Comparing the Performance of Database Selection Algorithms

James C. French[1], Allison L. Powell[1], Jamie Callan[2], Charles L. Viles[3]
Travis Emmitt[1], Kevin J. Prey[1], Yun Mou[2]

[1]Dept. of Computer Science*
University of Virginia
Charlottesville, VA

[2]Computer Science Dept.†
Univ. of Massachusetts
Amherst, MA

[3]School of Info. and Library Science‡
Univ. of North Carolina–Chapel Hill
Chapel Hill, NC

Abstract We compare the performance of two database selection algorithms reported in the literature. Their performance is compared using a common testbed designed specifically for database selection techniques. The testbed is a decomposition of the TREC/TIPSTER data into 236 subcollections. The databases from our testbed were ranked using both the *gGlOSS* and *CORI* techniques and compared to a baseline derived from TREC relevance judgements. We examined the degree to which *CORI* and *gGlOSS* approximate this baseline. Our results confirm our earlier observation that the *gGlOSS* $Ideal(l)$ ranks do not estimate relevance-based ranks well. We also find that *CORI* is a uniformly better estimator of relevance-based ranks than *gGlOSS* for the test environment used in this study. Part of the advantage of the *CORI* algorithm can be explained by a strong correlation between *gGlOSS* and a size-based baseline (SBR). We also find that *CORI* produces consistently accurate rankings on testbeds ranging from 100–921 sites. However for a given level of recall, search effort appears to scale linearly with the number of databases.

1 Introduction

The proliferation of online resources, the growing need to conduct searches across many of these resources, and the de-facto requirement of pruning the resource set of interest to manageable size has increased attention on retrieval in the distributed environment. Distributed information retrieval encompasses many important problems, including database or collection selection [7, 11, 15, 13, 18, 24, 25, 14], collection fusion or results merging [2, 7, 8, 23, 22], and dissemination of collection information to increase retrieval effectiveness [10, 21, 20].

In this paper we focus on *database selection* as a fundamental problem in the distributed environment, providing the first direct comparison of two database selection techniques described in the literature. The problem can be stated intuitively as the process of selecting a (hopefully) small set of databases to which to send a query. Database selection is the first step in a process that continues with search at the distributed sites and fusing or merging of result lists from the sites. The primary goal in this database selection step is to select as small a set of collections as possible to send a query to without sacrificing retrieval effectiveness.

Several techniques for database selection have been proposed and independently evaluated; however the test environments have varied in both the underlying data and the evaluation methods. There is very little literature on the direct comparison of competing techniques. This study directly compares two techniques, *gGlOSS*[13, 14] and *CORI*[7], in a common test environment using the same data and evaluation techniques. We had two goals for this study. The first goal was to demonstrate a sound methodology for the systematic study of these algorithms and their relative performance. The second, and more important, goal was to gain insight into both the collective and individual behavior of these algorithms.

2 Database Selection Experiment

Distributed searching is composed of three fundamental activities: (1) choosing the specific databases to search, (2) searching the chosen databases, and (3) merging the results into a cohesive response. In these experiments we focus specifically on the first activity. Callan *et al.*[7] call this the *collection selection* problem while Gravano *et al.*[15] refer to it as the *text database resource discovery* problem. In French *et al.*[9] we refer to this as *database selection* and will retain that terminology here for consistency.

In earlier work[9] we described a testbed and methodology for examining the behavior of algorithms such as these. We also proposed metrics and analyses that were designed to illuminate the behavior of the algorithm under study. In that work[9], we determined that *gGlOSS*[13] does not estimate relevance-based rankings very well. In the present paper we are concerned with determining how *gGlOSS* com-

*This work supported in part by DARPA contract N66001-97-C-8542 and NASA GSRP NGT5-50062.

†This work supported in part by NSF, the Library of Congress, and the Department of Commerce under agreement EEC-9209623, and by the U.S. Patent and Trademark Office and DARPA/ITO under contract F19628-95-C-0235.

‡This work supported in part by DARPA contract N66001-97-C-8542.

pares with another database selection algorithm, *CORI*[7]. In this section, we describe the *gGlOSS* and *CORI* algorithms and review the testbed and metrics for comparison.

2.1 The Testbed

A number of researchers have been working on issues in distributed information retrieval systems but the test environments are idiosyncratic in both data and evaluation measures, making it impossible to compare results. In French *et al.*[9] we proposed a test environment for the systematic study of distributed information retrieval algorithms. Our testbed is based on the TIPSTER data used in the TREC[16] conferences. We decompose the large collections into smaller subcollections that serve as hypothetical "sites" in our distributed information retrieval test environment. The data is decomposed by source, year, and month resulting in 236 sites. We used TREC topics 51-150 as the test queries in our earlier study[9]. The characteristics of this testbed, the queries used, and other details can be found in French *et al.*[9].

2.2 The Algorithms Considered

2.2.1 gGlOSS

Gravano *et al.*[15] proposed *GlOSS*, the *Glossary-of-Servers Server*, as an approach to the database selection problem for the Boolean IR model. Later *GlOSS* was generalized to *gGlOSS*[13] to handle the vector space information retrieval model. This generalization can be used for any IR model that computes a score to determine how well a document satisfies a query, provided that certain collection statistics can be made available to *gGlOSS*.

gGlOSS assumes that the databases can be characterized according to their *goodness* with respect to any particular query. *gGlOSS*'s job is then to estimate the goodness of each candidate database with respect to a particular query and then suggest a ranking of the databases according to the estimated goodness.

