Metrics for Evaluating Database Selection Techniques *

James C. French Allison L. Powell †
Department of Computer Science
University of Virginia
Charlottesville, VA
{french|alp4g}@cs.virginia.edu

Technical Report CS-99-19
June 18, 1999

Abstract

The increasing availability of online databases and other information resources in digital libraries has created the need for efficient and effective algorithms for selecting databases to search. A number of techniques have been proposed for query routing or database selection. We have developed a methodology and metrics that can be used to directly compare competing techniques. They can also be used to isolate factors that influence the performance of these techniques so that we can better understand performance issues. In this paper we describe the methodology we have used to examine the performance of database selection algorithms such as gGlOSS and CORI. In addition we develop the theory behind a "random" database selection algorithm and show how it can be used to help analyze the behavior of realistic database selection algorithms.

1 Introduction

As information resources proliferate on internets and intranets, algorithms for database selection, distributed searching and results merging become increasingly important. Several investigators have proposed solutions for the problem of database selection, but in general it is not possible to compare the results of these inquiries in a meaningful way. Nor is it the case that there is any reasonable expectation of what good performance would be. In particular, there has been no discussion of how these methods compare to an algorithm that selects databases at random. We investigate this issue in this paper. We characterize expected behavior for an algorithm that creates rankings randomly and we make concrete what it means to perform better than such an algorithm with respect to specific metrics.

Distributed searching can be decomposed into three fundamental activities: (1) choosing the specific database collections to search; (2) searching the chosen databases; and (3) merging the results into a cohesive response. Although there is considerable interest in all these aspects, we focus specifically on the first activity. Callan *et al.*[1] call this the *collection selection* problem while Gravano *et al.*[7] refer to it as the *text database resource discovery* problem. In prior work [5, 4, 3] we referred to this as *database selection* and we will retain that term here for consistency.

Both Callan *et al.*[1] and Gravano *et al.*[6] formulate the problem similarly. They first assume that they have a collection of databases that are candidates for search.Then, given a query, they *rank* the database collection, that is, they decide in what order to search the databases in the collection. There is some preferred order in which to search the collection, but the nature of that preferred order is cast differently by different researchers. Gravano *et al.*[6] used so-called $ideal(l)$ ranks as their standard while Callan *et al.*[1] used so-called $optimal$ ranks. The specific nature of these standard baselines is unimportant here; we are interested only in methodology for calibrating performance against some prespecified baseline.

Finally, there is an evaluation phase of the work in which the predicted ranks for queries are compared with the preferred orderings to decide how well the particular ranking methodology worked. Unfortunately, the nature of this comparison also differs from research group to research group. This point will be developed more fully in the section on evaluation below. The goal of the work reported here

*A shorter version of this paper appeared in the International Workshop on Internet Data Management (IDM'99) at the 10th International Conference and Workshop on Database and Expert Systems Applications (DEXA'99)[2].

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

is to provide a consistent framework for direct comparison of competing methods of database selection.

2 Evaluation Methodology

2.1 The Problem

To state the problem formally, we have a set of databases $DB = \{db_1, db_2, ..., db_N\}$ that we wish to search to satisfy some query q. We assume that each database db has some merit, denoted $merit(q, db)$, with respect to the query. We would like to search the databases in order of decreasing merit to the query.

Given DB, q, and $merit(q, db)$, there exists at least one permutation of the $db \in DB$, $\langle db_{i_1}, db_{i_2}, ..., db_{i_N} \rangle$ such that

$$merit(q, db_{i_j}) \geq merit(q, db_{i_{j+1}}) \quad (1)$$

where $j = 1, 2, ..., N-1$.

We will call a permutation satisfying property 1 a *baseline*.

2.2 Issues in Comparison

Given some goal baseline B and an algorithm producing an estimate, E, of that goal, we endeavor to determine the quality of the estimate by means of some measure $m(E, B)$ comparing the estimate to the goal.

