delay, access costs, and an increased risk of unavailability. Server selection aims to choose the server or servers most useful for answering each query. As well as minimising costs and increasing reliability, there is some evidence that a selection algorithm can improve the quality of a result set even over that from a single index (Craswell et al 2000; Powell et al 2000; Xu and Croft 1999).

In implementing personal metasearch, we observed collections which varied in size by several orders of magnitude and which are in some cases hard to sample. However, previous evaluations have typically used testbeds based on TREC ad hoc or Web Track data (French et al 1999, 1998; Powell and French 2003). This may not represent real DIR scenarios. Past evaluations have also assumed perfect knowledge of collection contents, which is not generally possible. This paper presents experiments evaluating server selection with varied collections and with imperfect knowledge of collections based on current sampling techniques. Twelve selection methods are considered in these experiments. Of these, CORI and extensions, CVV, vGlOSS, Kullback-Leibler divergence and extension, and ReDDE have been well-tested in other scenarios, but never all at once and never on a highly heterogenous testbed. bGlOSS, Zobel's "I", and CRCS are less well-studied.

2 Selection techniques

From the large number of server selection algorithms in the literature, our experiments consider twelve which we consider applicable to a variety of tasks. Each selects one or more collections from $\mathscr{C}$, the set of collections available, for each user query q. They do this by scoring each possibility: given the query, they produce a score $s_c(q)$ for every collection c in $\mathscr{C}$. A broker can rank the collections by this score, and choose either those collections with scores above a threshold or some number of top-scoring collections. The servers handling these collections are selected to receive the query.

2.1 The GlOSS family

The “glossary of servers server” (GlOSS) family of algorithms estimates the number of documents in each collection which a user will find interesting, and ranks collections accordingly. We consider two versions.

2.1.1 bGlOSS

The original GlOSS server selection algorithm, later called the Boolean version or bGlOSS (Gravano et al 1994), assumes a Boolean retrieval model and that the interesting documents in a collection are those which match all terms in the query q. If these terms, $\mathscr{T}_q$, appear independently, the expected number of interesting documents can be estimated by

$$s_c(q) = N_c \prod_{t \in \mathscr{T}_q} \frac{\mathrm{df}_c(t)}{N_c}, \tag{1}$$

where $\mathrm{df}_c(t)$ is the document frequency of term t in collection c and N_c is the size of c. As originally described, bGlOSS selects only the collection with the highest score. In the experiments described here, it is used in a more conventional way to rank all available collections.

Although the servers used in the experiments below do not use a Boolean model, bGlOSS is similar to other selection techniques based on different assumptions and is therefore included for comparison.

2.1.2 vGlOSS

The vector-space version of GlOSS, vGlOSS (Gravano et al 1999), assumes users are interested not in documents containing all query terms but in those documents whose similarity to the query is more than a threshold l. vGlOSS uses either of two algorithms—MAX and SUM—to estimate the number of interesting documents. A common application considers any document at all similar to the query to be interesting, and therefore sets $l = 0$. In this case, MAX and SUM are identical:

$$\begin{aligned} s_c(q) &= \textsc{max}(0, q, c) = \textsc{sum}(0, q, c) \\ &= \sum_{t \in \mathscr{T}_q} w_q(t)\, w_c(t) \end{aligned} \tag{2}$$

where $w_q(t)$ and $w_c(t)$ are the weight of term t in the query and in the collection respectively. In the original presentation, $w_c(t)$ is calculated at the server, for example using term frequency and inverse document frequency, and is then made available to the broker. In a DIR application without cooperative servers, however, this weight is more likely to be estimated from a sample.

2.2 CORI

The CORI algorithm (Callan et al 1995) adapts INQUERY document ranking. It treats each server as a compound “document”, using document frequency (df) instead of term frequency and collection frequency (cf) in place of document frequency. Each term in the query is scored separately, according to document frequency (T) and inverse collection frequency (I):

$$\begin{aligned} s_c(t) &= b + (1-b)TI; \\ \text{where } T &= \frac{\mathrm{df}_c(t)}{\mathrm{df}_c(t) + \mathit{df\_base} + \mathit{df\_factor}\,(cw_c/\overline{cw})} \\ \text{and } I &= \frac{\log\left(\frac{|\mathscr{C}|+0.5}{\mathrm{cf}_{\mathscr{C}}(t)}\right)}{\log(|\mathscr{C}|+1)}, \end{aligned}$$

where cw_c is the number of terms in collection c and $\overline{cw}$ is the mean number of terms across all collections. By analogy with $\mathrm{df}_c(t)$, $\mathrm{cf}_{\mathscr{C}}(t)$ is the number of collections containing t. The other terms are constants: $b = 0.4$, $\mathit{df\_base} = 50$, and $\mathit{df\_factor} = 150$. The score for a query is the mean of the per-term scores, $s_c(q) = \sum_{t \in \mathscr{T}_q} s_c(t)/|q|$.

CORI has been widely used and has proven effective in selection tasks based on TREC ad hoc data and Web data, although performance appears poorer when collection sizes are highly variable (D'Souza et al 2004; Si and Callan 2003a).

Extended CORI In practice, the frequency data used by CORI will be estimated from language models which may be of widely varying size both compared with each other and compared with the underlying collections. Later work (Si and Callan 2003a) suggested two extensions to the basic CORI algorithm, both of which make use of the N_m documents sampled from the collection in the course of building a language model. The first extension estimates $\mathrm{df}_c(t)$ from the model's $\mathrm{df}_m(t)$, and a scaling factor N_c/N_m; it also scales cw_c simlarly. The second extension uses both these modifications, and also scales *df_base* and *df_factor*. Since these variants expressly adjust for estimated size, they might be expected to perform better over heterogenous collections.

2.3 Lexicon inspection

Zobel's algorithm "I" (Zobel 1997) is a simple inner product without length normalisation:

$$s_c(q) = \sum_{t \in \mathscr{T}_q} w_q(t)\, w_c(t), \tag{3}$$

where the weights are based on analogues of term frequency and inverse document frequency: the query weight $w_q(t) = \log(\mathrm{tf}_q(t)+1)w(t)$; the collection weight $w_c(t) = \log(\mathrm{df}_c(t)+1)w(t)$; and, as an analogue of inverse document frequency, $w(t) = \log(N/\mathrm{df}_{\mathscr{C}}(t)+1)$. Here $\mathrm{df}_{\mathscr{C}}(t)$ describes the total number of documents containing a term t across all collections in $\mathscr{C}$. The function in Equation 3 is identical to MAX(0) and SUM(0) from vGlOSS (Equation 2), although in the case of vGlOSS weights are calculated with collection-specific data only and may vary, even for the same term, from collection to collection. In the experiments described below, this difference in weights results in a significant difference in performance.

One of Zobel's testbeds included collections which varied in size by three orders of magnitude, from 14 documents to 23,000. Although the testbed was in part randomly generated, this wide range of sizes suggests that algorithm "I" may be of general use.

2.4 CVV

Cue validity variance (CVV) (Yuwono and Lee 1997) is a server selection technique which weights terms according to their power to discriminate between collections. The technique begins by calculating the *cue validity* of each term t at each collection c, $\mathrm{CV}(t,c)$, which measures the extent to which t distinguishes c from other collections:

$$\mathrm{CV}(t,c) = \frac{\Pr(t|c)}{\Pr(t|c) + \Pr(t|\mathscr{C} \setminus c)}.$$

$\Pr(t|c)$ can be estimated by the relative frequency of t in c, $\mathrm{df}_c(t)/N_c$. $\Pr(t|\mathscr{C} \setminus c)$, the probability of t occurring in a "contrasting concept", is the relative frequency of t in all *other* collections: $\sum_{k\in\mathscr{C}\setminus c} \mathrm{df}_k(t) / \sum_{k\in\mathscr{C}\setminus c} N_k$.

(The implementation used in the experiments described below assigns $\mathrm{CV}(t,c) = 0$ when t is not present in any collection, meaning the term will not contribute to any collection's score.)

The cue validity of t at c gives an indication of how well t distinguishes documents in c from documents from all other collections. $\mathrm{CVV}(t)$, the cue validity variance of term t, is a measure of how useful a term is in distinguishing collections in $\mathscr{C}$ in general. $\mathrm{CVV}(t)$ is just the variance of $\mathrm{CV}(t,c)$. Finally, collections are scored according to the frequency of each query term, each weighted so that terms with more discriminative power are considered more important: $s_c(q) = \sum_{t\in\mathscr{T}_q} \mathrm{CVV}(t)\,\mathrm{df}_c(t)$.

2.5 Kullback-Leibler divergence

Kullback-Leibler (KL) divergence was suggested for server selection by Xu and Croft (1999) and interpreted in a language modelling framework Si et al (2002). The language modelling interpretation ranks each collection c according to its probability of being generated given a model of query q: i.e., $s_c(q) = \Pr(c|q)$. By Bayes's theorem therefore,

$$s_c(q) = \frac{\Pr(q|c)\Pr(c)}{\Pr(q)}. \tag{4}$$

Note that $\Pr(q)$, the prior probability of the query, does not depend on the collection and can therefore be ignored. Similarly, $\Pr(c)$, the prior probability of collection c, is normally considered constant and can also be ignored.