Goodness for each database, *db*, is defined as follows.

$$Goodness(l, q, db) = \sum_{d \in \{db | sim(q,d) > l\}} sim(q, d) \qquad (1)$$

where $sim(q, d)$ is a function that calculates the similarity between a query q and a document d. Once $Goodness(l, q, db)$ has been calculated for each database db with respect to q at threshold l, the ideal rank for the query at threshold l, $Ideal(l)$ can be formed by sorting the databases in descending order of their goodness.

Note that *gGlOSS* does not compute $Ideal(l)$ rather it is advanced as the goal to which *gGlOSS* estimated ranks $Max(l)$ and $Sum(l)$, defined in [13], will be compared. In French *et al.*[9] we showed that *gGlOSS* $Max(l)$ and $Sum(l)$ estimators do a good job of estimating $Ideal(l)$. We also showed that $Ideal(l)$ is not well-correlated to relevance.

Complete details for calculating the $Max(l)$ and $Sum(l)$ estimators are given in [13] and are not reproduced here. But, for later reference we note that

$$Max(0) = Sum(0) = Ideal(0), \qquad (2)$$

that is, at threshold $l = 0$ both estimators give identically the $Ideal(0)$ ranking of databases for all queries. In addition, $l = 0$ allows a consistent comparison of $Ideal(l)$ rankings when comparing different underlying retrieval systems that produce differently scaled similarity values (i.e. $sim(q, d)$ in Equation 1). Hence, in the evaluation to follow we will use $Ideal(0)$ as the *gGlOSS* estimate since *gGlOSS* can compute this exactly.

Note that *gGlOSS* needs two vectors of information from each database db_i in order to make its estimates.

1. the document frequency df_{ij} for each term t_j in db_i; and
2. the sum of the weight of each term t_j over all documents in db_i.

If the underlying database cannot be made to divulge this information directly, it is in principle possible to recover the information by issuing a single-term query for each vocabulary term. Our choice of $Ideal(0)$ obviates this; we can compute $Ideal(0)$ directly from the databases by simply issuing the test queries.

2.2.2 *CORI*

Given a set of databases to search, the *CORI* approach creates a *database selection index* in which each database is represented by its terms and their document frequencies *df*. Databases are ranked for a query q by a variant of the Inquery document ranking algorithm. The belief $p(r_k|db_i)$ in database db_i due to observing query term r_k is determined by:

$$\begin{aligned} T &= \frac{df}{df + 50 + 150 \cdot cw/\overline{cw}} \\ I &= \frac{\log\left(\frac{|DB|+0.5}{cf}\right)}{\log(|DB| + 1.0)} \\ p(r_k|db_i) &= 0.4 + 0.6 \cdot T \cdot I \qquad (3) \end{aligned}$$

where:

df	is the number of documents in db_i containing r_k,
cf	is the number of databases containing r_k,
$\lvert DB \rvert$	is the number of databases being ranked,
cw	is the number of words in db_i, and
$\overline{cw}$	is the mean cw of the databases being ranked.

The belief in a database depends upon the query structure, but is usually just the average of the $p(r_k|db_i)$ values for each query term [7].

The *CORI* approach to ranking collections can be summarized as $df.icf$, where icf is inverse collection frequency.

2.3 Controlling the Indexing Vocabulary

In [9] we prepared the test collection by using SMART version 11.0[3] using the same parameters as Gravano *et al.*[13]. Note that for those experiments each of the 236 sites used the same parameters and search engine (SMART) to process queries. Then we prepared a union vocabulary incorporating all the terms appearing at any of the separate collections. This gave us a canonical global vocabulary with which to store the document frequencies and weight sums required by *gGlOSS* to make its estimates.

To control the *CORI* evaluation properly it was necessary to guarantee that the same indexing vocabulary was used by *gGlOSS* and *CORI*. Sources of variability arise from: parsing the original TREC documents; tokenizing a document into words; stemming algorithm; and stoplist. To control for all these factors we synthesized pseudo-documents for each collection. We used the stemmed and stopped vocabulary from the SMART indexing done in [9], emitted each term tf times, and padded the document back to it original length with a fake word. Inquery was then run with stemming off and a single word stoplist. It was also necessary to handle special characters differently. The end result was that Inquery indexed exactly the same vocabulary that SMART did in the earlier experiments.

We verified that the steps we took to control the indexing vocabulary did not hinder the *CORI* performance. We ranked the collections using *CORI* with Inquery defaults then evaluated the ranking. The average performance results were nearly identical to the *CORI* results using the controlled vocabulary.

2.4 Evaluation—Baselines for Comparison

We refer to a number of baselines in the evaluation below, specifically: the *gGlOSS* baseline, $Ideal(0)$; the relevance-based ranking (RBR); and the size-based ranking (SBR). They are defined as follows.

$Ideal(0)$: This ranking is produced by processing each query at each of the 236 subcollections and then using the goodness (see Equation 1) to rank the subcollections.

RBR: These rankings were produced for each query by using the relevance judgements supplied with the TREC data. Databases are simply ordered by the number of relevant documents they contain.