Since $m(E, B)$ is a comparison of E to B, we might like to evaluate a particular algorithm from multiple viewpoints. In that case E is constant but B or $m(E, B)$ might change. This gives rise to three situations that must be considered.

1. Multiple measures, single baseline:

   $$m_1(E, B) \text{ vs. } m_2(E, B).$$

   This is probably the most common situation. Here we are evaluating the estimate against the baseline using multiple yardsticks. This is often useful to expose different aspects of the phenomena under study.

2. Single measure, multiple baselines:

   $$m(E, B_1) \text{ vs. } m(E, B_2).$$

   One example of this situation is when the baseline is parameterized and varying the baseline parameters while holding the measure constant is the technique being used.

3. Multiple measures, multiple baselines:

   $$m_1(E, B_1) \text{ vs. } m_2(E, B_2).$$

   This situation occurs when measuring against multiple baselines and no single measure is appropriate for all baselines.

There is no standardization in the approach taken here either. Evaluations reported in the literature are quite idiosyncratic, compounding the problem of comparing methods for collection selection.

2.3 Properties of Measures

It is often useful in an evaluation to understand the properties of the measures being employed. At the very least we want to know:

1. the value of the measure when the baseline is applied to itself, $m(B, B)$;

2. the value of the measure when the worst configuration of the baseline, say $\bar{B}$, is applied to the baseline, i.e., $m(\bar{B}, B)$; and

3. whether the measure is symmetric, $m(X, Y) = m(Y, X)$.

Other properties might be important as well. To be useful a ranking measure should also have the following property:

$$m(\bar{B}, B) \leq m(E, B) \leq m(B, B).$$

$m(\bar{B}, B)$ is the operational lower bound on performance. If a ranking measure $m(X, Y) \geq 0$ and satisfies this property then it can be normalized to be in $[0, 1]$.

2.4 Properties of Rankings

There are two situations under which we would say that two rankings E_1 and E_2 are *equivalent*. The two scenarios are:

1. equivalence classes of databases, i.e., one or more databases have identical merit; and

2. when two rankings have the same value under the measure, i.e., $m(E_1, B) = m(E_2, B)$.

In principle these two situations might have different implications. In practice, equivalence (1) above should imply (2), but the converse need not hold.

2.5 Some Examples of Baselines for Comparison

There are many possible baselines that could be used for performance evaluation. We discuss four here.

Count-Based Ranking (CBR): The baseline is constructed by ordering the databases in decreasing order of the number of records contained in the database that satisfy a Boolean predicate.

$Ideal(l)$: In [6] Gravano *et al.* assume that (a) all databases employ the same weighting strategies and similarity algorithm; and (b) the only documents in a database that are useful to a query q are those with a similarity greater than some threshold l. With these assumptions in mind they define

$$Goodness(l, q, db) = \sum_{d \in Rank(l,q,db)} sim(q, d)$$

where

$$Rank(l, q, db) = \{d \in db | sim(q, d) > l\}.$$

The $Ideal(l)$ rank is then formed by sorting the databases with respect to their goodness to q.

Relevance-Based Ranking (RBR): The baseline is constructed by ordering the databases in decreasing order of the number of relevant records contained in the database.

Size-based Ranking (SBR): Databases are ordered by the total number of documents they contain. Note that this ranking is constant for all queries.

There are many other possible rankings. Those listed above are representative of rankings that have been reported in the literature. For example, CBR was used by Gravano *et al.*[7] for evaluating *GlOSS*; $Ideal(l)$ was used by Gravano *et al.*[6] to report their performance evaluation of *gGlOSS*; RBR was used by Callan *et al.*[1] for their evaluation of *CORI* and by French *et al.*[5, 4, 3] in their investigations; SBR was used by French *et al.*[4, 3] to isolate and study a bias toward large collections that is exhibited by some database selection algorithms.

