Precision and Recall of *GlOSS* Estimators for Database Discovery

Luis Gravano* Héctor García-Molina† Anthony Tomasic‡

Abstract

The availability of large numbers of network information sources has led to a new problem: finding which text databases (out of perhaps thousands of choices) are the most relevant to a query. We call this the text-database discovery problem. Our solution to this problem, *GlOSS*–Glossary-Of-Servers Server, keeps statistics on the available databases to decide which ones are potentially useful for a given query. In this paper we present different query-result size estimators for *GlOSS* and we evaluate them with metrics based on the precision and recall concepts of text-document information-retrieval theory. Our generalization of these metrics uses different notions of the set of relevant databases to define different query semantics.

1 Introduction

On-line information vendors offer access to multiple databases. In addition, the advent of a variety of INTERNET tools [SEKN92, ODL93] has provided easy, distributed access to many more databases. The result is thousands of text databases from which a user may choose for a given information need (a user query). This paper presents a framework for (and analyzes a solution to) this problem, which we call the *text-database discovery problem.*

Our solution to the text-database discovery problem is to build a service that can suggest potentially good databases to search. Then, a user's query will go through two steps: first, the query is presented to our server (dubbed *GlOSS*, for **Gl**ossary-**O**f-**S**ervers **S**erver) to select a set of promising databases to search. During the second step, the query is actually evaluated at the chosen databases. As an intermediate step, *GlOSS* could show the chosen databases to the user, who would in turn select which ones to actually search. *GlOSS* gives a hint of what databases might be useful for the user's query, based on word-frequency information for each database. This information indicates, for each database and each word in the database vocabulary, how many documents at that database actually contain the word. For example, a Computer-Science library could report that the word *Knuth* occurs in 180 documents, the word *computer*, in 25,548 documents, and so on. This information is orders of magnitude smaller than a full index (see [GGMT94]) since for each word we only need to keep its frequency, as opposed to the identities of the documents that contain it.

Example 1.1 Consider three databases, A, B, and C, and suppose that *GlOSS* has collected the statistics of Figure 1. If *GlOSS* receives a query q=“*find retrieval* $\wedge$ *discovery*” (this query searches for documents that contain both words, “*retrieval*” and “*discovery*”), *GlOSS* has to estimate the

*Stanford University, Computer Science Dept., Margaret Jacks Hall, Stanford, CA 94305–2140. E-mail: `gravano@cs.stanford.edu`. Phone: (415) 723-0872. Fax: (415) 725-2588.

†Stanford University, Computer Science Dept., Margaret Jacks Hall, Stanford, CA 94305–2140. E-mail: `hector@cs.stanford.edu`

‡Princeton University, Department of Computer Science. Current Address: Stanford University, Computer Science Dept., Margaret Jacks Hall, Stanford, CA 94305–2140. E-mail: `tomasic@cs.stanford.edu`

number of matching documents in each of the three databases. Figure 1 shows that database C does not contain any documents with the word "*discovery*," and so, there cannot be any documents in C matching query q. For the other two databases, *GlOSS* has to "guess" what the number of documents matching query q is. There are different ways in which this can be done. An *estimator* for *GlOSS* uses the *GlOSS* information to make this guess. One of the estimators for *GlOSS* that we study in this paper, *Ind*, estimates the result size of the given query in each of the databases in the following way. Database A contains 100 documents, 40 of which contain the word "*retrieval*." Therefore, the probability that a document in A contains the word "*retrieval*" is $\frac{40}{100}$. Similarly, the probability that a document in A contains the word "*discovery*" is $\frac{5}{100}$. Under the assumption that words appear independently in documents, the probability that a document in database A has both the words "*retrieval*" and "*discovery*" is $\frac{40}{100} \times \frac{5}{100}$. Consequently, we can estimate the result size of query q in database A as $f(q, A) = \frac{40}{100} \times \frac{5}{100} \times 100 = 2$ documents. Similarly, $f(q, B) = \frac{500}{1000} \times \frac{40}{1000} \times 1000 = 20$, and $f(q, C) = \frac{10}{200} \times \frac{0}{200} \times 200 = 0$.

The *Ind* estimator chooses those databases with the highest estimates as the databases where to direct the given query. So, *Ind* will return $\{B\}$ as the answer to q (see Figure 2). This may or may not be a "correct" answer, depending on different factors. Firstly, it is possible that some of the result-size estimates given by *Ind* are wrong. For example, it could be the case that database B did not contain any matching document for q, while *Ind* predicted there would be 20 such documents in B. Furthermore, if database A did contain matching documents, then *Ind* would fail to pick any database with matching documents (since its answer was $\{B\}$).