SBR: Databases are ordered by the total number of documents they contain. Note that this ranking is constant for all queries.

2.5 Evaluation—Metrics for Comparison

As in our earlier work we use mean squared error, two recall metrics, $\mathcal{R}_n$ [13] and $\widehat{\mathcal{R}}_n$ [9], and a precision measure, $\mathcal{P}_n$ [13]. In addition we will also use the Spearman coefficient of rank correlation[12].

2.5.1 Recall and Precision Analogs

In [9] we discussed performance metrics that are analogous to the well known IR metrics of *recall* and *precision*. We briefly review the metrics here.

We provide a *baseline ranking*, B, that represents agoal for each query. An algorithm produces some *estimated ranking* for the query, E, and our goal is to decide how well E approximates B. We assume that each database db_i in the collection has some merit, $merit(q, db_i)$, to the query q. The baseline is expressed in terms of this merit; the estimate is formed by implicitly or explicitly estimating merit.

Let db_{b_i} and db_{e_i} denote the database in the i-th ranked position of rankings B and E respectively. Let

$$B_i = merit\,(q, db_{b_i}) \text{ and } E_i = merit\,(q, db_{e_i}) \tag{4}$$

denote the merit associated with the i-th ranked database in the baseline and estimated rankings respectively. In the results that follow we have the following convention. For the $Ideal(l)$ calculations $merit(q, db) = Goodness(l, q, db)$; for RBR we define $merit(q, db)$ to be the number of relevant documents in db; and for SBR we define $merit(q, db)$ to be the total number of documents in db.

Gravano *et al.*[13] defined $\mathcal{R}_n$ as follows.

$$\mathcal{R}_n = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n} B_i}. \tag{5}$$

This is a measure of how much of the available merit in the top n ranked databases of the baseline has been accumulated via the top n databases in the estimated ranking.

An alternative definition[9] is given by

$$\widehat{\mathcal{R}}_n = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n^*} B_i} \tag{6}$$

where

$$n^* = \max k \text{ such that } B_k \neq 0. \tag{7}$$

Intuitively, n^* is the breakpoint between the useful and useless databases. The denominator is just the total merit contributed by all the databases that are useful to the query. Thus, $\widehat{\mathcal{R}}_n$ is a measure of how much of the total merit has been accumulated via the top n databases in the estimated ranking. Lu *et al.*[17] have also suggested using this measure.

Gravano *et al.*[13] have also proposed a precision-related measure, $\mathcal{P}_n$. It is defined as follows.

$$\mathcal{P}_n = \frac{|\{db \in Top_n(E) | merit(q, db) > 0\}|}{|Top_n(E)|} \tag{8}$$

This gives the fraction of the top n databases in the estimated ranking that have non-zero merit.

2.5.2 Spearman Coefficient of Rank Correlation

The Spearman coefficient of rank correlation, ρ, is given by

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} D_i^2}{n(n^2 - 1)} \tag{9}$$

where D_i is the difference in the i-th paired ranks. We have $-1 \leq \rho \leq 1$ where $\rho = 1$ when two rankings are in perfect agreement and $\rho = -1$ when they are in perfect disagreement.

3 Results

In the results that follow, we report the comparison of *gGlOSS* and *CORI*; we use the baseline $Ideal(0)$ to represent *gGlOSS*. For both algorithms, the evaluation was conducted using the full 236 subcollections of the testbed described earlier. We used the TREC topics 51-150 as the test query set. We examine potential reasons for differences in performance and discuss a heuristic for a lower bound on performance.

On the graphs in this section, we label each curve E.B where E is the database selection algorithm (estimator) employed and B is the baseline to which E is being compared. A curve labeled B.B, e.g, RBR.RBR, is intended to show the best possible behavior, that is, when the baseline is used to estimate itself.

3.1 *CORI* Evaluation

The first step was to evaluate the performance of the *CORI* database selection algorithm using the evaluation measures described in French *et al.*[9]. The evaluation measures include mean-squared error, a precision analog, $\mathcal{P}_n$ and two recall analogs, $\mathcal{R}_n$ and $\widehat{\mathcal{R}}_n$. *CORI* was evaluated using all four measures. For comparison, we also show the performance of *gGlOSS* $Ideal(0)$ [9] for all measures.

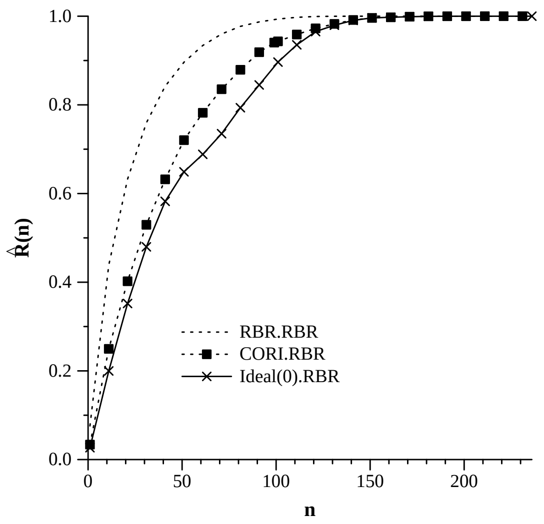

Figure 1: *CORI* results compared to *gGlOSS* and maximum achievable performance.