Note that when evaluating a set of queries Q, we will generally have a separate baseline instance for each $q \in Q$. SBR is one exception to this.

2.6 General Evaluation Strategy

Most evaluations are conducted over a query set Q and result in aggregate summary measures of the following form.

$$\frac{1}{|Q|} \sum_{q \in Q} m(E_q, B_q)$$

Note that $m(E_q, B_q)$ might itself already be an aggregate measure.

2.7 Specific Metrics for Comparison

There is no general agreement on how this type of comparison should be done. The general problem is that we are given a baseline ranking for some query and a ranking produced by some collection selection algorithm. The goal is to decide how well the candidate ranking approximates the baseline ranking. We describe some of the approaches given in the literature and discuss new measures here.

2.7.1 Mean Squared Error

Callan *et al.*[1] reported their comparisons using mean squared error of the predicted ranks and the desired ranks. So given a collection of N databases to rank for any candidate ranking we compute

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \left(base\_rank(db_i) - est\_rank(db_i)\right)^2$$

where $base\_rank(db_i)$ is the baseline or desired rank and $est\_rank(db_i)$ is the predicted rank for db_i.

This is a widely used measure of dispersion but does not give us any real intuition about ranking quality. We might benefit here from a comparison with worst case behavior. For any baseline B, the worst case configuration, $\bar{B}$, is simply the reverse ranking. We can calculate the sum of squared differences as follows.

$$\begin{aligned} (n-1)^2 + (n-3)^2 &+ (n-5)^2 + \cdots \\ \cdots &+ (3-n)^2 + (1-n)^2 \\ &= \sum_{i=1}^{n} [n - (2i-1)]^2 \\ &= \frac{n(n^2-1)}{3} \end{aligned} \quad (2)$$

We can use Eq. 2 to normalize the sum of squared differences and that might be more instructive.

2.7.2 Recall and Precision Analogs

In this section we discuss performance metrics that are analogous to the well known IR metrics of $recall$ and $precision$. We begin by introducing some terminology and notation that tries to make this analysis neutral and generalizes it to include a variety of baselines.

Recall that for each query we provide a *baseline ranking*, say B, that represents a desired goal or query plan. Given some algorithm that produces an *estimated ranking*, E, our goal is to decide how well E approximates B.

To begin, we assume that each database db in the collection has some merit, $merit(q, db)$, to the query q. We expect the baseline to be expressed in terms of this merit; we expect the estimated ranking to be formulated 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

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

denote the merit associated with the i-th ranked database in the baseline and estimated rankings respectively. The total merit, M, is given by $M = \sum_{i=1}^{N} B_i$

We note that for viable baseline rankings it should always be the case that

$$B_i \geq B_{i+1},\ i = 1...N-1.$$

For the baselines discussed here this is always true because we assume that the baseline ranking is determined by sorting the databases in decreasing order of merit for some appropriate definition of merit. However, it is not generally the case that $E_i \geq E_{i+1}$. The performance evaluation problem discussed here is an attempt to quantify the degree to which this is true for any estimated ranking.

This point needs a bit of explanation. Note that E_i is the *actual* merit associated with db_i. The estimators will rank databases in decreasing order of *estimated* merit. The degree to which $E_i \geq E_{i+1}$ reflects the accuracy of the algorithm's estimates of E_i.

Gravano *et al.*[6] defined $\mathcal{R}_n$ as follows.

$$\mathcal{R}_n(E,B) = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n} B_i}. \quad (3)$$

$\mathcal{R}_n(E,B)$ 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. This is a variant of the *normalized cumulative recall* measure defined by Tomasic *et al.*[12] and later generalized by Gravano *et al.*[8].

