IR-181

The Effects of Query-Based Sampling on Automatic Database Selection Algorithms

Jamie Callan*
School of Computer Science
Carnegie Mellon Univ.
Pittsburgh, PA 15243
callan+@cs.cmu.edu

Allison L. Powell†, James C. French,
Dept of Computer Science
Univ. of Virginia
Charlottesville, VA
alp4g@cs.virginia.edu
french@cs.virginia.edu

Margaret Connell‡
Computer Science Dept
Univ. of Massachusetts
Amherst, MA 01003
connell@cs.umass.edu

Abstract

Database selection algorithms need to know the subject areas covered by each text database, but this metadata can be difficult to acquire in multi-party environments, such as the Internet, where each party has different interests and capabilities. Query-based sampling is a relatively new technique in which metadata is inferred by interacting with each text database and observing the outcomes. Query-based sampling is a solution to the problem of discovering the contents of each database in multi-party environments, but its generality and effectiveness had not been tested under a wide range of conditions.

This paper investigates the generality and effectiveness of query-based sampling with three well-known database selection algorithms (*gGlOSS*, *CORI*, *CVV*). Experimental results support the generality of query-based sampling as a solution for acquiring database descriptions in multi-party environments. The experiments also compare the effectiveness of the database selection algorithms under different conditions.

1 Introduction

When many text databases are available for search, as is common in environments with access to wide-area networks, the first information access problem is deciding which databases to search. When the number of databases is small, a person can be familiar with the general subject area covered by each database. However, when hundreds or thousands of searchable text databases are available, a person may need help deciding which databases to select. Automatic *database selection* algorithms assist with this choice by identifying the databases that best satisfy the information need, according to some metric [8, 4, 15, 14, 17, 10, 5, 16].

Database selection algorithms need to know what each database contains. This information is often derived from a *resource description* that lists the words that occur in the database and statistics based on their frequencies of occurrence. Three methods have been proposed for acquiring such metadata automatically: The STARTS protocol [7], lightweight probes [10], and query-based sampling [2]. STARTS is a *cooperative* protocol, in which all parties are trusted to exchange accurate metadata upon request. Lightweight probes is also a cooperative protocol, in which short probe queries are sent to each database, and each database is trusted to return accurate occurrence statistics for query terms. Query-based sampling is a technique in which resource descriptions[1] are inferred by interacting with each database and observing the outcomes.

Query-based sampling is a relatively new technique, hence little is known about its generality and behavior under a variety of conditions. Prior research demonstrated its effectiveness at learning accurate metadata

*This work supported in part by NSF grants IIS-9873009 and EIA-9983253.
†This work supported in part by DARPA contract N66001-97-C-8542 and NASA GSRP NGT5-50062.
‡This work supported in part by the NSF, Library of Congress and Dept. of Commerce under cooperative agreement EEC-9209623, by NSF grant EIA-9983215, and by the U.S. Patent and Trademark Office and Defense Advanced Research Projects Agency/ITO under ARPA order D468, issued by ESC/AXS contract F19628-95-C-0235.
[1] Also called *language models* in [2].

1. Select an initial query term.
2. Run a one-term query on the database.
3. Retrieve the top N documents returned by the database.
4. Extract words and frequencies from the top N documents.
5. Add the words and their frequencies to the learned resource description.
6. If a stopping criterion has not yet been reached,
   (a) Select a new query term.
   (b) Go to Step 2.

Figure 1: The query-based sampling algorithm.

for several research testbeds of varying size and heterogeneity [2]. Later research demonstrated that learned metadata resulted in relatively accurate database selection [1].

The early results were encouraging, but they studied query-based sampling under a set of relatively narrow conditions. Retrieval results were obtained with just one database selection algorithm (*CORI*), and the queries that were used were relatively long by some standards. The research presented in this paper investigates the generality of query-based sampling by examining its effectiveness with queries of two lengths, and with three well-known database selection algorithms.

The next section describes query-based sampling. Sections 3 and 4 state a hypothesis and describe the experimental methodology and data used to test it. Sections 5 and 6 describe the experimental results, and Section 7 concludes.

2 Query-Based Sampling

Research on automatic database selection began with *complete resource descriptions*, which are a type of metadata that lists every term occurring in the database, and the degree to which each term describes the contents. Algorithms differ in how they represent the descriptive power of a term, but common choices are based on the term's frequency within documents and/or the collection [8, 4, 17].

Complete resource descriptions are a convenient form of metadata for the researcher, but they present problems in some environments. A complete resource description can only be obtained with the cooperation of the service provider, which must either provide the resource description, or allow a third party to traverse every element of the database. However, there are many reasons why a resource provider might be unable or unwilling to provide desired resource descriptions, and why it might not allow third parties to traverse its entire database. If it did choose to provide resource descriptions upon request, as it could do using the STARTS protocol [7] or XML, the recipient could not verify that the resource descriptions represent the database contents accurately. Database contents might be misrepresented, either deliberately or accidentally.