For these experiments, we used the RBR baseline as the standard against which to compare algorithm performance; the best possible performance under a given measure is to approximate RBR exactly.

We first considered the mean-squared error of the differences between the $Ideal(0)$ and *CORI* rankings and the RBR baseline. MSE can provide a general impression of the similarity of rankings on a query by query basis. The MSE values for both approaches were in the same range, with a minimum value of 940 and a maximum of 4942. However, the MSE values for *CORI* are lower on average with *CORI* having MSE values less than $Ideal(0)$ for 84% of the queries. This suggests that *CORI* may approximate RBR more closely than *gGlOSS*.

We then evaluated *CORI* using the $\mathcal{P}_n$, $\mathcal{R}_n$ and $\widehat{\mathcal{R}}_n$ measures. These measures can help reveal the nature and degree of the discrepancies shown by MSE. The results for $\widehat{\mathcal{R}}_n$ are shown in Figure 1; results for $\mathcal{P}_n$ and $\mathcal{R}_n$ are included in summary Figures 4 and 5.

In Figures 1, 2, 4, 5 and 6, the plots for $Ideal(0)$ and *CORI* should be compared to the plot RBR.RBR. For all measures *CORI* approximates the RBR baseline more closely than *gGlOSS* $Ideal(0)$.

3.2 $\widehat{\mathcal{R}}_n$ Difference

To get a different perspective on the impact of the performance difference, we examined the data plotted in Figure 1. Figure 1 displays the average performance over 100 queries of the $\widehat{\mathcal{R}}_n$ measure. This measure shows the rate at which relevant documents can be accrued as additional collections are considered. Another way to view this data is to determine how many collections from a ranking must be considered to reach a given value for $\widehat{\mathcal{R}}_n$. Table 1 shows the values for *CORI* and *gGlOSS* $Ideal(0)$ compared with the highest achievable value.

Examining Figure 1, it is readily apparent that fewer collections need be considered to reach each level of $\widehat{\mathcal{R}}_n$ when the *CORI* ranking was used than when the $Ideal(0)$ ranking was used. This is reinforced by Table 1. However, it is more apparent from Table 1 that the *CORI* ranking has a greater advantage for $\widehat{\mathcal{R}}_n$ values of 0.7 and above. If the goal is exhaustive searching, using the *CORI* ranking will be more efficient.

There are three regions of interest in Figure 1. These regions can also be seen in Figures 4, 5 and 6. For all measures, for n roughly less than 50, the performance of all of the algorithms is fairly close. For n greater than roughly 120, all algorithms are faced with a situation in which most of the relevant documents have been located and little improvement is possible. However for n in the range of 50-120, *CORI* has a more evident advantage, as seen in Table 1.

$\widehat{\mathcal{R}}_n$	Number of collections required to achieve $\widehat{\mathcal{R}}_n$		
	CORI	$Ideal(0)$	**Best Achievable**
0.1	4	5	2
0.2	9	12	4
0.3	15	18	7
0.4	21	25	10
0.5	29	33	14
0.6	38	43	19
0.7	49	64	26
0.8	64	83	36
0.9	86	102	52
1.0	209	225	145

Table 1: $\widehat{\mathcal{R}}_n$ for *CORI* and $Ideal(0)$

4 Analysis and Discussion

4.1 *CORI* and *gGlOSS* Comparison

The immediate question was why *CORI* approximated the RBR baseline more closely than *gGlOSS* $Ideal(0)$. Our initial observation was that the performance differences could be due to indexing differences. Our initial *gGlOSS* experiments were intended to reproduce the experimental setup of Gravano and Garcia-Molina [13]. As a result, we used the exact underlying retrieval engine and parameter settings reported—SMART using *ntc* weights for document terms and *nnn* weights for query terms.

It has been shown that using different weights and normalization approaches in SMART improves retrieval performance for the large TREC collections [4, 5, 19], so we considered that the improvement shown by *CORI* could be due simply to better-tuned indexing information. Therefore, we used the underlying Inquery collection indexes—used to generate the *CORI* inference net—to generate an additional $Ideal(0)$ baseline based upon the Inquery indexing information. We will refer to the new $Ideal(0)$ baseline as *Ideal(0)-inquery*. If *Ideal(0)-inquery* approximated RBR more accurately than $Ideal(0)$, it would imply that the indexing weights used by Inquery provided a better input to the *gGlOSS* algorithm. In this case, it would be more representative to compare the *CORI* performance to that of *Ideal(0)-inquery* than to that of $Ideal(0)$. The comparisons of $Ideal(0)$ and *Ideal(0)-inquery* to baseline RBR for evaluation measure $\widehat{\mathcal{R}}_n$ is shown in Figure 2. The comparisons for $\mathcal{P}_n$ and $\mathcal{R}_n$ are included in summary Figures 4 and 5.