We propose an alternative definition of a recall-like measure that can be used to present performance results. First we need one more definition. Let

$$n^* = k \text{ such that } B_k \neq 0 \text{ and } B_{k+1} = 0.$$

Intuitively, n^* is the ordinal position in the ranking of the last database with non-zero merit; it is the breakpoint between the useful and useless databases. Clearly $n^* \leq N$ and, moreover, the total merit, M, of a baseline B is given by $M = \sum_{i=1}^{n^*} B_i$. With this definition we define our alternative "recall" metric as follows.

$$\widehat{\mathcal{R}}_n(E,B) = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n^*} B_i}. \quad (4)$$

The denominator is just the total merit contributed by all the databases that are useful to the query. Thus, $\widehat{\mathcal{R}}_n(E,B)$ is a measure of how much of the total merit has been accumulated via the top n databases in the estimated ranking. This measure has also been proposed by Lu *et al.*[11] and was used to report results by French *et al.*[5, 4, 3].

These two measures are clearly related. Since

$$\mathcal{R}_n(E,B)\sum_{i=1}^{n} B_i = \widehat{\mathcal{R}}_n(E,B)\sum_{i=1}^{n^*} B_i, \quad (5)$$

we have $\mathcal{R}_n(E,B) \geq \widehat{\mathcal{R}}_n(E,B)$ and $\mathcal{R}_{n^*}(E,B) = \widehat{\mathcal{R}}_{n^*}(E,B)$. (Note that in the remainder of the paper we will simplify the notation as follows: $\mathcal{R}_{n^*}(E,B) = \mathcal{R}_*$ and similarly for $\mathcal{P}_*$.)

Theorem 1 $\widehat{\mathcal{R}}_n(E,B) = \mathcal{R}_n(E,B) \cdot \widehat{\mathcal{R}}_n(B,B)$

Proof: Follows directly from Equation 5.

From the theorem we can see that another way to interpret $\mathcal{R}_n(E,B)$ is to regard it as the rate at which the available baseline merit is being accumulated.

Gravano *et al.*[6] have also proposed a precision-related measure, $\mathcal{P}_n(E,B)$. It is defined as follows.

$$\mathcal{P}_n(E,B) = \frac{|\{db \in Top_n(E) | merit(q,db) > 0\}|}{|Top_n(E)|} \quad (6)$$

This gives the fraction of the top n databases in the estimated ranking that have non-zero merit. ($Top_n(E)$ is just the set of databases given in the first n ranks.)

Some additional properties follow.

1. $\mathcal{R}_n(E,B) \leq \mathcal{R}_n(B,B)$ and $\mathcal{P}_n(E,B) \leq \mathcal{P}_n(B,B)$
2. $\mathcal{R}_n(B,B) = 1$ and $\mathcal{P}_n(B,B) = 1$
3. $\widehat{\mathcal{R}}_n(B,B) \leq 1$

In the remainder of the paper we simplify the notation by dropping all arguments to the measures when it is clear that we are referring to a specific algorithm's estimates (E) and measuring against a prespecified baseline (B).

3 Randomly Generated Rankings

Another approach to evaluating database selection algorithms is to ask how they compare to randomly generated rankings. Losee[10] has suggested evaluating IR system performance analytically. This approach can be extended to database selection algorithms and lets us derive the expected performance of an algorithm that generates rankings randomly. We can then use this result to examine the behavior of other algorithms specifically to see if they have better performance than an algorithm that generates rankings randomly. We develop these ideas further in this section.

Definition 1 *Given a collection of databases, $\{db_1, db_2, ..., db_N\}$, a random ranking algorithm is one in which each of the $N!$ permutations is equally likely.*

Lemma 1 *Given a random ranking of N databases, let X_n denote the number of databases in the first n ranks having nonzero merit. X_n is a hypergeometric random variable with expected value given by $E[X_n] = n \cdot n^*/N$.*

Proof: This follows from the theorem in the appendix where $M = N$, and $W = n^*$.