A *partial* resource description lists only some of the terms occurring in the database and only estimates the degree to which each term describes database contents. Partial resource descriptions can be obtained for text databases relatively easily. Heap's law indicates that the size of a corpus vocabulary V is related to corpus size n by $V = Kn^{\beta}$, where n is the number of word occurrences, $K \approx 20$, and $\beta \approx 0.6$ [11]. As the corpus is examined, the rate of vocabulary growth declines, quickly reaching the point where new terms are introduced rarely. If it is possible to examine *some* of the documents of a corpus, Heap's law suggests that accurate resource descriptions can be constructed relatively quickly.

Query-based sampling is a technique for learning partial resource descriptions by running single-term queries on a database and examining a small number of the documents that are returned in response (Figure 1). Prior research showed that query-based sampling was robust with respect to how query terms were chosen, and how many documents were examined per query [2, 1]. Variations in parameter settings caused the speed of resource description convergence to vary, smoothly and relatively predictably. Simple combinations of parameter settings, for example, choosing query terms randomly from the resource description being learned, and examining 1-4 documents per query, were consistently effective. In tests with databases of different size and heterogeneity, accurate resource descriptions were learned after examining only 250-300 documents [2, 1].

3 Research Questions

Prior research showed that query-based sampling learned partial resource descriptions that correlated highly with the complete resource descriptions for those databases [2]. However, it was not known what degree of correlation was required in order to produce accurate database ranking. Preliminary experiments suggested that the correlation was sufficient for at least one ranking algorithm [1], but the queries used in that study were considerably longer and more structured than queries found in environments such as Web search.

The study presented here was intended to address questions about the generality of query-based sampling under a variety of conditions. Specifically, we wanted to study the effectiveness of several different database ranking algorithms with learned and complete resource descriptions, to determine whether database ranking algorithms are in general relatively insensitive to the minor inaccuracies in learned metadata. We also wanted to investigate whether partial resource descriptions made database ranking algorithms more sensitive to query length. The research questions and issues of interest were expressed by the following hypothesis.

Hypothesis: The *gGlOSS*, *CORI*, and *CVV* algorithms are all equally *unaffected* by the differences between complete and learned resource descriptions. For example, *gGlOSS* is expected to produce rankings of similar accuracy when using complete and learned resource descriptions.[2]

Our research interests are expressed as a hypothesis in order to make them precise. However, we would not be disappointed if it were contradicted by the experimental evidence. The hypothesis and the experiments that test it were intended to provide information about the behavior of query-based sampling and three well-known ranking algorithms under a range of realistic conditions. Whatever the outcome, the results would provide information to guide future research.

3.1 Database Ranking Algorithms

Three database ranking algorithms were studied: *gGlOSS* [8, 6], *CORI* [4, 6, 16], and *CVV* [17]. These algorithms were chosen because they are relatively well-known in database and/or information retrieval research communities. All three algorithms are easy to implement. None of them require training data. The three algorithms, and two additional baseline algorithms, are described briefly below.

3.1.1 *gGlOSS*

gGlOSS [8] is a database ranking algorithm based on the vector-space model of information retrieval. *gGlOSS* estimates the *goodness* of a database for a query as a function of the estimated similarities of its documents to the query. *gGlOSS* provides several different methods of ranking databases (i.e., combining document similarities), for example *Ideal(l)*, *Sum(l)*, and *Max(l)*.

We chose the *Ideal(0)* ranking method, which is consistent with prior research. In the simple case relevant to this research, the *Ideal(0)* ranking is equivalent to the *Sum(0)* ranking [6], which is calculated as [8]:

$$Score(Q, db_i) = \sum_{t_j \in Q} qtf_j \times w_{ij}$$

where qtf_j is the frequency of term t_j in query Q, and w_{ij} is the sum of the SMART *ntc* weights of t_j over all documents in db_i.

3.1.2 *CORI*

The *CORI* algorithm is based on a Bayesian Inference Network model of information retrieval [4]. Resource descriptions consist of the terms that occur in a database, information about how many documents contain each term, and information about the size of the database. The database ranking algorithm also monitors the number of databases that contain each term, so that an *inverse collection frequency* (*icf*) can be computed.

[2] There is no assumption that two different algorithms, for example, *gGlOSS* and *CORI*, will produce equally accurate rankings under a given set of conditions.

In the simple case relevant to this research, *CORI* is summarized by the following equation [4, 1]:

$$p(Q|db_i) = \frac{1}{|Q|} \cdot \sum_{t_j \epsilon Q} \left(0.4 + 0.6 \cdot \frac{df_{ij}}{df_{ij} + 50 + 150 \cdot cw_i / avg\_cw} \cdot \frac{\log\left(\frac{N+0.5}{cf_j}\right)}{\log(N+1.0)} \right)$$

where df_{ij} is the number of documents in db_i containing t_j, cw_i is the number of word occurrences in db_i, avg_cw is the average of the cw_i values from all databases, N is the number of databases, and cf_j is the number of databases containing term t_j.