With these two simplifications, and assuming that terms occur independently, we can say $\Pr(q|c) = \prod_{t\in\mathscr{T}_q} \Pr(t|c)$. $\Pr(t|c)$ can be estimated from a model m. This can be shown to rank collections equivalently to KL divergence (Thomas 2008).

Smoothing can be used to account for infrequent terms. Si et al (2002) use a global model, m_g, which captures term occurrences across all collections. Term data from this global model is then combined with data from each collection in a modification of the above:

$$\Pr(q|c) = \prod_{t\in\mathscr{T}_q} \left(\lambda \Pr(t|c) + (1-\lambda)\Pr(t|m_g)\right). \tag{5}$$

The parameter λ controls the mixing and is typically 0.5 (Si and Callan 2003a; Si et al 2002). Note that when $\lambda = 1$, KL divergence ranks equivalently to bGlOSS.

Extended Kullback-Leibler divergence There is no explicit control for collection size in the method described above. Si and Callan's extended algorithm (Si and Callan 2003a) assumes that larger collections are more likely to be relevant and assigns the prior probability of collection c being relevant, $\Pr(c)$, according to collection size: $\Pr(c) = N_c / \sum_{i \in \mathscr{C}} N_i$. Otherwise, this variant is as above.

2.6 ReDDE

The relevant document distribution estimation method, or ReDDE, is also due to Si and Callan (2003b). As with the GlOSS family, ReDDE attempts to estimate the distribution of relevant documents across all collections based on a simple approximation of relevance. The algorithm computes an estimate of $|\textsc{rel}_c(q)|$, the number of documents in c which are relevant to the query q; this forms the basis of the eventual ranking.

$$|\widehat{\textsc{rel}_c(q)}| = \sum_{d \in \mathscr{D}_c} \Pr(\text{relevant}|d) \Pr(d|c) N_c$$

If m, the model of c, is representative, it is possible to estimate $\Pr(d|c)$ by assuming a uniform distribution: let $\Pr(d|c) = 1/N_m$. Still assuming that the model is representative, we can also estimate $\Pr(\text{relevant}|d)$ using the documents sampled when building the model, $\mathscr{D}_m$:

$$|\widehat{\textsc{rel}_c(q)}| = \frac{N_c}{N_m} \sum_{d \in \mathscr{D}_m} \Pr(\text{relevant}|d) \tag{6}$$

Note that this is the expected number of relevant documents; the formulation is similar to that used by bGlOSS (Equation 1). The remaining task is to estimate the probability that each document d in $\mathscr{D}_m$ is relevant to q.

ReDDE estimates this from a hypothetical ranking of all documents in all collections. Each of the top-ranked documents in this complete ("central") list has a fixed chance of relevance and contributes to the score of the collection it comes from:

$$\Pr(\text{relevant}|d) = \begin{cases} k & \text{if } \textsc{rank\_central}(d) < rN_{\text{all}} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $\textsc{rank\_central}(d)$ is the rank, over all documents in all collections, of d; and N_{all} is the total number of documents in all collections, $\sum_{c \in \mathscr{C}} N_c$. r and k are tunable parameters.

To compute this complete ranking would require a copy of every document in every collection, which is of course infeasible. $\textsc{rank\_central}(d)$ can instead itself be estimated using the samples collected in the course of building models for the collections. All documents sampled from all collections are indexed by the DIR tool itself, and ranked for each query by some effective method; the ranking of sample document d is $\textsc{rank\_sample}(d)$. With this sample rank, $\textsc{rank\_central}(d)$ can be estimated from the total number of documents ranked ahead of d: the intuition is that each document sampled from c stands for N_c/N_m documents in the complete ranking.

$$\widehat{\textsc{rank\_central}}(d) = \sum_{\substack{d' \in \mathscr{D}_m \\ \textsc{rank\_sample}(d') < \\ \textsc{rank\_sample}(d)}} \frac{N_{c_{d'}}}{N_{m_{d'}}}, \tag{8}$$

where $c_{d'}$ is the collection from which d' is drawn, and $m_{d'}$ the sample from which d' is drawn.

Substituting the estimates in Equations 7 and 8 into Equation 6 lets us estimate the total number of documents in each collection which are relevant to q. The final score is a normalised version of this total:

$$s_c(q) = \frac{|\widehat{\mathrm{REL}_c(q)}|}{\sum_{c' \in \mathscr{C}} |\widehat{\mathrm{REL}_{c'}(q)}|.}$$

2.7 Central-rank-based

The central-rank-based collection selection (CRCS) algorithms (Shokouhi 2007) are similar to ReDDE, and also make use of an index of sample documents. As for ReDDE, these documents are assumed to be representative of the collections they are drawn from; they are ranked for each query, and those collections which contribute highly-ranked sample documents are selected.

ReDDE awards collections a fixed score for each highly-ranked sample document (Equation 7). Shokouhi notes that this does not reflect the documents' likely utility: when ordered by an effective system, top-ranked documents are generally more useful than those of lower ranks. CRCS therefore allocates a rank-based, rather than fixed, score to each of the top γ sample documents. Shokouhi gives two variations. In the linear version, CRCS(l),

$$R(d) = \begin{cases} \gamma - \text{RANK\_SAMPLE}(d) & \text{if } \text{RANK\_SAMPLE}(d) < \gamma \\ 0 & \text{otherwise,} \end{cases}$$

where $\text{RANK\_SAMPLE}(d)$ is the rank of sample document d, as before. In the exponential version, CRCS(e),

$$R(d) = \begin{cases} \alpha\, e^{-\beta\, \text{RANK\_SAMPLE}(d)} & \text{if } \text{RANK\_SAMPLE}(d) < \gamma \\ 0 & \text{otherwise.} \end{cases}$$

These per-document scores are summed for each collection, in the same manner as ReDDE, and a final score is calculated with

$$s_c(q) = \frac{N_c}{N_{\max} N_m} \sum_{d \in \mathscr{D}_m} R(d).$$

$N_{\max}$ is the size of the largest collection, and is used to normalise the scores. Besides this normalisation, CRCS is very similar to ReDDE. Where ReDDE expressly rewards collections with a large number of high-ranked documents, however, CRCS rewards collections with both large numbers of documents and documents with particularly high ranks.

2.8 Decision theoretic models

Starting from assumptions regarding the independence of relevance judgements and the cost of retrieving documents, Fuhr (1999) derives a decision-theoretic formulation of the server selection problem. This method aims to minimise the total cost of retrieval, including costs of searching at each server, retrieving results, and presenting documents. Experiments used

100 collections of uniform size drawn from TREC data (Nottelmann and Fuhr 2003, 2004); it showed greater precision than CORI, but depends both on training data and on accurate cost information. It is not possible at present to do this for DIR collections in the most general case.

These methods seem to offer advantages when coping with the wide variety of servers found in personal metasearch, since they expressly model per-server attributes such as effectiveness and retrieval cost, and can potentially be used to retrieve more results from a server estimated to have a larger set of relevant documents.

The major limitation of the decision-theoretic framework in practical applications is the need to specify cost functions in advance. Our (admittedly anecdotal) observations of the use of personal metasearch tools suggest that:

- Cost functions for each personal metasearch installation would have to be specified individually because of variations in the sources being integrated, and in the nature of search activity by different individuals.
- The appropriate cost functions are likely to change from task to task and even from query to query. Consider for example the following search activities plausibly undertaken by a single individual: (a) locating the YouTube home page, (b) finding cheap flights to Boston, (c) gathering comprehensive information to provide investment advice to a “high net worth” client, or (d) locating their employer’s standard investment advice contract.

It is our belief that individual users would find it difficult to specify cost functions and a burden to do so. In the future it may be possible to provide personal metasearch tools with effective query classification capabilities and to enable the tool to adaptively learn appropriate cost functions from user behaviour and feedback.

Following similar lines, Wu and Crestani (2002) considered the decision-theoretic model “too abstract”, and have instead suggested a simpler “multi-objective” model. The multi-objective method has not been compared with other selection methods, although absent an agreed set of costs and weights it is not clear how this could be done.

2.9 Other methods

As well as those above, a large number of server selection methods have been described in the literature (including contributions from Craswell et al (2000); Hawking and Thistlewaite (1999); Larson (2003); Rasolofo et al (2001); Shen and Lee (2002); Si and Callan (2005); Wu et al (2001), and many others). In our experiments we have retricted ourselves to those methods which seem most generally applicable: those which make few demands on individual servers, have few or no parameters that require tuning, and do not need bootstrapping with sample queries or known-good relevance judgements.