Theorem 2 *The expected value of the precision, $E[\mathcal{P}_n]$, of the first n elements of a randomly generated ranking is given by*

$$E[\mathcal{P}_n] = \frac{n^*}{N}.$$

Proof: Given a random ranking, let X_n denote the number of databases in the first n ranks having nonzero merit. Then

$$\mathcal{P}_n = \frac{X_n}{n} \text{ so that } E[\mathcal{P}_n] = \frac{E[X_n]}{n}.$$

The result now follows directly from Lemma 1.

We would expect a good ranking algorithm to have significantly greater precision.

Corollary 2.1 $E[\mathcal{P}_*] = \frac{n^*}{N}$.

We have fully characterized $\mathcal{P}_n$ for randomly generated rankings. Now let's take a look at $\mathcal{R}_n$ and $\widehat{\mathcal{R}}_n$ for these rankings.

Condition 1 *The total merit M is spread evenly over all n^* databases.*

The net effect of this condition is to make all rankings with $\mathcal{P}_* = 1$ equivalent.

Theorem 3 *Let X_n denote the number of databases having nonzero merit in the first n ranks of a randomly generated ranking. If Condition 1 holds then*

1. $\mathcal{R}_n = \mathcal{P}_n$
2. $E[\mathcal{R}_n] = n^*/N$
3. $\widehat{\mathcal{R}}_n = X_n/n^*$
4. $E[\widehat{\mathcal{R}}_n] = n/N$

Proof: Condition 1 implies that each database with nonzero merit contributes M/n^* merit to the accumulated total.
Part 1:

$$\mathcal{R}_n = \frac{X_n \cdot (M/n^*)}{n \cdot (M/n^*)} = \frac{X_n}{n} = \mathcal{P}_n.$$

Part 2: Taking expectations from Part 1 we have $E[\mathcal{R}_n] = E[\mathcal{P}_n]$. The result now follows directly from Theorem 2.
Part 3:

$$\widehat{\mathcal{R}}_n = \frac{X_n \cdot (M/n^*)}{M} = \frac{X_n}{n^*}.$$

Part 4: From Part 3 and Lemma 1 we have

$$E[\widehat{\mathcal{R}}_n] = \frac{E[X_n]}{n^*} = \frac{n}{N}.$$

Corollary 3.1 *If Condition 1 holds then for all rankings in which $\mathcal{P}_n = 1$, we have $\mathcal{R}_n = 1$ and $\widehat{\mathcal{R}}_n = n/n^*$, $n \le n^*$.*

Corollary 3.2 *If Condition 1 holds then*

$$\widehat{\mathcal{R}}_n = \frac{n}{n^*}\mathcal{P}_n = \frac{n}{n^*}\mathcal{R}_n$$

where $n \le n^*$.

Proof: From Theorem 3(3)and Theorem 3(1).

Theorem 4 $E[\widehat{\mathcal{R}}_n] = \frac{n}{N}$

Proof:

$$\begin{aligned} E[\widehat{\mathcal{R}}_n] &= \frac{\text{expected merit}}{\text{total merit}} \\ &= \frac{n \cdot \text{expected merit per database}}{M} \\ &= \frac{n\left(\frac{M}{N}\right)}{M} = \frac{n}{N} \end{aligned}$$

Note that this is a stronger result than Theorem 3(4) since it does not require Condition 1. So Theorem 4 can be used to determine the expected value of $\widehat{\mathcal{R}}_n$ for an arbitrary baseline.

Corollary 4.1 $E[\mathcal{R}_*] = E[\widehat{\mathcal{R}}_*] = E[\mathcal{P}_*] = \frac{n^*}{N}$.

Proof: By definition $\mathcal{R}_* = \widehat{\mathcal{R}}_*$, so by Theorem 4 $\mathcal{R}_* = \widehat{\mathcal{R}}_* = \frac{n^*}{N}$. Together with Corollary 2.1 this completes the proof.

Note that Corollary 4.1 explains why the $(\mathcal{R}_*, \mathcal{P}_*)$ scatter plots tend to cluster around the line $y = x$. (See, for example, Figure 5 in French *et al.*[5].)