3.1.3 *CVV*

The *CVV* database ranking algorithm uses a combination of document frequency and cue validity variance information [17]. Cue validity variance characterizes the distribution of the density of *df* values, i.e., the variability of the fraction of documents in a database that contain a given term. Document frequency information is used to estimate the importance of a term within a database; the *CVV* component estimates whether a term is useful for differentiating one database from another. Database scores are computed as:

$$\begin{aligned}
Score(db_i, Q) &= \sum_{\{t_j \epsilon Q\}} df_{ij} \cdot CVV_j \\
CVV_j &= \frac{1}{N} \cdot \sum_{i=1}^{N} (CV_{ij} - \overline{CV_j})^2 \\
CV_{ij} &= \frac{\frac{df_{ij}}{|db_i|}}{\frac{df_{ij}}{|db_i|} + \frac{\sum_{k \neq i}^{N} df_{kj}}{\sum_{k \neq i}^{N} |db_k|}} \\
\overline{CV_j} &= \frac{1}{N} \cdot \sum_{i=1}^{N} CV_{ij}
\end{aligned}$$

where $|db_i|$ is the number of documents in database db_i.

The goal of the *CVV* ranking algorithm is to identify databases with a high concentration of query terms.

3.1.4 RBR and SBR

Two other database rankings were included as baselines for comparison: RBR and SBR.

RBR (Relevance Based Ranking) is an omniscient algorithm that ranks databases by the number of relevant documents they contain for a query [5]. The RBR baseline is based on the assumption that databases containing many relevant documents should be ranked ahead of databases containing few or no relevant documents. RBR is included as an upper bound on the accuracy of database ranking algorithms, because it shows what can be achieved with complete knowledge.

SBR (Size Based Ranking) ranks databases by the number of documents they contain [5]. SBR is based on the assumption that large databases should be ranked ahead of small databases. Note that SBR ranks databases in the same order for all queries. We include SBR in our experiments as a lower bound because it shows what can be achieved without detailed knowledge of the contents of a database.

4 Experimental Methodology

A research goal was to compare the three automatic database selection algorithms on a reasonably large testbed under a variety of conditions. We were specifically interested in how the algorithms were affected by two types of resource descriptions (complete and sampled) and two types of queries (short, long). Prior work evaluated the accuracy of database selection algorithms using two complementary approaches: i) measure how well the algorithm's *database* rankings match desired rankings [6, 5], and ii) measure the precision and/or recall of the final *document* rankings [4, 16]. Each approach provides a different perspective on database selection accuracy, so both were used in the work reported here.

	Minimum	Average	Maximum
Documents Per DB	752	10,782	39,723
Megabytes Per DB	28.0	33.4	41.8

Table 1: Summary statistics for the 100 databases in the testbed.

Query Set Name	TREC Topics	Topic Field	Avg. Length (Words)
Short	51-150	Title	3.4
Long	51-150	Concepts	24.1

Table 2: Summary statistics for the query sets used with the testbed.

4.1 Data

Research was conducted on a testbed of 100 databases (Table 1). The testbed was created from 3 gigabytes of data provided by the U.S. National Institute for Standards and Technology (NIST) to participants in its Text REtrieval Conferences (TREC) [9]. Documents on TREC CDs 1, 2, and 3 were organized into document collections of about 30 megabytes each, ordered as they appeared on the TREC CDs, and with the additional restriction that the documents in a collection were from the same source (e.g., Wall Street Journal). This approach to constructing multi-database testbeds is consistent with past research practice (e.g., [4, 15, 14, 9, 6, 10, 5, 16], and this testbed was used in prior research [5].

Two sets of TREC topics, 51-100 and 101-150, were used in the research. These topics were chosen because they have relevance judgements for all parts of TREC CDs 1, 2, and 3. Sets of short and long queries were created by extracting text from different fields of the TREC topics, as summarized in Table 2. Query operators, for example, Proximity, Phrase, or differential weighting, are often used to create queries from TREC topics, but none were used in this research, because the three algorithms vary in their ability to cope with query structure. The queries were all unstructured, "bag of words" queries.

The experimental results obtained with the two topic sets were relatively consistent throughout all of our experiments, so in this paper we present only results for the combined set of 100 topics, to reduce space and improve clarity. This decision is consistent with recent papers in this research area [12].

4.2 Resource Descriptions (Metadata)

Two types of resource descriptions were used in the experiments. The *complete* resource descriptions (abbreviated to *complete description* in this paper) listed each database term and associated statistics computed from basic term frequency information. Each algorithm formats its resource descriptions differently, so they did not all use the same resource description, although it is in principle possible to do so. Instead, three complete resource descriptions (one per algorithm) were created for each of the 100 databases, with each algorithm having access to precisely the same information about the contents of each database.