3 Selection experiments

In general, the performance of the selection techniques above has been evaluated using TREC ad hoc or Web Track data—often partitioned with an eye to producing collections of approximately equal size (Table 1). The availability of extensive relevance judgments for the TREC ad hoc collections has made them the basis for most DIR testbeds. However, such

		Size (k docs)		
Testbed	Colls.	Range	Mean	Partitions
TREC ad hoc:				
TREC123-17-bysource	17	7–226	64	By source and date
UBC-100, uniform	100	1–40	11	Sequential and by source, for size
relevant	62	1–243	17	From uniform, for density
nonrelevant	83	1–226	13	From uniform, for density
representative	62	1–232	17	From uniform, for density
SYM-236	236	$<$ 1–8	3	By source and date
UDC-236	236	3–3	5	Sequential, for size
TREC4-kmeans	100	$<$ 1–83	6	By topic
TREC123-10col	10	18–263	108	Arbitrary
Web:				
WT2g	951	$<$ 1–8	$<$ 1	By server
WT10g	11590	1–26	$<$ 1	By server
.GOV	7792	$<$ 1–16	$<$ 1	By server
.GOV2	17186	$<$ 1–717	1	By server
Other:				
Personal metasearch	6	1–1200	234	By source

Table 1 A selection of DIR testbeds. Figures for Web testbeds assume that documents are divided according to Web server.

collections do not resemble those likely to be used in working DIR applications: the Web is extremely large and highly diverse, while TREC collections are all of one data type.

The experiments reported below consider the likely performance of these techniques in personal metasearch, where collections are likely to vary in data type and especially in size. We consider three questions:

1. How does selection performance compare across methods, and across queries?
2. How does selection performance vary, given samples and size estimates of different quality?
3. Which, if any, of the methods is appropriate for selection in such an environment?

We also examine the performance of these techniques on common testbeds derived from TREC data. Although these testbeds do not correspond to any particular application of selection techniques, we are able to confirm the trends seen on our personal metasearch testbed.

Previous work has presented performance figures for CORI and extensions, KL divergence and extension, and ReDDE on a common testbed (Si and Callan 2003a). The implementation of these algorithms used in these experiments was validated against this data. One further method is included as a baseline: the "random" method ranks each server randomly for each query, without using any information about each server.

3.1 Personal metasearch testbed

The first experiments reported here use six collections. These are representative of the range of resources which are likely to be used in personal metasearch applications: sizes range over three orders of magnitude, data types are varied, and topic areas range from the very focussed to the very broad. Each collection is mostly English-language. None are on the scale of the largest likely collections, such as the Web or Dialog. However, rather than ask each user's tool to estimate the size of (for example) the public Web, it is likely that a broker would use pre-computed characteristics for very large collections. The collections used here span the likely size range of local, private, or enterprise collections, where pre-computed data is infeasible.

- The “calendar” collection contains 1049 appointments from a calendar application. Documents are typically short sentence fragments.
- The “zsh-list” and “procmail” collections are archives of public mailing lists discussing narrow technical topics, and have around 9000 and 24,000 documents respectively.
- The “email” collection includes around 25,000 documents from a personal email archive with much broader topics.
- “WSJ” incorporates around 99,000 articles from several years of the Wall Street Journal.
- “.GOV”, the largest collection used here, is a 1.2 million page crawl of Web hosts in the `.gov` domain.

Compared with other testbeds, the personal metasearch testbed used here has fewer collections, a much larger range of collection sizes (except as compared to one testbed from Zobel (1997)), and a much wider variety of document topics and sources.

3.2 Queries

Queries for these experiments were created with the intention of representing the variety of topics covered in the testbed. They were generated in a number of ways and formed six sets of 20 queries, each based on one of the six collections.

Past work has often used long queries; for example, Xu and Croft (1999) report a series of experiments with a mean 34.5 terms per query. This is evidently much larger than the 1.7–2.6 terms typical of queries to web search services (Beitzel et al 2004; Jansen et al 2000; Silverstein et al 1999; Spink et al 2002; Zhang and Moffat 2006), and the queries used here were accordingly kept short.

- Subject and location fields were extracted from calendar entries, stopwords were removed, and 20 of these reduced fields selected at random made up the first set of queries. Queries had a mean length of 1.95 terms, with a standard deviation of 0.59 terms.
  PADRE (Hawking et al 2000) was used to retrieve the top five documents from each collection for each of these 20 queries, and relevance judgements were conducted manually for each returned document. (This is a pooling technique in the manner of Harman (2005), with depth five.) The vast majority of relevant documents came from the calendar collection, with a number also from the email collection; since these two collections overlap somewhat in subject matter, and many pieces of email also contain dates and places, this is expected.
- A similar process was used to construct queries from subject lines in the email collection. Queries had a mean length of two terms and a standard deviation of 0.55 terms, and top results were manually judged. Most relevant documents were drawn from the email collection, with some from the calendar collection.
- The procmail and zsh-list collections represent mailing lists on narrow technical subjects. The same process was used to construct queries based upon these two collections, but there was no topical overlap; relevant documents from each set of queries came only from the original collection.
  As may be expected from more focussed sets of documents, these two sets of queries were longer on average (mean 3.90 terms for procmail, 3.30 for zsh-list; standard deviation 3.47 terms and 1.23 terms).

- The “topic” field of twenty topics from the TREC ad hoc track, all of which had at least one relevant document in the WSJ collection, were used as a fifth set of queries. Relevance judgements were made over the top results from the calendar, email, procmail, zsh-list, and .GOV collections; relevance judgements from TREC were used for documents from the WSJ collection. These queries were rather long (mean 4.20 terms, standard deviation 2.52 terms).
- The “topic” field of twenty topics from the TREC Web track made up the final set of queries, and relevance judgements were made in a similar way to the previous set. This set had a mean length of 3.30 terms and standard deviation of 1.23 terms.

Queries in other test collections have typically been generated manually. The first four sets of queries used in these experiments were instead created automatically from identifying fields; the intention was to mimic the terms a user might type if they were familiar with the contents of each collection. It is not clear how well these automatically-generated queries match the queries DIR users would really type, but nor is this clear for the manually-generated queries of the last two sets.

Our testbed also differs in the density of relevant documents. In most previous work, collections are of approximately unform document density, so that for example a collection ten times larger will have about ten times as many relevant documents. The testbed used here instead is constructed under the assumption that collections are approximately equally useful, in the long run, and therefore collections have approximately the same number of relevant documents regardless of size. Over all 120 queries, the number of relevant documents per collection varied from 45 for the calendar collection (1049 documents) to 101 for WSJ (99,000 documents). This difference in density means algorithms should not simply select the largest collection.

3.3 Models and size estimates

The selection techniques used in these experiments rely on language models and size estimates of the collections involved. This data was drawn from three sources.

As a baseline, models were built using all documents in each collection, and “estimates” of collection size were entirely accurate. Although not possible in practice, this provides a best case for comparison. Two further sets of models represented the best-performing and the most-used of the present techniques. These are discussed further in Section 4.2 below.

3.4 Measures

Let each collection c have an associated merit, denoted $\text{MERIT}_c(q)$, which is a measure of how good a choice c is for this query. $\mathscr{R}_n$ is the proportion of this merit captured by the n top-ranked collections (Gravano and García-Molina 1995):

$$\mathscr{R}_n = \frac{\sum_{i=1\ldots n} \text{MERIT}_{est_i}(q)}{\sum_{i=1\ldots n} \text{MERIT}_{optimal_i}(q)},$$

where est_i is the i’th ranked collection in the estimated ranking and $optimal_i$ is the i’th ranked collection in the optimal ranking. $\mathscr{R}_n$ ranges from 0, meaning there is no merit in the first n collections selected, to 1, meaning the first n collections selected are as good as possible.

$\text{MERIT}_c(q)$, the (real) utility of a collection c for a query q, has been defined in a number of ways. Gravano and García-Molina (1995) define MERIT in terms of the total similarity between c and q: the sum, over all documents $d \in \mathscr{D}_c$, of the similarity between d and q. If servers work on a vector-space model, this is an approximation of what an effective server might retrieve in response to a question. This has the effect of including an aspect of server performance, but as well as assuming a vector-space model at the servers it assumes that documents similar to the query are likely to be useful. This definition has also been used by French et al (1998) and French et al (1999) (who also use the relevance-based definition discussed below), and Gravano et al (1999).

An alternative definition of MERIT simply counts the number of relevant documents at each collection, $\text{MERIT}_c(q) = |\text{REL}_c(q)|$. Since the relevant documents in a collection may or may not be returned by a server, this measure is independent of server performance. It has been commonly used for this reason (French et al 1998, 1999; Hawking and Thistlewaite 1999; Si and Callan 2003a,b), and it is this definition we use in the results reported below. (Where servers vary in the cost of retrieving results, this may not be appropriate. A measure of selection quality that properly accounts for this, but which can be compared across testbeds, remains for future work.)

For the testbed used here, a random ordering has an expected $\mathscr{R}_n$ of around $n/6$ since there are few queries where more than one collection holds relevant documents. The same is true for a size-based ordering, since each collection holds around the same number of relevant documents. The best ordering, ranking servers according to the number of relevant documents they manage, has a score of $\mathscr{R}_n = 1$ for all n.

We chose to use the $\mathscr{R}_n$ measure in order to focus on the effect of variations in collection size on selection effectiveness, using the measure most commonly used in previous studies of the algorithms we have evaluated. We wanted to see whether previous observations on these algorithms could be confirmed over a significantly wider range of collection sizes than previously studied, and in a simulation of a realistic DIR application.