Theorem 5 $E[\mathcal{R}_n] = \frac{M}{\sum_{i=1}^{n} B_i} E[\widehat{\mathcal{R}}_n]$

Proof: From Theorem 1 we have

$$\widehat{\mathcal{R}}_n(E,B) = \mathcal{R}_n(E,B) \cdot \widehat{\mathcal{R}}_n(B,B).$$

Taking expectations yields

$$E[\widehat{\mathcal{R}}_n(E,B)] = \widehat{\mathcal{R}}_n(B,B) \cdot E[\mathcal{R}_n(E,B)]$$

since $E[cX] = cE[X]$. The desired result follows from $\widehat{\mathcal{R}}_n(B,B) = \frac{\sum_{i=1}^{n} B_i}{M}$.

The following corollary to Theorem 5 is immediate from Theorem 4.

Corollary 5.1 $E[\mathcal{R}_n] = \frac{nM}{N\sum_{i=1}^{n} B_i}$

4 An Example

In this section we show the performance of several database selection techniques, evaluated using the measures described in this paper. The plots are taken from French *et*

al.[4], a recent performance study of database selection algorithms. Figures 1, 2 and 3 contain data from Figures 4, 5 and 6 of [4], respectively.

Here, we include a new comparison of that data to the randomly generated ranking approach introduced in this paper.

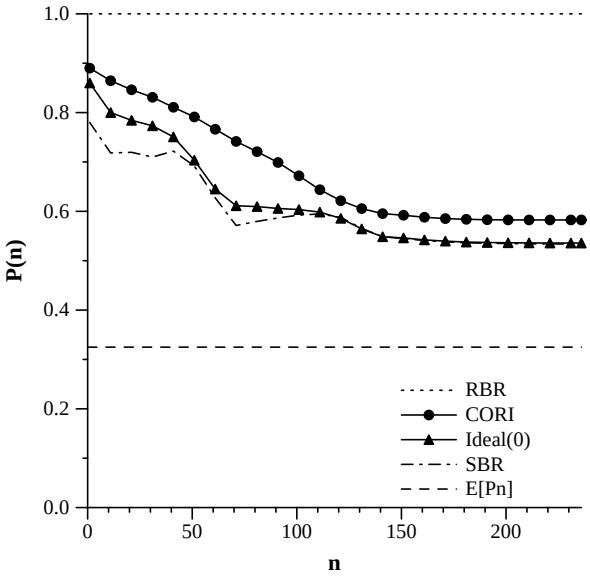

Figure 1. Comparison of approaches using $\mathcal{P}_n$.

In Figures 1, 2 and 3, we show the performance of the *CORI* and *gGlOSS* algorithms in relation to the best possible performance (labelled as RBR), the performance of the SBR baseline and the expected performance of a randomly generated ranking. The results shown are averaged over 100 queries. On average, *CORI*, *gGlOSS* and the SBR baseline all outperform the expected performance of a randomly generated ranking by a large margin. However, it is also illustrative to consider query-by-query performance.

Using the same data that was used to generate Figure 3, we compared the algorithms to the expected value of $\widehat{\mathcal{R}}_n$ given randomly generated rankings. For each n Table 1 records the number of queries for which each algorithm performed worse than the expected value given randomly generated rankings. Since we evaluated 100 queries at each n, the values in the table may be interpreted as the percentage of queries for which the algorithm performed worse than the random algorithm. In [3] we concluded that for low values of n there was not much difference in the performance of the competing algorithms. This conclusion was reached by examining Figure 3 and Figure 2. The table tells a different story and implies that *CORI* has fewer failures than the other algorithms tested. This is an additional data point for consideration when evaluating such algorithms.