Each *learned* resource description was created from a sample of 300 database documents obtained by query-based sampling. Samples were obtained following the methodology described in [2, 1]. An initial query term was selected randomly from a large corpus of documents, and submitted to a searchable version of the database (searched by Inquery [3]). The texts of the top 4 documents were used to update the resource description. Subsequent query terms were selected randomly from the resource description being learned. This process continued until 300 unique documents were obtained (about 75 queries). The stopping criterion was based on prior work suggesting that resource descriptions tended to be stable and accurate after 300 documents [2, 1].

Three learned resource descriptions (one per algorithm) were created for each of the 100 databases. Although different algorithms can not share a single resource description for a given database, due to their differing requirements, all of the learned resource descriptions for a given database were based on the same set of 300 sampled documents. Each algorithm had the same information about the contents of each database.

4.3 Measuring the Accuracy of Database Rankings

There is considerable variation in how the accuracy of database selection algorithms is measured. We believe that it is important to measure which of two database ranking algorithms is more accurate without making assumptions about how, or how well, other components of a multi-database system function [6]. That is, the accuracy of the database ranking algorithm should be measured independently of how well document rankings from different databases are merged, or how well the effects of differing corpus statistics are handled.

Gravano, *et al.*, [8] define database ranking metrics P_n and R_n that are analogous to the precision and recall metrics for document ranking. P_n is the percentage of databases ranked [1..n] that have some merit for the query. R_n compares the amount of merit in databases ranked [1..n] with the amount of merit in the top n databases of a desired database ranking [8, 6].

We chose the merit function to be the number of relevant documents a database contained for a query, and the desired database ranking to be a relevance based ranking (RBR, Section 3.1). These choices are consistent with recent research [6, 5]. Given these choices, for a database ranking e ('estimated') and its corresponding relevance based database ranking b ('baseline'), P_n and R_n were defined at rank n as:

$$P_n = \frac{\sum_{i=1}^{n} \begin{cases} 1 & \text{if NumRel } (db_{e_i}) > 0 \\ 0 & \text{otherwise} \end{cases}}{n} \quad (1)$$

$$R_n = \frac{\sum_{i=1}^{n} \text{NumRel } (db_{e_i})}{\sum_{i=1}^{n} \text{NumRel } (db_{b_i})} \quad (2)$$

where NumRel (db_{j_i}) is the number of relevant documents in the i'th database of database ranking j (baseline b, or estimated e).

4.4 Measuring the Accuracy of Document Rankings

Although it is good experimental methodology to evaluate the components of an IR system independently, people using the system care more about the accuracy of the final document rankings than about the accuracy of any single system component. Metrics such as P_n and R_n provide no information about how a difference in database ranking accuracy affects the final document rankings. A common choice, which we adopted, is to search the highest ranked databases, merge the results returned, and then measure the quality of the final, merged list of documents [4, 15, 16, 10].

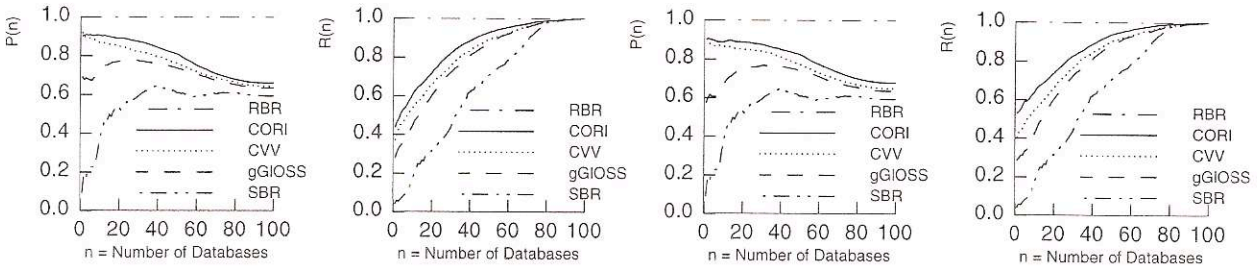
After a database selection algorithm ranked the 100 databases, the 10 most highly ranked databases were considered "selected for search" by that algorithm. The query used to rank databases was broadcast to Inquery IR systems serving each selected database [3, 4], which each returned ranked lists of 100 documents. Each database was independent of the other databases; there was no exchange among databases of corpus statistics or other information, and no global corpus statistics were computed or maintained.

The merging of document rankings produced from different databases is a well-known difficult IR problem. Differences in corpus statistics (particularly inverse document frequency, or *idf*) make document scores from different databases incomparable [14]. Common solutions are to maintain global corpus information, which is not always practical, or to recompute document scores at the search client, which is undesirable excess computation (although not impractical). A third choice is to estimate normalized document scores heuristically, which has been effective in spite of its lack of theoretical support. We selected the third choice, in part because it is the default behavior of the Inquery IR system [4].