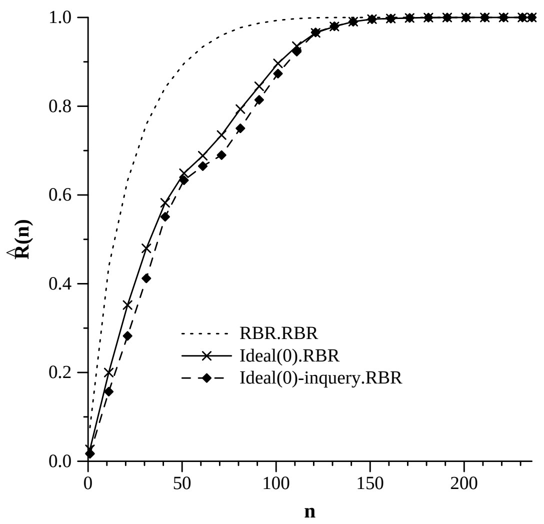

Figure 2: The two $Ideal(0)$s

Note that, in fact, the performance of *gGlOSS* declined when the Inquery weighting information was used. Therefore, we conclude that the improvement in performance of Inquery is not due to differences in term weighting. At present, we are not certain what aspect of the *CORI* database selection algorithm is responsible for its performance difference. We have conjectured that to *gGlOSS* a database with many documents of low similarity may appear more useful than a database with a few documents of high similarity [9]. It is possible that *CORI* is utilizing a better length normalization strategy that allows it to avoid this difficulty.

4.2 Formulation of SBR

Our 236 collection testbed is produced by partitioning the TREC data by publishing source, year and month. As a result, the number of documents in a collection varies widely for our testbed. During a detailed examination of results from our *gGlOSS* experiments [9], we noted some recurring themes in the rankings produced by $Ideal(0)$.

It appeared that $Ideal(0)$ routinely ranked highly collections with very large numbers of documents. An examination of the RBR baseline revealed that collections with a large number of documents also tended to have a good representation of relevant documents. Voorhees noticed a similar phenomenon in her work[22], specifically that the distribution of relevant documents was uneven across document sources. She noted that a strategy of retrieving from AP and WSJ was often very effective with TREC data. In our decomposition, subcollections derived from AP and WSJ tend to have a large number of documents.

We created a new baseline, size-based ranking (SBR), in which databases are ordered by the total number of documents that they contain. We used the SBR baseline to determine if a correlation between collection size and $Ideal(0)$ or RBR did in fact exist. We used Spearman's ρ to measure the correlation between $Ideal(0)$ and SBR and between RBR and SBR. The results are shown in Figure 3.

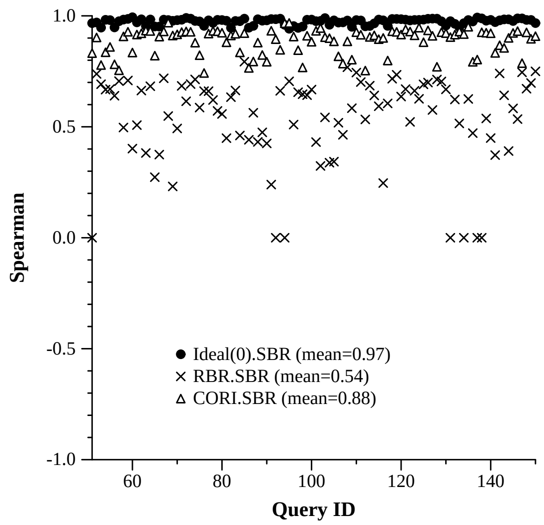

Figure 3: Spearman's ρ values for $Ideal(0)$ and RBR compared to SBR.

We noted that there is a very strong positive correlation between $Ideal(0)$ and SBR for all queries. The correlation between *CORI* and SBR is not as pronounced, but still strong. There is also a moderate positive correlation between RBR and SBR for most queries. SBR is a simple heuristic that is constant for all queries and therefore not at all computationally intensive. We will show later that SBR can closely approximate the performance of $Ideal(0)$ for this testbed. We consider SBR a useful lower bound on database selection performance for this collection.

4.3 Relationship between SBR and Ideal(0)

gGlOSS computes collection scores by summing over documents in a collection. As a result, *gGlOSS* favors collections with a large number of documents. For this testbed, the performance of *gGlOSS* $Ideal(0)$ is strongly influenced by this feature of the algorithm. $Ideal(0)$ is strongly positively correlated with SBR. Evidence of this correlation can be seen in Figures 4, 5 and 6. The approximation of RBR by $Ideal(0)$ closely tracks the approximation of RBR by SBR. However, $Ideal(0)$ does consistently outperform SBR.

Figures 4, 5 and 6 also show how all of the approaches discussed here approximate RBR. For all measures, *CORI* outperforms *gGlOSS*. Also note that SBR represents a useful lower bound for performance in this testbed.

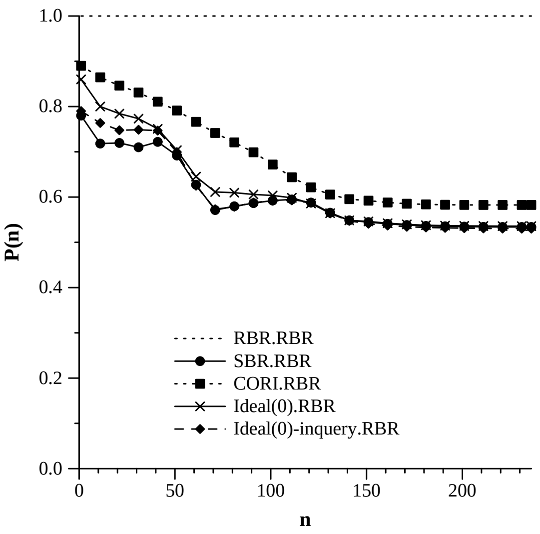

Figure 4: *CORI* results compared to *gGlOSS* and maximum achievable performance using $\mathcal{P}_n$.