Secondly, even if the estimates given by *Ind* are accurate, the correctness of the produced answer depends on the user's *semantics* for the query. Assume in what follows that the result-size estimates given above are correct (i.e., there actually are two documents matching query q in database A, 20 in database B, and none in database C). Given a query q and a set of databases, the user may be interested in one out of (at least) two different sets of databases over which to evaluate query q:

- *Matching*, the set of all of the databases containing matching documents for the query. For the sample query, this set is $\{A, B\}$, whereas *Ind* produced $\{B\}$ as its answer. Therefore, if the semantics intended by the query submitter are "recall oriented," in the sense that *all* of the databases in *Matching* should be searched, then *Ind*'s answer is not correct. Such a user is interested in getting exhaustive answers to the queries. (Section 7.2 presents the *Bin* estimator, aimed at addressing these semantics.) If, on the other hand, the intended semantics are "precision oriented," in the sense that *only* databases in *Matching* should be searched, then *Ind*'s answer is correct. In this case, the user is in "sampling" mode, and simply wants to obtain *some* matching documents, without searching useless databases.

- *Best*, the set of all of the databases containing more matching documents than any other database. Searching these databases yields the highest payoff (i.e., the largest number of documents). For the sample query, this set is $\{B\}$, which is also the answer produced by *Ind*. Again, users might be interested in emphasizing "precision" or "recall," in the sense described for the *Matching*-set case. □

To evaluate the set of databases that *GlOSS* returns for a given query, we present a framework based on the precision and recall metrics of information-retrieval theory. In that theory, for a given query q and a given set S of relevant documents for q, *precision* is the fraction of documents in the answer to q that are in S, and *recall* is the fraction of S in the answer to q. We borrow these notions to define metrics for the text-database discovery problem: for a given query q and a given set of "relevant databases" S, P is the fraction of databases in the answer to q that are in S, and

Database	A	B	C
Number of documents	100	1000	200
Number of documents with the word "*retrieval*"	40	500	10
Number of documents with the word "*discovery*"	5	40	0

Figure 1: A portion of the database frequency information that *GlOSS* keeps for three databases.

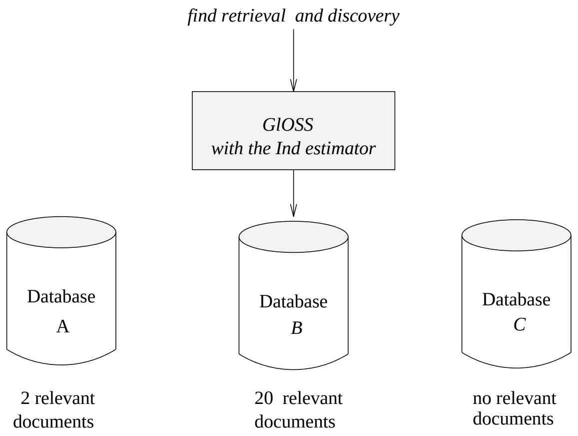

Figure 2: The *Ind* estimator for *GlOSS* chooses the most promising databases for a given query. In the example, database B, which is actually the database containing the highest number of matching documents, is chosen.

R is the fraction of S in the answer to q. We further extend our framework by offering different definitions for a "relevant database." We have performed experiments using query traces from the FOLIO library information-retrieval system at Stanford University, and involving six databases available through FOLIO. As we will see, the results obtained for *GlOSS* and several estimators are very promising. Even though *GlOSS* keeps a small amount of information about the contents of the available databases, this information proved to be sufficient to produce very useful hints on where to search.

Another advantage of *GlOSS* is that its frequency information can be updated mechanically, that is, sources can periodically extract word counts and send them to *GlOSS*. Other approaches (see Section 2) require human-generated summaries of the contents of a database, and are prone to errors or very out-of-date information. Also, *GlOSS*'s storage requirements are low: a rough estimate suggested that 22.29 MBytes were enough to keep all of the data needed by *GlOSS* for the six databases we studied [GGMT94], or only 2.15% of the estimated size of a full index of the six databases. Therefore, it is straightforward to replicate the service at many sites. Thus, a user may be able to consult *GlOSS* at the local machine or cluster, and immediately determine the candidate databases for a given query.

In [GGMT94] we focused on the storage requirements of *GlOSS*. We also studied the performance of a single estimator for *GlOSS*, namely the *Ind* estimator, for a set of evaluation criteria different from the P and R parameters we use in this paper. The contributions of this paper are:

- The definition of *Min* and *Bin*, two new estimators that *GlOSS* may use for making decisions.
- An experimental evaluation of *GlOSS* according to modified precision and recall parameters

from information-retrieval theory. This evaluation uses:

  - several different query semantics, corresponding to different definitions of the "right set" of databases where to evaluate each query, and
  - two different real-user query traces.

- The introduction of a more-flexible version of the *Ind* estimator, and of more-flexible evaluation metrics.