The default Inquery multi-database merging algorithm uses a combination of the score for the database and the score for the document to estimate a normalized score. In order to avoid a bias towards the *CORI* algorithm, the database scores assigned by the three database ranking algorithms were discarded.New, algorithm-independent database scores were calculated, based on database ranks, according to the function:

$$C' = (101 - R)/100 \quad (3)$$

where R is the database rank. The fifth-ranked database ($R = 5$) received a score of (101 - 5) / 100 = 0.96.



Short queries, P_n (a) Short queries, R_n (b) Long queries, P_n (c) Long queries, R_n (d)

Figure 2: Measures of how well three database ranking algorithms match a relevance based ranking for long and short query sets constructed from TREC topics 51-150. Complete resource descriptions.

Precision	Short Queries (51-150)			Long Queries (51-150)		
at Rank	*gGlOSS*	*CORI*	*CVV*	*gGlOSS*	*CORI*	*CVV*
5 docs	0.4060	0.4640 (+14.3%)	0.4420 (+8.9%)	0.5940	0.5700 (−4.0%)	0.6220 (+4.7%)
10 docs	0.3850	0.4290 (+11.4%)	0.3980 (+3.4%)	0.5590	0.5610 (+3.6%)	0.5750 (+2.9%)
15 docs	0.3653	0.4127 (+13.0%)	0.3913 (+7.1%)	0.5273	0.5500 (+4.3%)	0.5487 (+4.1%)
20 docs	0.3460	0.3935 (+13.7%)	0.3770 (+9.0%)	0.4995	0.5405 (+8.2%)	0.5300 (+6.1%)
30 docs	0.3220	0.3737 (+16.1%)	0.3607 (+12.0%)	0.4600	0.5170 (+12.4%)	0.4927 (+7.1%)
100 docs	0.2404	0.2907 (+20.9%)	0.2759 (+14.8%)	0.3256	0.4089 (+25.6%)	0.3725 (+14.4%)

Table 3: A comparison of how well 3 algorithms rank databases, as measured by the precision of the document rankings that are produced. 10 (out of 100) databases searched. Complete resource descriptions.

The normalized document score D'' for a document with an initial score D was computed as:

$$D' = (D - D_{\min})/(D_{\max} - D_{\min}) \quad (4)$$

$$D'' = (D' + 0.4 \cdot C' \cdot D')/1.4 \quad (5)$$

where $D_{\max}$ and $D_{\min}$ were the highest and lowest scores that the document ranking algorithm could produce for that query in that database. D'' was normalized by 1.4 to keep document scores in the range [0..1]. These merging functions are the defaults for the multi-database version of Inquery [1].

Normalized scores were computed for each of the 100 documents returned by each of the 10 databases (1,000 documents). Documents were ranked by their normalized scores. All but the top 100 were discarded.

The quality of document rankings is often measured by precision and recall. Recent trends towards interactive search of very large databases have increased the importance of precision and reduced the importance of recall in many environments. If a person will only examine 20 or 30 documents, it does not matter whether the system retrieved 25% or 75% of the 300 relevant documents that were available in the database(s). We measured precision at document ranks 5, 10, 15, 20, 30, and 100, as is done for the TREC conferences [9].

5 Experimental Results: Comparing DB Selection Algorithms

Our first set of experiments compared the accuracy of *gGlOSS*, *CORI*, and *CVV* using *complete* resource descriptions. These experiments provided baseline measurements for the experiments with *partial* resource descriptions (Section 6). They are also one of the first direct comparisons of these three algorithms.

A set of 100 *complete* resource descriptions was created, one for each database. The experiments followed the experimental methodology described in Section 4, in which an algorithm selected 10 (out of 100) databases, each of the 10 was searched independently by Inquery, and the document rankings returned by each database were merged to produce a final ranked list of 100 documents.

The effectiveness of the three algorithms at ranking databases is summarized in Figure 2. *CORI* was the best at ranking databases containing many relevant documents ahead of databases containing few relevant

Precision at Rank	Long Queries (51-150), 10 databases searched			Long Queries (51-150), 5 databases searched		
	gGlOSS	*CORI*	*CVV*	*gGlOSS*	*CORI*	*CVV*
5 docs	0.5940	0.5700 (−4.0%)	0.6220 (+4.7%)	0.3800	0.5560 (+46.3%)	0.5340 (+40.5%)
10 docs	0.5590	0.5610 (+3.6%)	0.5750 (+2.9%)	0.3550	0.5450 (+53.5%)	0.5160 (+45.4%)
15 docs	0.5273	0.5500 (+4.3%)	0.5487 (+4.1%)	0.3393	0.5193 (+53.1%)	0.4873 (+43.6%)
20 docs	0.4995	0.5405 (+8.2%)	0.5300 (+6.1%)	0.3250	0.4945 (+52.2%)	0.4620 (+42.2%)
30 docs	0.4600	0.5170 (+12.4%)	0.4927 (+7.1%)	0.2920	0.4607 (+57.8%)	0.4183 (+43.3%)
100 docs	0.3256	0.4089 (+25.6%)	0.3725 (+14.4%)	0.2067	0.3478 (+68.3%)	0.2986 (+44.5%)