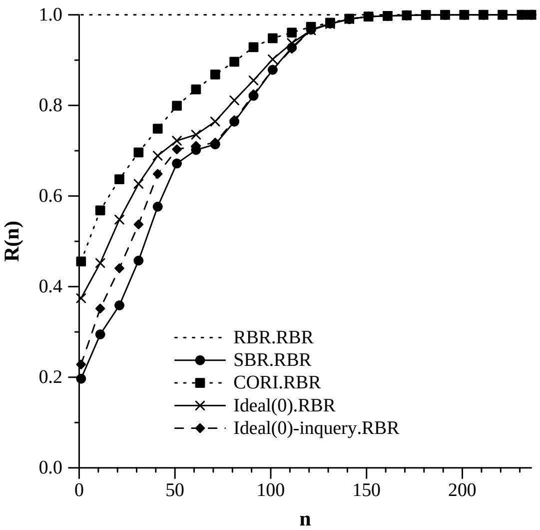

Figure 5: *CORI* results compared to *gGlOSS* and maximum achievable performance using $\mathcal{R}_n$.

The *CORI* and *Ideal*(0) plots appear similar in Figure 6, making it difficult to judge the performance difference between *CORI* and *Ideal*(0). Examining the plotted data, we determined that *CORI* shows at least 25% improvement over *Ideal*(0) for $n \leq 10$ and approximately 10% improvement for $n \leq 90$. We also note that *Ideal*(0) shows at least 20% improvment over SBR for $n < 50$; however, for $n > 50$, *Ideal*(0) tracks SBR much more closely.

Future experiments will need to examine the implications of the correlation between *Ideal*(0) and SBR. For this

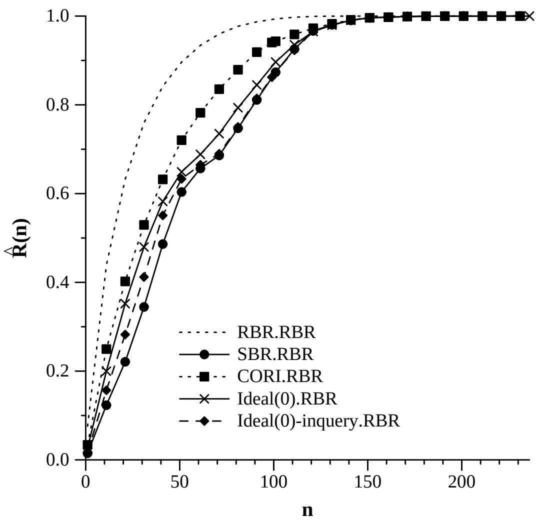

Figure 6: *CORI* results compared to *gGlOSS* and maximum achievable performance using $\widehat{\mathcal{R}}_n$.

testbed and the experiments reported here, it is not clear if the preference for large collections degraded the performance of *gGlOSS*; for this testbed there was also a positive correlation between SBR and RBR. However, given a testbed with a negative correlation between SBR and RBR, a preference for large collections could degrade the performance of *gGlOSS* or any other database selection algorithm with this feature.

In future experiments, we will create additional testbeds in which the number of documents in a collection will be held constant. This will allow us to examine algorithm performance when collection-size preference is controlled for.

5 Scaling Up Database Selection Algorithms

Database selection algorithms become increasingly necessary as the number of databases grows from $O(10)$, to $O(100)$, to $O(1,000)$ databases. Early research on database selection was based on testbeds of $O(10)$ databases [7, 15], and suggested that selection algorithms were very accurate. However, as the algorithms were improved and applied to testbeds of $O(100)$ databases, the experimental results became more ambiguous [13, 17, 24]. A high priority for our recent research has been to determine whether the *CORI* algorithm remains effective as the number of databases increases.

5.1 Testbeds

Experiments were conducted with the 236 collection testbed used throughout this paper, as well as two additional testbeds of 100 collections and 921 collections. Summary characteristics of these three testbeds are provided in Table 2.

The 100 collection testbed was created by dividing TREC CDs 1, 2, and 3 into 100 smaller databases of roughly equal size (about 30 megabytes each). Each database contained documents from a single source, ordered as they occur on the TREC CDs; hence documents in a database were also usu-

Num. DB	Source	Queries	Size (GB)	Total Docs
100	TREC 1+2+3	51-100	3.3	1,078,166
236	TREC 1+2+3 subset	51-150	2.7	691,058
921	TREC-6 VLC	301-350	20.1	7,492,048

Table 2: Characteristics of testbeds used in scaling experiments.

ally from similar timeframes. CD 1 contributed 37 databases, CD 2 contributed 27 databases, and CD 3 contributed 36 databases.

The 921 collection testbed was created by dividing the TREC-6 VLC corpus into 921 smaller databases of roughly equal size, subject to the same source and document adjacency constraints used for creating the 100 collection testbed.