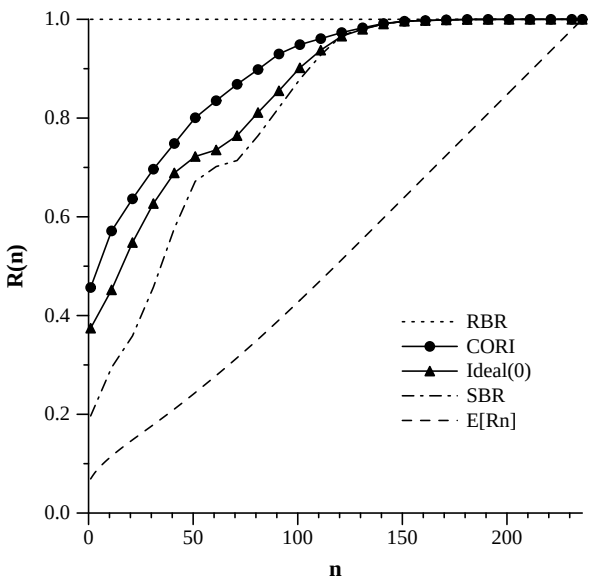

Figure 2. Comparison of approaches using $\mathcal{R}_n$.

The table was truncated at $n = 32$. $Ideal(0)$ had a small number of failures through $n = 73$, while SBR continued through $n = 85$. No algorithm had a failure beyond $n = 85$.

5 Conclusions

We have described metrics and a methodology for comparing database selection algorithms. This methodology has been used in earlier studies [5, 4, 3]. The main contribution of this paper was to introduce the notion of randomly generated rankings and to demonstrate how they can be used to establish an operational lower bound on performance. We derived the expected behavior of all our performance measures and gave an explicit example using $\widehat{\mathcal{R}}_n$ to demonstrate how a comparison could be done. Clearly an effective database selection algorithm should outperform randomly generated rankings in most cases particularly when useful data is unevenly distributed across database sites. However, there is at least one case when one could only hope to do as well as a randomly generated ranking and that is when all the useful data is uniformly distributed across all the databases. In that case every ranking is as good as every other, so no particular ranking would be preferred. By this argument, our analysis also gives a tight lower bound for expected behavior of database selection algorithms.

Appendix

The following is taken from Larson[9] but can be found in any introductory probability textbook.

Definition 2 *An urn contains M balls of which W are white. Let X denote the number of white balls that occur*

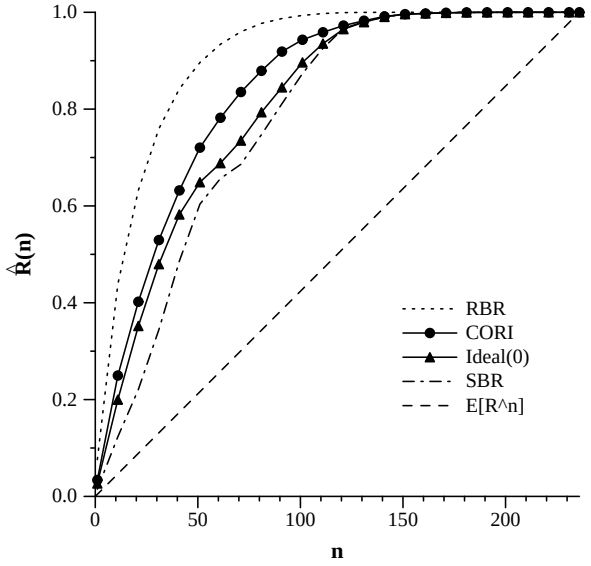

Figure 3. Comparison of approaches using $\widehat{\mathcal{R}}_n$.

in a sample of n balls drawn at random from the urn without replacement. X is called the hypergeometric random variable.

Theorem 6 *If X is the hypergeometric random variable, then*

$$p_X(k) = \frac{\left(\begin{array}{c} W \\ k \end{array}\right)\left(\begin{array}{c} M-W \\ n-k \end{array}\right)}{\left(\begin{array}{c} M \\ n \end{array}\right)}, k = 0, 1, ..., n.$$

The mean, μ_X, and variance, σ_X^2, of the hypergeometric random variable X are given by

$$\mu_X = \frac{nW}{M}$$

and

$$\sigma_X^2 = \frac{nW}{M} \cdot \frac{M-W}{M} \cdot \frac{M-n}{M-1}$$

respectively.