We recognize that $\mathscr{R}_n$ may not correlate well with searcher satisfaction (the gold standard) and that extensive user observation is needed to choose the evaluation measure (and methodology) and to determine to what extent the choice is dependent upon the query or search task. Readers are referred to Thomas and Hawking (2008) for the results of a small scale pilot study of real personal metasearch users, designed to shed light on whether user judgments tend to confirm testbed evaluations of selection methods.

4 Results

We discuss the results of our experiments when selection algorithms were given "perfect", entirely correct, data on each collection; and when they were given more realistic data estimated from document samples. We also report the observed correlation between data quality and performance of two promising selection algorithms.

4.1 Selection with perfect models and sizes

The first experiment provided a baseline for later comparisons. Each method was run with parameters set as described by the original authors; each method was also allowed a perfectly accurate model built from all documents in the collection, and a perfectly accurate size "estimate". In the case of tied scores $s_c(q)$, collections were ordered arbitrarily.

The bGlOSS algorithm has no tunable parameters. Other algorithms used the following:

- vGlOSS used threshold $l = 0$, so any documents at all similar to the query were considered interesting. Weights were assigned according to term frequency and inverse document frequency (Gravano et al 1999):

$$\begin{aligned} w_q(t) &= \mathrm{tf}_q(t)\log(N_c/\,\mathrm{df}_c(t)) \\ \text{and } w_c(t) &= \mathrm{tf}_c(t)\log(N_c/\,\mathrm{df}_c(t)). \end{aligned}$$

  Any query terms which did not appear at all in c were assigned a collection weight of zero. Weights were normalised by dividing through by the Euclidean norm.
- CORI used $b = 0.4$, *df_base* $= 50$, and *df_factor* $= 150$ (Callan et al 1995).
- Zobel's algorithm "I" used weights as described in Section 2.3. If a query term did not feature in any collection (so $\mathrm{df}_{\mathscr{C}}(t) = 0$), the weight for that term $w(t)$ was assigned zero.
- The KL divergence implementation used mixes in a global language model as in Equation 5. The mixing parameter λ was set to 0.5, following Si and Callan (2003a).
- ReDDE is considered robust with values of r from 0.002 to 0.005 (the top 0.2% to 0.5% of all documents considered relevant) (Si and Callan 2003b); the experiments here used $r = 0.003$. Note that the other parameter k, the score given to a collection for each highly-ranked sample document, is irrelevant due to the normalisation step at the end of the algorithm.
  Rather than correct models, ReDDE used a "sample" index of all documents in all collections. Documents were ranked for each query by PADRE (Hawking et al 2000), which uses a slight variant of Okapi BM25 (Robertson et al 1994).
- Both CRCS methods give no points to any document not in the top 50 (so $\gamma = 50$). For the exponential scoring of CRCS(e), $\alpha = 1.2$ and $\beta = 0.28$ (Shokouhi 2007). (The originally published figure $\beta = 2.8$ was in error.[1])

Figure 1 summarises the performance of each method given this baseline data. (Note that lines are interpolated for convenience; n must be integral.) Since models are built from all documents, $N_m = N_c$ and so the three variants of CORI are identical.

In this case the KL divergence method is clearly the best-performing of the techniques tested; $\mathscr{R}_n$ scores from this technique are significantly higher than those from any other method for $n = 1$ to 5 (one-sided t test, $\alpha = 0.05$). vGlOSS, on the other hand, does relatively poorly and has significantly lower scores than all other methods for $n = 2$ or 3. bGlOSS, CORI and variants, extended KL divergence, and CRCS variants also perform well.

4.1.1 Correlation with size-based ranking

French et al. noted a very strong correlation between collection ranks from vGlOSS and ranks based on collection size alone (French et al 1999). Collection sizes in DIR applications may range from a few hundred documents (for small email archives, for example) to billions of documents (for the Web or Dialog). If smaller collections contain useful documents, any tendency to select larger collections could result in poor overall performance. A second set of experiments investigated this bias.

Figure 2 plots the correlation between rankings based on server scores $s_c(q)$ and those based on collection size N_c, over all 120 queries for each selection method. Spearman's rank

[1] Personal communication from Milad Shokouhi.

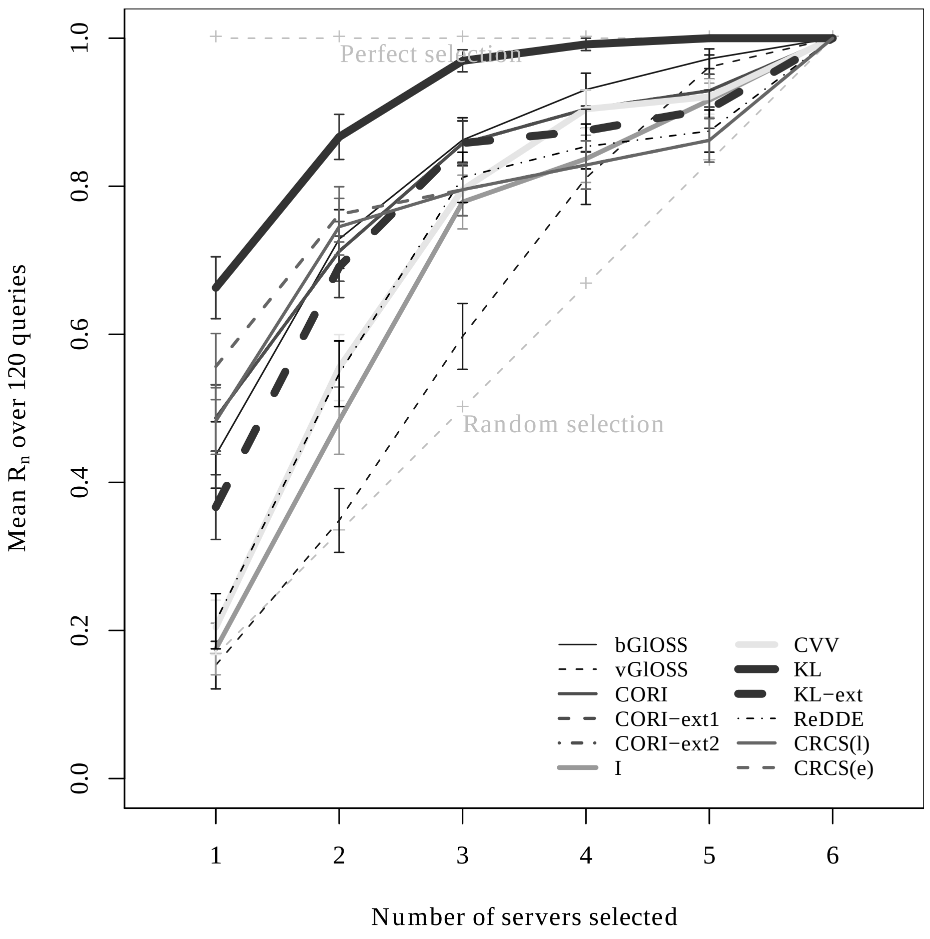


Fig. 1 Performance of server selection methods, with correct sizes and models. Bars are $\pm$ one standard error.

correlation coefficient r_s is used and takes values in $[-1,1]$. A correlation of -1 indicates collections are ranked smallest to largest, 0 indicates there is no correlation between N_c and $s_c(q)$, and a correlation of 1 indicates that the collections are ordered largest to smallest.

It is clear that most methods do in fact correlate highly with a size-based ranking. From Figure 2, it seems only vGlOSS and KL divergence are relatively weakly correlated, while bGlOSS, CORI and extensions, "I", CVV, extended KL divergence, and the CRCS variants seem prone to ranking large collections highly regardless of the query. For example, bGlOSS ranks .GOV highest for 80 of the 120 test queries despite there only being 20 queries for which this collection actually contains the largest number of relevant documents. Most other methods are similar, ranking .GOV first for between 58 and 119 of the 120 queries. vGlOSS, which is less prone to ranking by size, ranks .GOV first for only 4 of the 120 queries, and KL divergence does the same for only 37.

A straightforward examination of the methods demonstrates why large collections are favoured. bGlOSS and extended KL divergence include an explicit adjustment for N_c, the size of the collection; CORI and extensions, "I", and CVV use either df_c or tf_c, which may

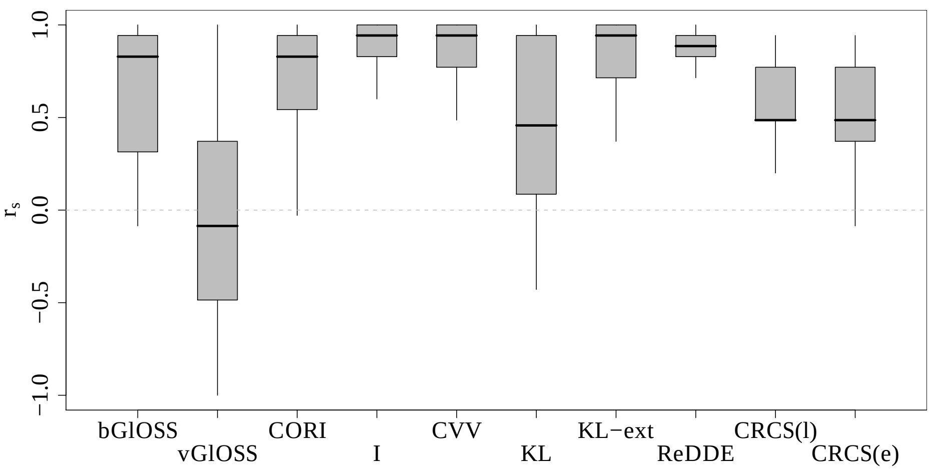


Fig. 2 Correlation (Spearman's r_s) of collection rankings with rankings based on size alone, over 120 queries. CORI-ext1 and -ext2 are identical to CORI.

be expected to correlate highly with collection size. Although ReDDE and CRCS do not explicitly adjust for collection size in this instance (since $N_m = N_c$), the sample index will be dominated by documents from larger collections. In both vGlOSS and KL divergence frequency information is normalised on a collection-by-collection basis and larger collections are not favoured.