Table 4: A comparison of how changing the number of databases searched affects 3 database selection algorithms, as measured by the precision of the document rankings that are produced. 100 database testbed. Complete resource descriptions.

documents (Figures 2b and 2d) and was least likely to be distracted by databases containing no relevant documents (Figures 2a and 2c). *CVV* was also good at ignoring databases containing no relevant documents (Figures 2a and 2c), but was slightly less effective at ranking databases containing many relevant documents ahead of databases containing few relevant documents (Figures 2b and 2d). *gGlOSS* was most likely to rank highly databases containing no relevant documents (Figures 2a and 2c).

Figures 2a and 2c confirm prior research reporting that *gGlOSS* has a bias towards databases containing many documents [5], illustrated here by the tendency to follow the size-based ranking (SBR) curve. *gGlOSS* ranked DOE databases highly for 22% of the queries, but these databases, which contain about three times as many documents as other databases, usually contained no relevant documents.

Table 3 summarizes how the different database ranking algorithms affected the final document rankings. *gGlOSS* is used as a baseline because it is the most well-known of the three algorithms [8, 6]. The document rankings produced with *CORI* were usually the most accurate. Document rankings produced with *CVV* were the second most accurate, and document rankings produced with *gGlOSS* were consistently the least accurate. Conventional wisdom in IR research is that people don't usually notice relative differences in precision of less than 10%. The difference between *gGlOSS* and *CORI* might be noticeable when short queries are used. Other differences among the three algorithms would be less noticeable.

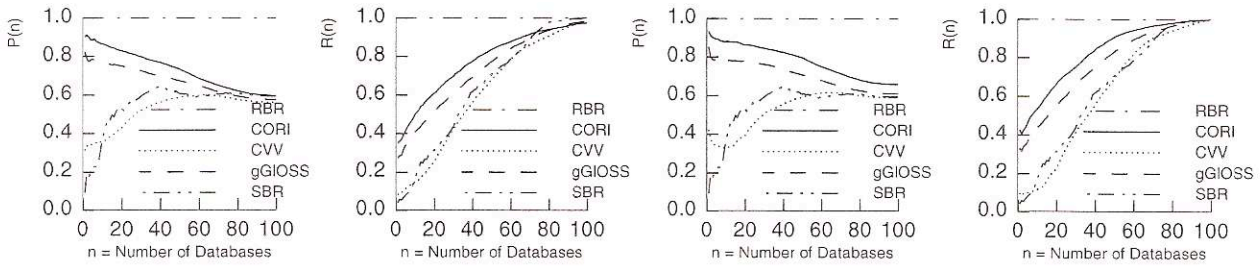
One might have expected a larger difference between document rankings produced with *gGlOSS* and the other algorithms, given its less accurate database rankings. *gGlOSS* was helped by an experimental methodology in which 10% of the databases were searched. As long as *gGlOSS* got 3-4 good databases in the top 10, the Inquery search and result-merging algorithms would find and move the relevant documents to the top of the rankings. When half as many databases were searched, *gGlOSS* database rankings produced significantly less accurate document rankings, whereas the more accurate *CORI* algorithm was affected only slightly (Table 4). This result suggests that *gGlOSS* is very sensitive to the operational environment.

6 Experimental Results: The Effects of Query-Based Sampling

The second set of experiments we present investigated the hypothesis that all three database selection algorithms are equally *unaffected* by the differences between complete and learned resource descriptions. This hypothesis can be viewed as an extension of work reported in [2] which showed that the *CORI* database ranking algorithm worked equally well with both complete and sampled resource descriptions.

A set of 100 *complete* resource descriptions was created, one for each database. Another set of 100 resource descriptions was created by sampling each database with enough queries to obtain 300 documents per database, as described in Section 4.2. The experiments followed the experimental methodology described in Section 4, in which an algorithm selected 10 (out of 100) databases, each of the 10 was searched independently by Inquery, and the document rankings returned by each database were merged to produce a final ranked list of 100 documents.

The *CORI* database rankings were the most accurate. *CORI* was the best at ranking databases containing many relevant documents ahead of databases containing few relevant documents (Figures 3b and 3d), and it was the least likely to be distracted by databases containing no relevant documents (Figures 3a and 3c).



Short queries, P_n (a) Short queries, R_n (b) Long queries, P_n (c) Long queries, R_n (d)

Figure 3: Measures of the effects of query-based sampling on how well three database ranking algorithms match a relevance based ranking for long and short query sets constructed from TREC topics 51-150. Partial resource descriptions.