Acknowledgement. We would like to thank Travis Emmitt for a careful reading of an early draft of this paper.

References

[1] J. P. Callan, Z. Lu, and W. B. Croft. Searching Distributed Collections with Inference Networks. In *Proceedings of the 18th International Conference on Research and Development in Information Retrieval*, pages 21–29, 1995.

[2] J. C. French and A. L. Powell. Metrics for Evaluating Database Selection Techniques. In *Proc. International Workshop on Internet Data Management (IDM'99) at the 10th International Conference and Workshop on Database and Expert Systems Applications (DEXA'99)*, 1999.

n	**SBR**	$Ideal(0)$	*CORI*	**RBR**
1	38	19	13	0
2	33	16	8	0
3	31	15	8	0
4	25	14	5	0
5	22	12	3	0
6	19	11	3	0
7	20	9	4	0
8	17	8	4	0
9	14	5	3	0
10	16	6	3	0
11	13	5	2	0
12	13	5	1	0
13	14	4	1	0
14	11	4	0	0
15	10	4	0	0
16	11	4	0	0
17	12	4	0	0
18	11	4	0	0
19	11	4	0	0
20	11	4	0	0
21	10	4	0	0
22	11	4	0	0
23	10	3	0	0
24	5	3	0	0
25	6	3	0	0
26	6	3	0	0
27	6	3	0	0
28	8	3	0	0
29	8	3	0	0
30	7	3	0	0
31	6	3	0	0
32	7	3	0	0

Table 1. Number of times each algorithm achieved a lower value of $\widehat{\mathcal{R}}_n$ than a random algorithm.

[3] J. C. French, A. L. Powell, and J. Callan. Effective and Efficient Automatic Database Selection. Technical Report CS-99-08, Department of Computer Science, University of Virginia, February 1999.

[4] J. C. French, A. L. Powell, J. Callan, C. L. Viles, T. Emmitt, K. J. Prey, and Y. Mou. Comparing the Performance of Database Selection Algorithms. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval*, 1999. To appear.

[5] J. C. French, A. L. Powell, C. L. Viles, T. Emmitt, and K. J. Prey. Evaluating Database Selection Techniques: A Testbed and Experiment. In W. B. Croft, A. Moffat, and C. J. van Rijsbergen, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 121–129, Melbourne, Australia, 24-28 August 1998.

[6] L. Gravano and H. Garcia-Molina. Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies. In *Pro-*

ceedings of the 21st International Conference on Very Large Databases (VLDB), Zurich, Switzerland, 1995.

[7] L. Gravano, H. Garcia-Molina, and A. Tomasic. The Effectiveness of GlOSS for the Text Database Discovery Problem. In *SIGMOD94*, pages 126–137, Minneapolis, MN, May 1994.

[8] L. Gravano, H. Garcia-Molina, and A. Tomasic. Precision and Recall of GlOSS Estimators for Database Discovery. In *Proceedings of the 3rd International Conference on Parallel and D istributed Information Systems*, pages 103–106, Austin, TX, September 1994.

[9] H. J. Larson. *Introduction to Probability Theory and Statistical Inference, (2nd. edition)*. John Wileyi & Sons, Inc., 1974.

[10] R. M. Losee. Determining Information Retrieval and Filtering Performance without Experimentation. *Information Processing & Management*, 31(4):555–572, 1995.

[11] 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.

[12] A. Tomasic, L. Gravano, C. Lue, P. Schwarz, and L. Haas. Data Structures for Effi cient Broker Implementation. *ACM Transactions on Information Systems*, 15(3):223–253, July 1997.