For example, consider that subset of queries for which .GOV (the largest collection, with 1.2M documents) is the best answer, and that subset for which calendar (the smallest, with 1k documents) is the best answer. Performance on the first set, where success corresponds to choosing the largest collection, is near-perfect for most methods: this is expected, since most methods are highly correlated with a size-based ranking which would rank .GOV first. vGlOSS, however, which is less strongly correlated, is only significantly better than random selection for $n = 5$. The situation is reversed for the second subset of queries, where the task is to choose the smallest collection (Figure 3); only KL divergence and vGlOSS, the two methods least prone to size-based ranking, are significantly better than random selection for any n. Other methods tend to rank the small calendar collection low despite it being the best choice in these instances.

French et al (1999) report much higher correlation between a size-based ranking and vGlOSS (mean $r_s = 0.97$) than is seen here (mean $r_s = -0.06$). This may be due to differences in the distribution of relevant documents: French et al. observe that the larger collections in their testbed tended to have more relevant documents, which would lead them to be highly ranked by any effective selection technique, and a relevance-based ranking of collections in their testbed correlates moderately well with a size-based ranking ($r_s = 0.54$ on average, and $r_s \geq 0$ for all queries). The distribution of relevant documents in the testbed used here, however, is much more uniform. A difference in calculating the weights $w_c(t)$ and $w_q(t)$, for example by normalising across all collections rather than across a single collection at a time, could also give rise to this disagreement. Data from French et al. and these experiments are in broad agreement however on the correlation between CORI and a size-based ranking, and this correlation was also mooted by D'Souza et al (2004).

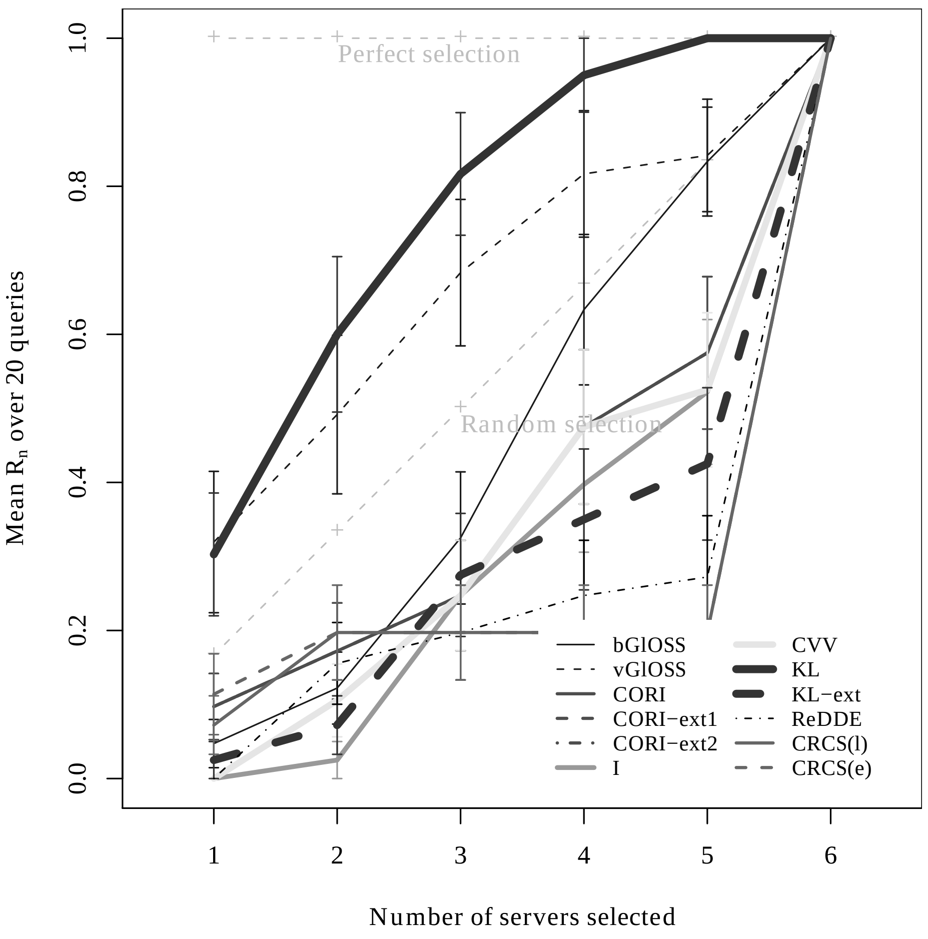


Fig. 3 Queries for which calendar (1k documents) is the best collection.

4.2 Selection with samples and estimated sizes

The experiments described above use perfect models, or complete samples, and exact size "estimates". This provides a baseline, but is of course impractical in most real DIR applications; a third set of experiments therefore considered selection performance with language models and size estimates built from sampled documents.

Two sets of samples and size estimates were used. The first set used samples of 300 documents per collection generated by the multiple queries sampler (Thomas and Hawking 2007) and size estimates from multiple capture-recapture (Shokouhi et al 2006)); these were the best-performing sampler and size estimator in previous experiments (Thomas and Hawking 2007; Thomas 2008). (The multiple queries sampler could only sample 200 documents from the calendar collection.) The second used samples of 300 documents generated by query-based sampling (Callan et al 1999) and size estimates from capture-recapture (Liu et al 2001), which have been commonly used in previous work (Hawking and Thomas 2005; Shokouhi et al 2007; Si and Callan 2003a,b).

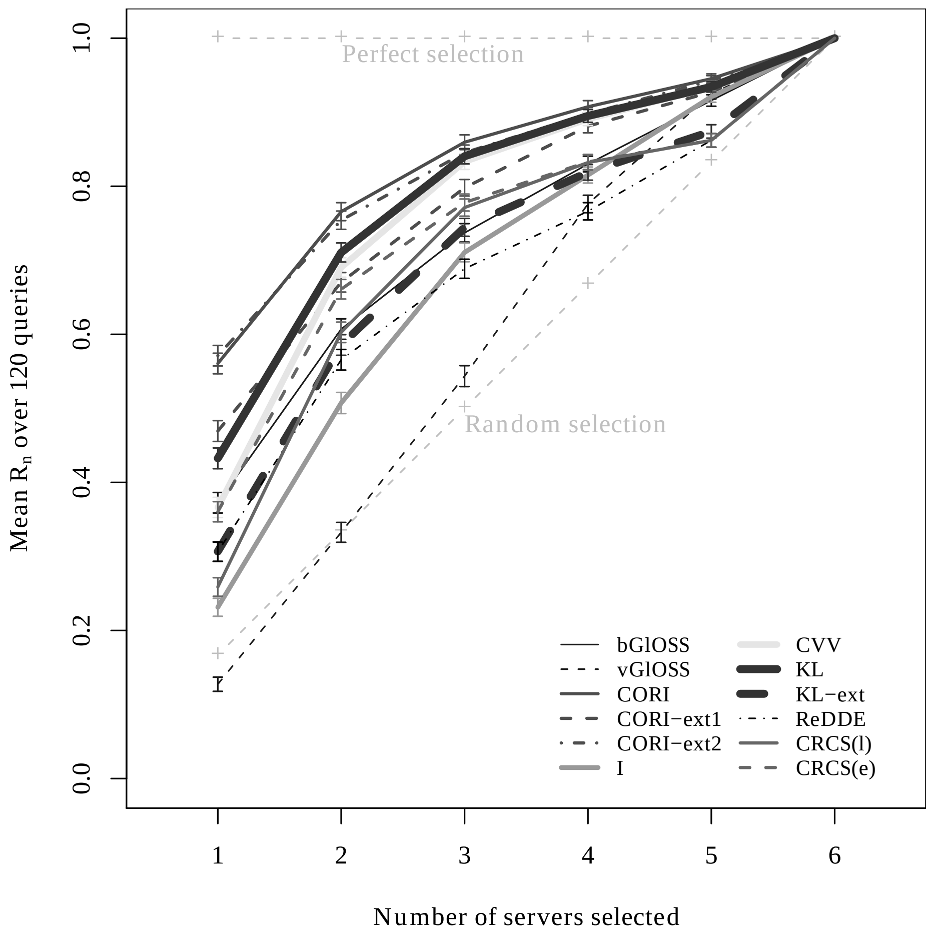


Fig. 4 Performance of server selection methods, with one to six servers selected; models built from documents sampled with multiple queries; and collection sizes estimated with multiple capture-recapture. Bars are ± one standard error.