Short Queries (51-150)						
Precision	*gGlOSS*		*CORI*		*CVV*	
at Rank	Complete	Sampled	Complete	Sampled	Complete	Sampled
5 docs	0.4060	0.3660 (−9.9%)	0.4640	0.4400 (−5.2%)	0.4420	0.2320 (−47.5%)
10 docs	0.3850	0.3500 (−9.1%)	0.4290	0.4090 (−4.7%)	0.3980	0.2150 (−46.0%)
15 docs	0.3653	0.3347 (−8.4%)	0.4127	0.3840 (−7.0%)	0.3913	0.1987 (−49.2%)
20 docs	0.3460	0.3235 (−6.5%)	0.3935	0.3690 (−6.2%)	0.3770	0.1860 (−50.7%)
30 docs	0.3220	0.3007 (−6.6%)	0.3737	0.3417 (−8.6%)	0.3607	0.1633 (−54.7%)
100 docs	0.2404	0.2240 (−6.8%)	0.2907	0.2642 (−9.1%)	0.2759	0.1015 (−63.2%)
Long Queries (51-150)						
Precision	*gGlOSS*		*CORI*		*CVV*	
at Rank	Complete	Sampled	Complete	Sampled	Complete	Sampled
5 docs	0.5940	0.4860 (−18.2%)	0.5700	0.5580 (−2.1%)	0.6220	0.2480 (−60.1%)
10 docs	0.5590	0.4740 (−15.2%)	0.5610	0.5720 (+2.0%)	0.5750	0.2220 (−61.4%)
15 docs	0.5273	0.4593 (−12.9%)	0.5500	0.5480 (−0.4%)	0.5487	0.2053 (−62.6%)
20 docs	0.4995	0.4335 (−13.2%)	0.5405	0.5270 (−2.5%)	0.5300	0.1920 (−63.8%)
30 docs	0.4600	0.4153 (−9.7%)	0.5170	0.5030 (−2.7%)	0.4927	0.1680 (−65.9%)
100 docs	0.3256	0.3160 (−3.0%)	0.4089	0.3808 (−6.9%)	0.3725	0.1113 (−70.1%)

Table 5: The effects of query-based sampling on database ranking algorithms, as measured by the precision of the document rankings that result. 10 databases searched in a 100 database testbed. TREC topics 51-150. Partial resource descriptions.

The *gGlOSS* rankings were nearly as accurate (Figure 3). This result might be viewed as surprising, because the *gGlOSS* rankings with complete resource descriptions were not particularly accurate (Figure 2). However, all of the resource descriptions in this experiment were created from samples of equal size, thus neutralizing the *gGlOSS* bias towards large databases.

The *CVV* database rankings were extremely poor (Figure 3). Closer inspection reveals that *CVV* regularly ranked the FR (U.S. Federal Register) and PATN (U.S. patents) databases highly, which was rarely a good choice. The bias towards FR and PATN databases was a consequence of the long documents they contain. Long documents have more unique terms, and more term occurrences, than short documents, so the FR and PATN resource descriptions had more terms and higher term frequencies than resource descriptions for other databases. For example, a resource description created from 300 1989 Wall Street Journal documents contained 18,671 terms (75,487 word occurrences), but a comparable resource description for 1988 Federal Register data contained 28,367 terms (387,846 word occurrences). *CVV* has performed well in experiments with other multi-database testbeds [13], so we hypothesize that the cause of its poor performance in these experiments is incompatibility with partial resource descriptions. One might conclude that *CVV* could be improved by scaling frequencies by a metric related to document length.

Table 5 summarizes the impact of these database rankings on the precision of the final document rankings.

The *gGlOSS* algorithm was affected moderately by the learned resource descriptions. The drop in precision was moderate ($\sim$ 8%) with short queries, and noticeable ($\sim$ 12%) with long queries. Although these results confirm that *gGlOSS* can be used with query-based sampling, the performance using *gGlOSS* with sampled descriptions was weaker than expected. The *gGlOSS* bias towards large databases depressed the baseline precision obtained with complete descriptions, but had no effect on the precision obtained with sampled descriptions (because they each contained the same number of documents). Thus, we expected the results with sampled descriptions and complete descriptions to be more similar than they were in this experiment.

The *CORI* algorithm was the only algorithm to demonstrate consistent behavior in these tests. The drop in precision was about 7% with short queries and about 2% with long queries. These results are consistent with previously published results [2, 1], and support the view that the *CORI* algorithm is relatively unaffected by the use of learned resource descriptions.

Learned resource descriptions caused a dramatic loss of precision with the *CVV* algorithm. Precision losses ranged from 50−60%, making learned resource descriptions an unacceptable choice with this algorithm.

Differences in the effectiveness of short and long queries with learned resource descriptions (Table 5) were consistent with the differences obtained with complete resource descriptions (Table 3). Partial resource descriptions learned by query-based sampling did not introduce additional sensitivity to query length.

These tests do not support the hypothesis that all three database ranking algorithms are equally unaffected by learned resource descriptions. Indeed, the differences are dramatic, providing strong evidence that some of the algorithms are not as robust as thought previously, and revealing algorithm characteristics that should be addressed by future research.