Queries were based on TREC topics 51-150 and 301-350. We used query sets INQ001, INQ026, and INQ411, all created by the UMass CIIR as part of its participation in TREC and Tipster evaluations [1, 6]. Queries in these sets are long, complex, and have undergone automatic query expansion.

The relevance assessments were the standard TREC relevance assessments supplied by the U.S. National Institute for Standards and Technology. Relevance judgements for the VLC data are relatively sparse, so additional relevance judgements for 20 queries were gathered by trained undergraduates. The additional relevance judgements had little impact on results, and so were not used in the results reported here.

5.2 Experiments

The effectiveness of the *CORI* database selection algorithm was measured with $\mathcal{R}$ and $\widehat{\mathcal{R}}$, described above. In these tests, $\mathcal{R}$ and $\widehat{\mathcal{R}}$ were applied to the *percentage* of databases searched, so that experiments on testbeds of different sizes could be compared. We call these variations $\mathcal{R}(p)$ and $\widehat{\mathcal{R}}(p)$, where $p = n/N$, and N is the number of databases in the testbed.

5.3 Results

Figures 7 and 8 summarize the experimental results. Measurements with $\widehat{\mathcal{R}}(p)$ (Figure 7) show that on all testbeds the collections ranked highly by the *CORI* selection algorithm contain a large percentage of the relevant documents. This is an encouraging sign.

It may appear that the algorithm is *more* effective as more databases are available for searching, but this is an artifact of the distribution of relevant documents across the databases. Measurements with $\mathcal{R}$ (Figure 8), which normalizes $\widehat{\mathcal{R}}$ by the best result possible from searching that percentage of databases, show that effectiveness on the 100 and 921 collection testbeds is very similar. The only difference is when a very small percentage of collections is searched; in this case, effectiveness with the large testbed is substantially lower.

The 236 collection testbed appears to be a slightly easier testbed than the 100 collection and 921 collection testbeds,

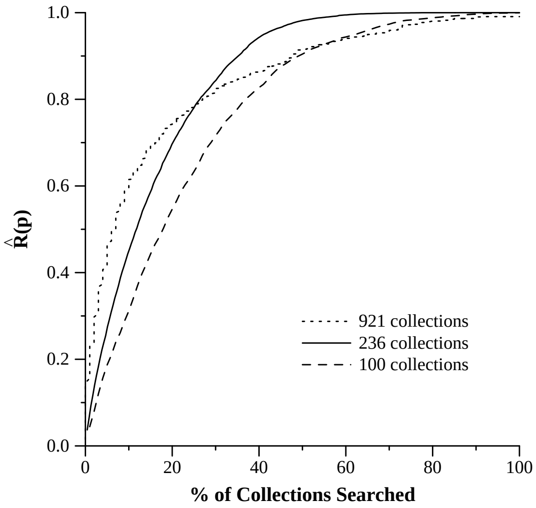

Figure 7: Database selection accuracy on testbeds of different sizes.

but the results are otherwise similar.

The effectiveness of the *CORI* algorithm appears to be related consistently to the percentage of databases searched, irrespective of the testbed size. This is not inappropriate for recall-oriented metrics such as $\widehat{\mathcal{R}}$ and $\mathcal{R}$, but it implies that a tenfold increase in available databases requires a tenfold increase in search effort to achieve a given level of recall. We would prefer a less linear relationship between testbed size and search effort, which suggests that additional research on database selection algorithms is required.

6 Conclusions

We have examined the performance of two database selection algorithms, *gGlOSS* and *CORI*. We have demonstrated the high correlation between $Ideal(l)$ ranks and the SBR baseline. This supports our conjecture[9] that to *gGlOSS* a database with very many documents of marginal similarity will appear, in aggregate, to be more useful than a database having a few documents with large similarity. We have shown that *CORI* is a better estimator of RBR than *gGlOSS* and is at least 10% more effective through $\widehat{\mathcal{R}}_n$ levels of 90%. In part this may be due to a better length normalization strategy so that *CORI* is not as confounded by databases with many documents of small similarity. We have also shown that the difference in performance between *CORI* and *gGlOSS* is not due to Inquery weighting versus SMART weighting.

In our experiments, the *CORI* approach required fewer databases to be searched than *gGlOSS* on average to achieve a specific recall level. This implies that *CORI* will be more cost-effective in large, distributed environments having lower latency for searches.

Our experiments investigating how well the *CORI* algorithm scales to large numbers of databases produced a more mixed message. The *CORI* selection algorithm produced consistently accurate rankings on testbeds ranging from 100

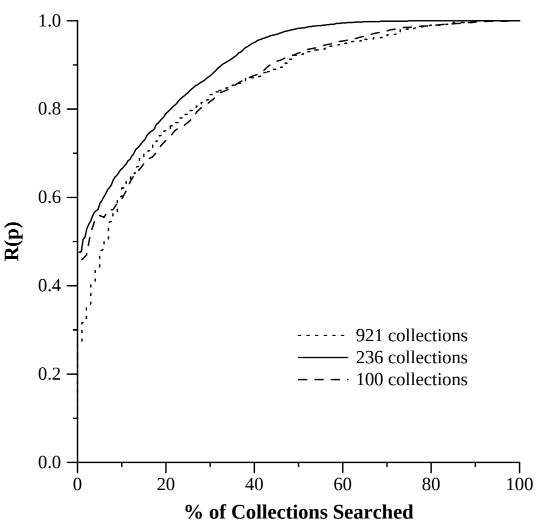

Figure 8: Scaled database selection accuracy on testbeds of different sizes.

to 921 collections. This is clearly a positive result. However, consistency was observed in the *percentage* of databases searched, not in the *number* of databases searched. For a given level of recall, search effort appears to scale linearly with the number of available databases. A less linear relationship is desirable, and perhaps required, in environments containing very large numbers of databases. Our results suggest that further research on database selection algorithms is required.