The choice of a 300-document sample size follows previous work (Callan et al 1999; Hawking and Thomas 2005; Nottelmann and Fuhr 2003; Si et al 2002; Si and Callan 2003a,b) but is essentially arbitrary. A more sophisticated approach may attempt to sample documents until, for example, the learned language model appears stable (Baillie et al 2006); on these measures, and on the testbed used here, early experiments suggest this is at about 300–400 documents per collection.

Selection methods needed only minor adaptations to operate with models built from sampled documents. Term and document frequencies for bGlOSS, vGlOSS, CORI, "I", CVV, and KL divergence were estimated from models and scaled according to the ratio $\hat{N}_c/N_m$. ReDDE, extended CORI, and the two CRCS variants explicitly adjust for differences in model size. Otherwise, parameters were as described in Section 4.1.

Most methods performed worse with these lower-quality models than with the perfect models of Section 4.1 for at least some values of n. KL divergence was previously the best-performing method; with models built from 300-document samples from the multiple

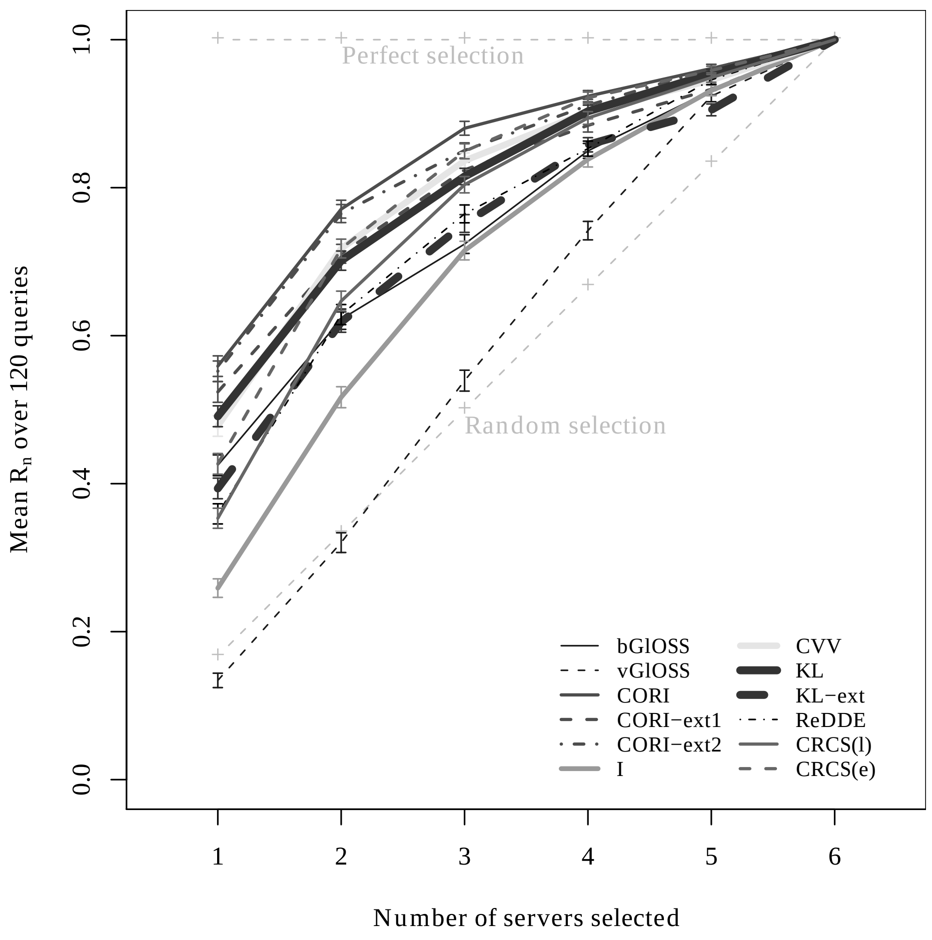


Fig. 5 Performance of server selection methods, with one to six servers selected; models built from documents sampled with query-based sampling; and collection sizes estimated with capture-recapture. Bars are ± one standard error.

queries sampler and size estimates from multiple capture-recapture, it is significantly worse for $n = 1$ to 5 (one-tailed Wilcoxon test, $\alpha = 0.05$). bGlOSS, another method which performed well with correct data, is significantly worse for $n = 1$ and 3 to 5; extended KL divergence is significantly worse for $n = 1$ to 5; and both CRCS variants are significantly worse for $n = 1$ and 2. CORI, which performed relatively well in earlier experiments, is not significantly worse for any n.

Similar trends held for the second set of runs, which used data from the query-based sampler and (single) capture-recapture. Again KL divergence, its extension, bGlOSS, and CRCS were significantly worse for several values of n, although KL divergence and the CRCS variants still performed well; CORI was not significantly worse than before at any point, and in fact was significantly better for some n. Taken with the observations on size--based ranking above, this suggests that KL divergence is an appropriate server selection method when collections vary greatly. CORI, CRCS(l), or CRCS(e) may also be appropriate, although they do a poorer job of selecting smaller collections when needed. Choice of selection method may also be influenced by the quality of samples and size estimates.

The extensions to CORI, "CORI-ext1" and "CORI-ext2" in Figures 4 and 5, have little effect on CORI's overall performance. This is consistent with earlier results from Si and Callan (2003a) and Hawking and Thomas (2005).

4.3 Measures of model quality

Results from earlier experiments suggested that the quality of a language model has a significant effect on the performance of selection algorithms. A further set of experiments investigated this connection, and asked: is there a correlation between the quality of a language model and the performance of these algorithms? If so, any technique which improves the quality of models should have an impact in improved selection as well as in any other applications.

Ninety sets of samples were generated, ten each of 100–900 documents (in steps of 100), chosen randomly from each collection. Models of each collection were built from each of these ninety sets, and each model's quality was calculated according to three measures: ctf ratio (Callan et al 1999), which measures the proportion of term occurrences in the collection which are accounted for by terms in a model; Spearman's r_s; and Kullback-Leibler divergence (D_{KL}). The mean measure, over each set of six collections, was treated as an indication of the quality of that set of samples. In each case, the mean measure varied from set to set but improved as more documents were included.

To investigate the correlation between these quality measures and the performance of server selection methods, the CORI and KL divergence algorithms—two of the most promising selection algorithms from earlier experiments—were run with each set of models over the same 120 queries as before. (Size "estimates" in these runs were correct, to isolate the effect of model quality.) Results are illustrated in Figure 6, which plots the performance of the algorithm (measured as mean $\mathscr{R}_1$, or the mean recall at one collection selected over all 120 queries) against ctf ratio. Similar results were seen for r_s and D_{KL}.

All three measures—ctf ratio, r_s, and D_{KL}—correlate highly with $\mathscr{R}_1$ for both CORI and KL divergence, with absolute coefficients of correlation (Spearman's r_s) of between 0.64 and 0.82 ($p \ll 0.05$ in each case). This suggests that selection performance, at least for the CORI and Kullback-Leibler algorithms, will increase if language models are improved and degrade if they are made worse. Further, all three measures are useful; all are good predictors of selection performance.

4.4 Other testbeds

The results presented above are derived from our testbed, which represents a particular application of distributed IR techniques. Our final set of experiments investigated the performance of server selection algorithms on alternative testbeds.

We consider four testbeds generated from TREC ad-hoc documents and queries. Although these testbeds are essentially arbitrary and do not represent any particular application, the availability of relevance judgements means they have been widely used in earlier selection experiments (for example Powell et al (2000); Powell and French (2003); Si and Callan (2003b,c); Shokouhi (2007)). Each is comprised of the same documents, divided to give four different distributions of those which are relevant.