7 Conclusions

Our primary research goal was to investigate the generality of acquiring resource descriptions by query-based sampling. Prior research showed that query-based sampling produced resource descriptions that were relatively accurate, and that were compatible with one database selection algorithm when used with long, highly structured queries. The research described in this paper extends prior research on query-based sampling along two different dimensions, which we summarize briefly.

Resource descriptions acquired by query-based sampling were compatible with two of the three database selection algorithms tested (*CORI*, *gGlOSS*). Four of six tests showed only small or moderate differences in the results obtained with complete and partial resource descriptions. However, the complete incompatibility with *CVV* suggests that query-based sampling works best with algorithms that have good normalization for corpus statistics.

Query-based sampling also produced resource descriptions that were robust with respect to query length. This result is important because prior research was conducted with only long queries, which might be less sensitive than short queries to 'gaps' in a resource description. However, query length caused no additional sensitivity to partial resource descriptions in these experiments.

Our research also extends prior research on database selection algorithms, by testing three well-known algorithms under a variety of new conditions.

Weaknesses were found in the size normalization components of the *gGlOSS* and *CVV* algorithms. *gGlOSS* has a bias towards databases with many documents, and *CVV* has a bias towards databases with long documents. The *gGlOSS* bias was identified in earlier research, but its effect on retrieval results was unknown previously. The *CVV* bias was unknown previously, and its triggering conditions remain somewhat unclear. When triggered, effectiveness drops dramatically.

CORI was the most accurate and stable of the three algorithms in these tests. *CVV* was also very accurate when used with complete resource descriptions, but it failed dramatically when used with partial resource descriptions. *gGlOSS* was the least accurate when used with complete resource descriptions, and the differences would be noticeable to people using short queries.

When the percentage of databases that is searched is small, as would be the case in large scale environments, the differences among the algorithms are accentuated. Our results indicate that *CORI* is the only algorithm accurate enough for use in such environments.

The research reported here suggests several questions for future research, questions that might not arise in a study of a single algorithm or with a less extensive set of metrics. We hope to see more distributed IR research of this type in the future.

Acknowledgements

Any opinions, findings, conclusions, or recommendations expressed in this paper are the authors', and do not necessarily reflect those of the sponsors.

References

[1] J. Callan. Distributed information retrieval. In W.B. Croft, editor, *Advances in information retrieval*, chapter 5, pages 127–150. Kluwer Academic Publishers, 2000.

[2] J. Callan, M. Connell, and A. Du. Automatic discovery of language models for text databases. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, pages 479–490. ACM, 1999.

[3] J. P. Callan, W. B. Croft, and J. Broglio. TREC and TIPSTER experiments with INQUERY. *Information Processing and Management*, 31(3):327–343, 1995.

[4] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle, 1995. ACM.

[5] J. French, A. Powell, J. Callan, C. Viles, T. Emmitt, K. Prey, and Y. Mou. Comparing the performance of database selection algorithms. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 238–245. ACM, 1999.

[6] J.C. French, A.L. Powell, C.L. Viles, T. Emmitt, and K.J. Prey. Evaluating database selection techniques: A testbed and experiment. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1998.

[7] L. Gravano, K. Chang, H. García-Molina, and A. Paepcke. STARTS Stanford proposal for Internet meta-searching. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, 1997.

[8] L. Gravano and H. García-Molina. Generalizing GLOSS to vector-space databases and broker hierarchies. In *Proceedings of the 21st International Conference on Very Large Databases (VLDB)*, pages 78–89, 1995.

[9] D. Harman, editor. *Proceedings of the Third Text REtrieval Conference (TREC-3)*. National Institute of Standards and Technology Special Publication 500-225, Gaithersburg, MD, 1995.

[10] D. Hawking and P. Thistlewaite. Methods for information server selection. *ACM Transactions on Information Systems*, 17(1):40–76, 1999.

[11] H. S. Heaps. *Information Retrieval: Computational and Theoretical Aspects*. Academic Press, New York, 1978.

[12] A. Powell, J. French, J. Callan, M. Connell, and C. Viles. The impact of database selection on distributed searching. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–239. ACM, 2000.

[13] Allison L. Powell. *Database Selection in Distributed Information Retrieval: A Study of Multi-Collection Information Retrieval*. PhD thesis, Department of Computer Science, University of Virginia, January 2001.

[14] C. L. Viles and J. C. French. Dissemination of collection wide information in a distributed Information Retrieval system. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 12–20, Seattle, 1995. ACM.

[15] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. Learning collection fusion strategies. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–179, Seattle, 1995. ACM.

[16] J. Xu and W.B. Croft. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 254–261, Berkeley, 1999. ACM.

[17] B. Yuwono and D. L. Lee. Server ranking for distributed text retrieval systems on the Internet. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 41–49, Melbourne, 1997.