We have demonstrated a systematic methodology for examining database selection algorithms. This is the first study of this kind. We can now investigate other database selection algorithms in the same framework and be able to compare performance in a common experimental environment.

It will be instructive to repartition our testbed into a similar number of sites, but with each site having about the same number of documents. There are a number of ways to do this, but each has drawbacks. Such a partition would completely defeat the SBR baseline because every site would have equal merit and thus all sites would be equally useful under the SBR baseline. We think that this might be a more demanding search environment and offer greater insight into the behavior of these kinds of algorithms.

Acknowledgements. We thank Margaret Connell for her assistance with experiments reported here.

References

[1] J. Allan, J. P. Callan, W. B. Croft, L. Ballesteros, D. Byrd, R. Swan, and J. Xu. INQUERY does battle with TREC-6. In *Proc. TREC-6*.

[2] N. J. Belkin, P. Kantor, E. A. Fox, and J. A. Shaw. Combining the Evidence of Multiple Query Representations for Informa tion Retrieval. *Information Processing & Management*, 31(4):431–448, 1995.

[3] C. Buckley. SMART version 11.0, 1992. ftp://ftp.cs.cornell.edu/pub/smart.

[4] C. Buckley, J. Allan, and G. Salton. Automatic routing and ad-hoc retrieval using SMART : TREC 2. In *Proc. TREC-2*, pages 45–56, 1994.

[5] C. Buckley, G. Salton, and J. Allan. Automatic retrieval with locality information using SMART. In *Proc. TREC-1*, pages 59–72, 1993.

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

[7] J. P. Callan, Z. Lu, and W. B. Croft. Searching Distributed Collections with Inference Networks. In *Proc. SIGIR'95*, pages 21–29, 1995.

[8] E. A. Fox, M. P. Koushik, J. Shaw, R. M. odlin, and D. Rao. Combining Evidence from Multiple Searches. In *Proc. TREC-1*, pages 319–328, 1992.

[9] J. C. French, A. L. Powell, C. L. Viles, T. Emmitt, and K. J. Prey. Evaluating Database Selection Techniques: A Testbed and Experiment. In *Proc. SIGIR'98*, pages 121–129, 1998.

[10] J. C. French and C. L. Viles. Ensuring Retrieval Effectiveness in Distributed Digital Libraries. *Journal of Visual Communication and Image Representation*, 7(1):61–73, 1996.

[11] N. Fuhr. A Decision-Theoretic Approach to Database Selection in Networked IR. *ACM Trans. on Information Systems.* to appear.

[12] J. D. Gibbons. *Nonparametric Methods for Quantative Analysis.* Holt, Rinehart and Winston, 1976.

[13] L. Gravano and H. Garcia-Molina. Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies. In *Proc. VLDB'95*, 1995.

[14] L. Gravano, H. Garcia-Molina, and A. Tomasic. GlOSS: Text-source discovery over the internet. *ACM Trans. on Database Systems*, To appear.

[15] L. Gravano, H. Garcia-Molina, and A. Tomasic. The Effectiveness of GlOSS for the Text Database Discovery Problem. In *Proc. SIGMOD'94*, pages 126–137, May 1994.

[16] D. Harman. Overview of the Fourth Text Retrieval Conference (TREC-4). In *Proc. TREC-4*, 1996.

[17] Z. Lu, J. P. Callan, and W. B. Croft. Measures in collection ranking evaluation. Technical Report TR-96-39, Computer Science Department, University of Massachusetts, 1996.

[18] A. Moffat and J. Zobel. Information Retrieval Systems for Large Document Collections. In *Proc. TREC-3*, pages 85–94, 1995.

[19] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proc. SIGIR'96*, pages 21–29, 1996.

[20] C. L. Viles and J. C. French. TREC4 Experiments Using Drift. In *Proc. TREC-4*, 1996.

[21] C. L. Viles and J. C. French. Dissemination of Collection Wide Information in a Distributed Inform ation Retrieval System. In *Proc. SIGIR'95*, pages 12–20, July 1995.

[22] E. Voorhees, N. Gupta, and B. Johnson-Laird. Learning Collection Fusion Strategies. In *Proc. SIGIR'95*, pages 172–179, 1995.

[23] E. Voorhees, N. Gupta, and B. Johnson-Laird. The Collection Fusion Problem. In *Proc. TREC-3*, pages 95–104, 1995.

[24] J. Xu and J. Callan. Effective Retrieval with Distributed Collections. In *Proc. SIGIR'98*, pages 112–120, 1998.

[25] B. Yuwono and D. Lee. Server Ranking for Distributed Text Retrieval Systems on Internet. In *Proc. of the Int. Conf. on Database Systems for Adv. Applications*, pages 41–49, 1997.