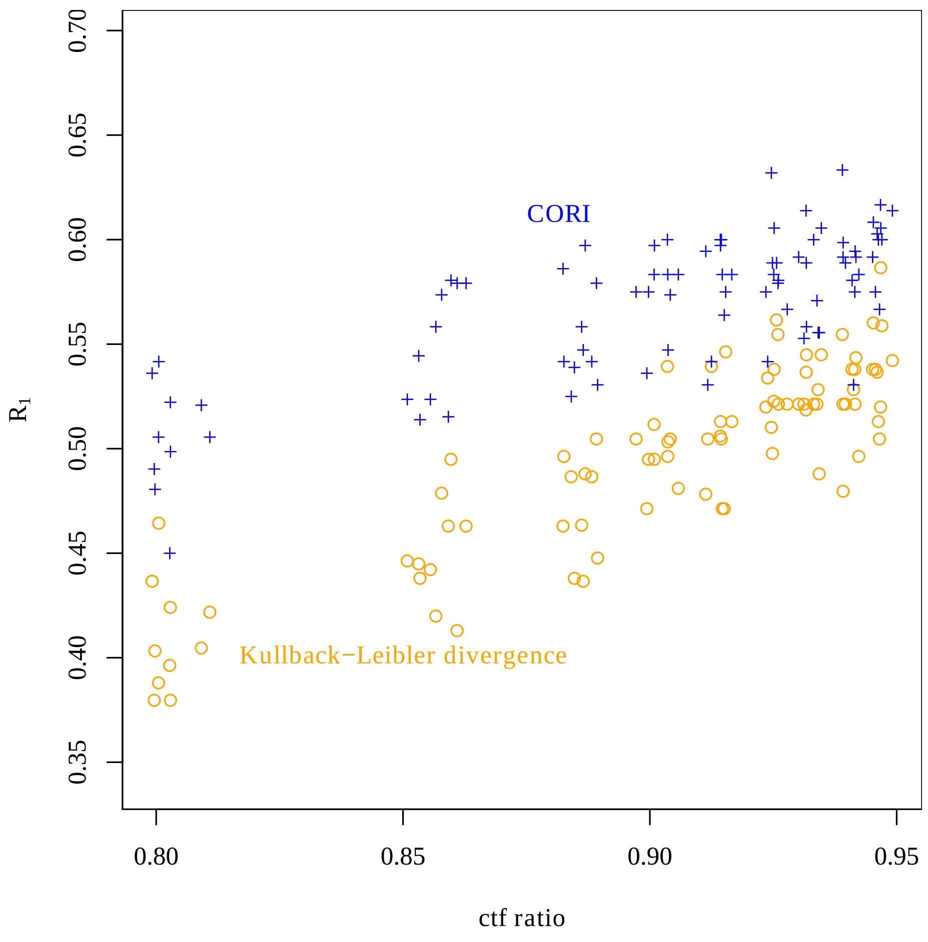
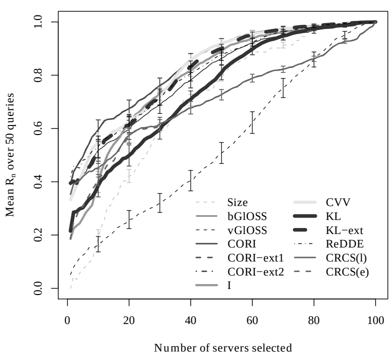


Fig. 6 Correlation between mean ctf ratio and mean selection performance, over ninety sets of samples across each collection in the personal metasearch testbed.

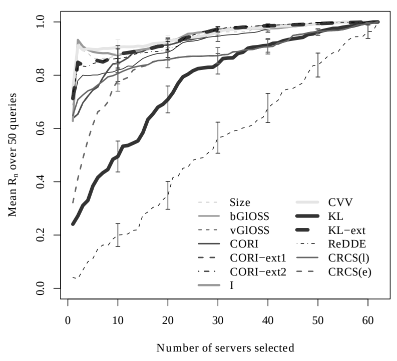
- The "uniform" testbed takes documents from TREC CDs 1, 2, and 3 and divides them amongst 100 collections according to their original source and date. Collection sizes are kept relatively homogenous, with a mean 10,782 documents and no single collection larger than 39,713 documents.

Other testbeds are derived from the uniform testbed by aggregating some smaller collections into new, larger ones.

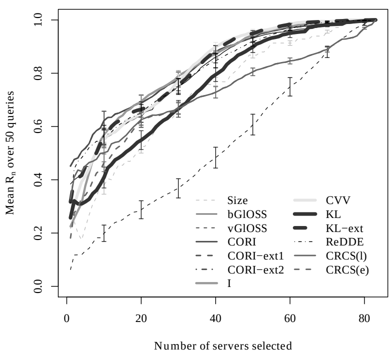
- In the "relevant" testbed, documents from the Associated Press which comprised 24 collections in the uniform testbed are merged into a single collection; documents from the Wall Street Journal are similarly merged from 16 collections to one. These two collections are much larger than the remaining 60, and feature both many more relevant documents and a higher density of relevant documents.
- The "nonrelevant" testbed is built similarly, but the two large collections comprise documents from the Federal Register and the Department of Energy. These new collections are again relatively large but have a lower density of relevant documents.



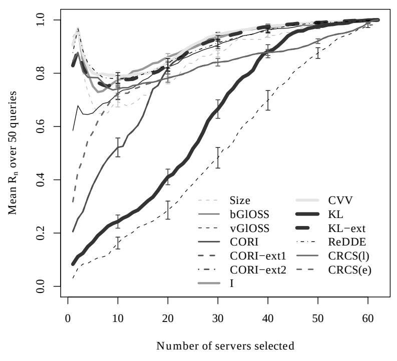
(a) “Uniform” testbed



(b) “Relevant” testbed



(c) “Nonrelevant” testbed



(d) “Representative” testbed

Fig. 7 Performance of server selection methods on TREC data, with correct sizes and models. Bars are $\pm$ one standard error.

– Finally, in the “representative” testbed two large collections are built each with 20% of the original, smaller, collections; the collections to be merged are taken uniformly from the original 100. These new collections are again an order of magnitude larger than the remaining 60, but the density of relevant documents is similar.

Queries 51 to 100, and the associated TREC relevance judgements, were used in each case. Results for each testbed are illustrated in Figure 7.

Results are clearly different with the TREC testbeds than with the personal metasearch testbed; further, the results from the uniform and nonrelevant testbeds are similar as are those from the relevant and representative.

This observation is explained by characteristics of the testbeds. The relevant testbed, for example, is constructed such that the large collections are more often the best choice for any query; the two largest collections have around 120 relevant documents per query, on average, while the remaining collections typically have four or fewer, and there is a strong correlation between size and mean number of relevant documents (Spearman’s $\rho = 0.95$, $p \ll 0.01$). In the representative testbed, although the density of relevant documents is roughly equal the absolute number is not and again the larger collections are a better choice (and here again $\rho = 0.95$, $p \ll 0.01$). In these cases, selection techniques which correlate well with a size-based ranking do well and those which do not (vGlOSS and Kullback-Leibler) fare poorly. A simple size-based ranking is almost indistinguishable from the best of the algorithms.

The correlation between size and number of relevant documents still holds in the nonrelevant testbed, but is much less pronounced ($\rho = 0.55$, $p \ll 0.01$), and there is no significant correlation in the uniform testbed.

4.5 Summary of results

Selection experiments based on perfectly accurate models and size estimates demonstrated that KL divergence performed well, as to a lesser extent did CORI and variants, extended KL divergence, and bGlOSS. Many methods are prone to ranking larger collections highly, regardless of their usefulness for any particular query; only KL divergence and vGlOSS were found to be able to select small collections better than chance.

With less accurate models and size estimates, almost all methods were significantly poorer. CORI, which is less affected by inaccuracies in the model, and KL divergence, which is less prone to size-based ranking, seem appropriate choices for selection algorithms in real-world tools.

Comparing results from the TREC testbed with those from the personal metasearch testbed, it is clear the measured performance of selection algorithms varies greatly depending on the characteristics of the testbed used. In particular, the density of relevant documents in each collection is significant. For the personal metasearch application, or others where size varies greatly but the larger collections are not always a good choice, this means conclusions based upon earlier evaluations may not apply.

5 Discussion and conclusions

Personal metasearch is a novel but real IR application, where a broker provides a single search interface over all of a user's online resources. Since many disparate sources must be included, there is no feasible alternative to distributed IR (DIR) techniques. The application of these techniques to the personal metasearch task is complex, and at this stage little understood.

Most DIR tools, including personal metasearch tools, need a query-time process of server selection to identify which servers may be useful. This process can reduce costs, especially if servers charge for access, can improve reliability and efficiency, and if done well enough can improve results even compared to a single index. Many selection methods have been suggested; twelve of the most generally useful have been tested here, in a testbed with widely varied collection sizes which is representative of personal metasearch applications.

Overall, the CORI algorithm is promising when collection sizes vary, and is robust to poorer quality models; Kullback-Leibler divergence is also robust and is much more likely to select smaller collections when appropriate. Experiments in this paper have shown several further trends, and corresponding directions for further work.

In the case of personal metasearch, collections will vary greatly in size from the fairly small (individual folders, addressbooks, calendars) to the very large (the web, Dialog). However, we do not expect the size of a collection to correlate with the number of times it is useful: to take a simple example, an email archive will be a good source for a large proportion of queries despite being several orders of magnitude smaller than the largest collections. This is not true of some other DIR applications, and is certainly not assumed by the common TREC-based testbeds.

Correcting for collection size substantially reduces performance when smaller collections are required, and such corrections are unlikely to improve performance overall unless most user needs are in fact answered well by larger collections. A bias towards large (or indeed small) collections is therefore undesirable in personal metasearch, but most of the methods tested exhibit a strong bias (Figure 2). A working broker in this application would need to choose small collections when appropriate.

Personal metasearch is also distinguished by its coverage of a wide variety of data types; work to date has generally considered the small range of data in TREC-based testbeds. In most cases, the methods tested here perform more poorly on our application-specific testbed than on the "relevant" and "representative" TREC testbeds, and this is especially true when few servers are selected.