Of course, *GlOSS* is not the only solution to the text-database discovery problem, and in practice we may wish to combine it with other complementary strategies. These strategies are described in Section 2. Incidentally we note that, to the best of our knowledge, experimental evaluations of these other strategies *for the text-database discovery problem* are rare: in most cases, strategies are presented with no statistical evidence as to how good they are at locating sites with documents of interest *for actual user queries*. Thus, we view the experimental methodology and results of this paper (even though they still have limitations) as an important contribution to this emerging research area.

Section 3 introduces *GlOSS* and the concept of an *estimator*. In particular, Section 3.4 describes *Ind*, the first estimator for *GlOSS* that we will evaluate in the paper. Section 4 defines our evaluation metrics, based on the precision and recall parameters [SM83]. Section 5 describes the experiments performed to assess the effectiveness of *GlOSS*. Section 5.3 identifies three different "right" sets of databases where users might want to evaluate their queries. Section 6 reports the experimental results, including experiments on two query traces to assess how dependent our results are on a specific query trace (Section 6.4). Section 7.1 introduces variants to *Ind* and to our evaluation metrics. Section 7.2 presents *Min* and *Bin*, two new estimators for *GlOSS*. Finally, Section 8 summarizes our work.

2 Related work

Many solutions have been presented recently for the text-database discovery problem, or, more generally, for the resource-discovery problem: the text-database discovery problem is a subcase of the resource-discovery problem, since the latter generally deals with a larger variety of types of information [SEKN92, ODL93].

One solution to the text-database discovery problem is to let the database selection be driven by the user. Thus, the user will be aware of and an active participant in this selection process. Different systems follow different approaches to this: one such approach is to let users "browse" through information about the different databases. Examples include Gopher [SEKN92], where users navigate through the network following a hierarchy of indexes, and World-Wide Web [BLCGP92], which uses a hypertext interface to do this. Various search facilities are being created for these systems, like the Veronica Service [Fos92] for Gopher, for example. The Prospero File System [Neu92] lets users organize information available in the INTERNET through the definition (and sharing) of customized views of the different objects and services available to them.

A different approach is to keep a database of "meta-information" about the available databases and have users query this database to obtain the set of databases to search. For example, WAIS [KM91] provides a "directory of servers." This "master" database contains a set of documents, each describing (in English) the contents of a database on the network. The users first query the master database, and once they have identified potential databases, direct their query to these databases. One disadvantage is that the user typically needs two different queries. Also, the master-database documents have to be written by hand to cover the relevant topics, and have to be manually kept up to date as the underlying database changes. However, freeWAIS [FW+93] automatically adds the 50 most frequently occurring words in an information server to the associated description in the directory of servers. Another drawback is that in general, databases containing relevant documents might be missed if they are not chosen during the database-selection phase. [DS93] shows sample queries for which very few of the existing relevant servers are found by querying the WAIS directory of servers (e.g., only 6 out of 223 relevant WAIS servers).

[Sch90] follows a probabilistic approach to the resource-discovery problem. A resource-discovery protocol is presented that conceptually consists of two phases: a dissemination phase, during which information about the contents of the databases is replicated at randomly chosen sites, and a search phase, where several randomly chosen sites are searched in parallel. Also, sites are organized into "specialization subgraphs." If one node of such a graph is reached during the search process, the search proceeds "non-randomly" in this subgraph, if it corresponds to a specialization relevant to the query being executed. See also [Sch93].

In Indie (shorthand for "Distributed Indexing") [DLO92], information is indexed by "Indie brokers," each of which has associated, among other administrative data, a boolean query (called a "generator rule"). Each broker indexes (not necessarily local) documents that satisfy its generator rule. Whenever a document is added to an information source, the brokers whose generator rules match the new document are sent a descriptor of the new document. The generator objects associated with the brokers are gathered by a "directory of servers," that is queried initially by the users to obtain a list of the brokers whose generator rules match the given query. See also [DANO91]. [SA89], [BC92], and [OM92] are other examples of this type of approach in which users query "meta-information" databases.

A "content-based routing" system is used in [SDW+94] to address the database-discovery problem. The "content routing system" keeps a "content label" for each information server (or collection of objects, more generally), with attributes describing the contents of the collection. Users assign values to the content-label attributes in their queries until a sufficiently small set of information servers is selected. Also, users can browse the possible values of each content-label attribute.

The WHOIS++ directory service [WS93] organizes the WHOIS++ servers into a distributed "directory mesh" that can be searched: each server automatically generates a "centroid" listing the words it contains (for the different attributes). Centroids are gathered by index servers, that in turn must generate a centroid describing their contents. The index server centroids may be passed to other index servers, and so on. A query that is presented to an index server is forwarded to the (index) servers whose centroids match the query.