Sampling and selection methods currently take no account of the size, type, genre or structure of documents within collections. These aspects could potentially be exploited. For example, if a searcher is known to be seeking movie reviews or media releases, type- or genre-aware selection techniques could easily determine that calendars, staff databases, dictionaries or source code repositories are poor candidates for inclusion. Investigation of these dimensions of sampling and selection is left for future work.

The quality of language models and collection size estimates used in selection tasks has been shown to be important: almost all methods suffered a drop in performance when given models inferred from sampled documents. Further, for the Kullback-Leibler divergence and CORI techniques there is a strong correlation between measures of model quality and early performance. This suggests that techniques for building representative models will be of importance to any real-world tool using any of the most promising selection techniques.

Unfortunately current size estimation techniques are impractical for collections on the scale encountered by personal metasearch tools (up to 10^6 documents in our testbeds, and several orders of magnitude larger in practice), and sampling techniques and hence language models are poor at many scales—compare for example Figures 4 and 5 with Figure 1.

Although it is not yet clear what impact selection may have on final user satisfaction it appears that these DIR techniques, tested largely on artificial testbeds, are inappropriate for the real application of personal metasearch. Techniques are needed which are better able to cope with wide varieties of collection size, poor quality language models, and a variety of subject matter.

The decision-theoretic or multi-objective frameworks are theoretically general enough to accommodate this problem but seem too difficult to apply in a practical broker, due to the dependence of cost functions not only on users and collections but on individual queries; it is so far impractical to specify them on this basis. Much more observation is needed of personal metasearch in practice to identify modes of use which could be associated with particular clusters of cost functions. To date it is not known how well these methods work when given approximate input such as is derived from sampling.

Finally, we note that a personal metasearch tool receiving frequent use is in an ideal position to observe its owner's behaviour. If it included an effective task/query classifier, it could almost certainly learn to improve its performance over time, and would certainly provide an insight into effective techniques for server selection as well as other DIR tasks. A broker with the ability to learn from user interactions may also be able to adapt its selection algorithm to the characteristics of the collections being searched. Our work in these directions is continuing.

Acknowledgements We thank the anonymous reviewers for their helpful comments.

References

Avrahami TT, Yau L, Si L, Callan J (2006) The FedLemur project: Federated search in the real world. JASIST 57(3):347–358

Baillie M, Azzopardi L, Crestani F (2006) Adaptive query-based sampling of distributed collections. In: Proc. SPIRE, no. 4209 in Lecture Notes in Computer Science, pp 316–328

Beitzel SM, Jensen EC, Chowdhury A, Grossman D, Frieder O (2004) Hourly analysis of a very large topically categorized web query log. In: Proc. ACM SIGIR, pp 321–328

Callan J, Connell M, Du A (1999) Automatic discovery of language models for text databases. In: Proc. ACM SIGMOD, pp 479–490

Callan JP, Lu Z, Croft WB (1995) Searching distributed collections with inference networks. In: Proc. ACM SIGIR, pp 21–28

Craswell N, Bailey P, Hawking D (2000) Server selection on the world wide web. In: Proc. ACM International Conference on Digital Libraries, pp 37–46

D'Souza D, Thom JA, Zobel J (2004) Collection selection for managed distributed document databases. Information Processing and Management 40(3):527–546

French JC, Powell AL, Viles CL, Emmitt T, Prey KJ (1998) Evaluating database selection techniques: A testbed and experiment. In: Proc. ACM SIGIR, pp 121–129

French JC, Powell AL, Callan J, Viles CL, Emmitt T, Prey KJ, Mou Y (1999) Comparing the performance of database selection algorithms. In: Proc. ACM SIGIR, pp 238–245

Fuhr N (1999) A decision-theoretic approach to database selection in networked IR. ACM Trans Info Systems 17(3):229–249

Gravano L, García-Molina H (1995) Generalizing GlOSS to vector-space databases and broker hierarchies. In: Proc. VLDB, pp 78–89

Gravano L, García-Molina H, Tomasic A (1994) The effectiveness of GlOSS for the text database discovery problem. In: Proc. ACM SIGMOD, pp 126–137

Gravano L, García-Molina H, Tomasic A (1999) GlOSS: Text-source discovery over the internet. ACM Trans Database Systems 24(2):229–264

Harman DK (2005) The TREC test collections. In: Voorhees EM, Harman DK (eds) TREC: Experiment and Evaluation in Information Retrieval, MIT Press, pp 21–52

Hawking D, Thistlewaite P (1999) Methods for information server selection. ACM Trans Info Systems 17(1):40–76

Hawking D, Thomas P (2005) Server selection methods in hybrid portal search. In: Proc. ACM SIGIR, pp 75–82

Hawking D, Bailey P, Craswell N (2000) Efficient and flexible search using text and metadata. Tech. Rep. 2000/83, CSIRO Mathematical and Information Sciences

Jansen BJ, Spink A, Saracevic T (2000) Real life, real users, and real needs: A study and analysis of user queries on the web. Information Processing and Management 36(2):207–227

Larson RR (2003) Distributed IR for digital libraries. In: Proc. European Conf. on Research and Advanced Technology for Digital Libraries, Springer-Verlag, Lecture Notes in Computer Science, vol 2769

Liu KL, Santoso A, Yu C, Meng W, Zhang C (2001) Discovering the representative of a search engine. In: Proc. CIKM, pp 577–579, poster

Liu KL, Meng W, Qui J, Yu C, Raghavan V, Wu Z, Lu Y, He H, Zhao H (2007) AllInOneNews: Development and evaluation of a large-scale news metasearch engine. In: Proc. ACM SIGMOD, pp 1017–1028

Nottelmann H, Fuhr N (2003) Evaluating different methods of estimating retrieval quality for resource selection. In: Proc. ACM SIGIR, pp 290–297

Nottelmann H, Fuhr N (2004) Combining CORI and the decision-theoretic approach for advanced resource selection. In: Proc. ECIR

Powell AL, French JC (2003) Comparing the performance of collection selection algorithms. ACM Trans Info Systems 21(4):412–456

Powell AL, French JC, Callan J, Connell M, Viles CL (2000) The impact of database selection on distributed searching. In: Proc. ACM SIGIR, pp 232–239

Rasolofo Y, Abbaci F, Savoy J (2001) Approaches to collection selection and results merging for distributed information retrieval. In: Proc. CIKM

Robertson SE, Walker S, Jones S, Hancock-Beaulieu MM, Gatford M (1994) Okapi at TREC-3. In: Proc. TREC, pp 109–126

Shen Y, Lee DL (2002) A meta-search method reinforced by cluster descriptors. In: Proc. International Conference on Web Information Systems Engineering

Shokouhi M (2007) Central-rank-based collection selection in uncooperative distributed information retrieval. In: Proc. ECIR, pp 160–172

Shokouhi M, Zobel J, Scholer F, Tahaghoghi SMM (2006) Capturing collection size for distributed non-cooperative retrieval. In: Proc. ACM SIGIR, pp 316–323

Shokouhi M, Baillie M, Azzopardi L (2007) Updating collection representations for federated search. In: Proc. ACM SIGIR, pp 511–518

Si L, Callan J (2003a) The effect of database size distribution on resource selection algorithms. In: Proc. ACM SIGIR, pp 31–42

Si L, Callan J (2003b) Relevant document distribution estimation method for resource selection. In: Proc. ACM SIGIR, pp 298–305

Si L, Callan J (2003c) A semisupervised learning method to merge search engine results. ACM Trans Info Systems 21(4)

Si L, Callan J (2005) Modeling search engine effectiveness for federated search. In: Proc. ACM SIGIR, pp 83–90

Si L, Jin R, Callan J, Ogilvie P (2002) A language modeling framework for resource selection and results merging. In: Proc. CIKM, pp 391–397

Silverstein C, Henzinger M, Marais H, Moricz M (1999) Analysis of a very large web search engine query log. SIGIR Forum 33(1):6–12

Spink A, Ozmutlu S, Ozmutlu HC, Jansen BJ (2002) U.S. versus European web searching trends. SIGIR Forum 36(2):32–38

Thomas P (2008) Server characterisation and selection for personal metasearch. PhD thesis, Australian National University

Thomas P, Hawking D (2007) Evaluating sampling methods for uncooperative collections. In: Proc. ACM SIGIR, pp 503–510

Thomas P, Hawking D (2008) Experiences evaluating personal metasearch. In: Proc. IIiX Symposium on Information Interaction in Context, pp 136–138

Wu S, Crestani F (2002) Multi-objective resource selection in distributed information retrieval. In: Proc. International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Annecy

Wu Z, Meng W, Yu C, Li Z (2001) Towards a highly-scalable and effective metasearch engine. In: Proc. WWW, pp 386–395

Xu J, Croft WB (1999) Cluster-based language models for distributed retrieval. In: Proc. ACM SIGIR, pp 254–261

Yuwono B, Lee DL (1997) Server ranking for distributed text retrieval systems on the internet. In: Proc. 5th Int. Conf. on Database Systems for Advanced Applications, pp 41–49

Zhang Y, Moffat A (2006) Some observations on user search behaviour. In: Proc. Australasian Document Computing Symposium

Zobel J (1997) Collection selection via lexicon inspection. In: Proc. Australasian Document Computing Symposium