The master-database idea can be enhanced if we can use the semantics of queries and databases. In particular, assume we can automatically extract the semantic "concepts" involved in a user query. Also assume that we can extract the semantic concepts appearing in a collection of documents (in a database). Assuming that the number of concepts is much smaller than the number of words appearing in documents, then the concepts can be used for distributed indexing. That is, the user query is processed to extract the concepts; these are matched against the set of concepts and the potential sites identified. With our sample query "*find retrieval* $\wedge$ *discovery*," the processing could extract the concept *computer science* and the index would determine that documents on this concept appear in the Computer Science and the Medical databases. A related approach has been followed in [GS93].

In [FY93], every site keeps statistics about the type of information it receives along each link connecting to other sites. When a query arrives in a site, it is forwarded through the most promising link according to these statistics.

References [MTD92], [MDT93], and [ZC92] follow an expert-systems approach to solving the related problem of selecting online business databases.

Other approaches to solving the resource-discovery problem guarantee exhaustive answers to the users' queries. For example, the archie system [ED92] periodically obtains a recursive listing of the contents of all the available FTP sites in order to answer users' queries.

A complementary approach to *GlOSS* is taken by Chamis [Cha88]. Briefly, the approach this paper takes is to expand a user query with thesaurus terms. The expanded query is compared with a set of databases, and the query terms with exact matches, thesauri matches, and "associative" matches are counted for each database. Each database is then ranked as a function of these counts. We believe that this approach is complementary in its emphasis on thesauri to expand the meaning of a user query.

3 *GlOSS*: Glossary-Of-Servers Server

Consider a query q (permissible queries are defined in Section 3.1) that we want to evaluate over a set of databases DB. *GlOSS* selects a subset of DB consisting of "good candidate" databases for actually submitting q. To make this selection, *GlOSS* uses an *estimator* (Section 3.3), that assesses how "good" each database in DB is with respect to the given query, based on the word-frequency information on each database (Section 3.2).

3.1 Query representation

In this paper, we will only consider *boolean "and"* queries, that is, queries that consist of positive atomic subqueries connected by the boolean "and" operator (denoted as "$\wedge$" in what follows). An atomic subquery is a *keyword field-designation* pair. An example of a query is:

"find author Knuth $\wedge$ subject computer."

This query has two atomic subqueries: "*author Knuth*" and "*subject computer.*" In "*author Knuth,*" *author* is the field designation, and *Knuth* the corresponding keyword [1].

We consider only boolean queries so far because this model is used by library systems and information vendors worldwide. Also, the system we had available to perform our experiments uses only boolean queries (see Section 5.1). Nevertheless, we should stress that we can generalize this paper's approach to the vector-space retrieval model [SM83][2]. The reason why we restrict our study to "and" queries is that we want to understand a simple case first. Also, most of the queries in the trace we studied (see Section 5.1) are "and" queries. However, we can extend our approach to include "or" queries in a variety of different ways. For example, in [GGMT94] we analyze a limited form of "or" queries, showing how *GlOSS* can handle this type of queries.

3.2 Database word-frequency information

GlOSS keeps the following information about the databases:

- $DBSize(db)$, the total number of documents in database db, $\forall\ db \in DB$, and
- $freq(t, db)$, the number of documents in db that contain t, $\forall\ db \in DB$, and for all keyword field-designation pairs t. Note that *GlOSS* does not have available the actual "inverted lists"

[1] Uniform field designators for all the databases we considered (see Section 5.1) were available for our experiments. However, *GlOSS* does not rely completely on this, and could be adapted to the case where the field designators are not uniform across the databases, for example.

[2] For example, we could extend *GlOSS* with a "dot product" estimator that would produce a ranking of all databases given a query.

corresponding to each keyword-field pair and each database, but just the length of these inverted lists. The value $freq(t, db)$ is the size of the result of query "*find t*" in database *db*.

If $freq(t, db) = 0$, *GlOSS* does not need to store this explicitly, of course. Therefore, if *GlOSS* finds no information about $freq(t, db)$, then $freq(t, db)$ will be assumed to be 0.

A real implementation of *GlOSS* would require that each database cooperate and periodically submit these frequencies to the *GlOSS* server following some predefined protocol.

In [GGMT94], we modify the frequency information that *GlOSS* keeps on each database so as to reduce its size.

3.3 Estimators

Given $freq$ and $DBSize$ for a set of databases DB, *GlOSS* uses an *estimator* EST to select the set of databases to which to submit the given query. An estimator consists of a function ESize_{EST} that estimates the result size of a query in each of the databases, and a "matching" function (the max function below) that uses these estimates to select the set of databases ($Chosen_{EST}$ below) to which to submit the query. Once $\mathsf{ESize}_{EST}(q, db)$ has been defined, we can determine $Chosen_{EST}(q, DB)$ in the following way:

$$\begin{aligned} Chosen_{EST}(q, DB) \;\; = \;\; & \{db \in DB \mid \mathsf{ESize}_{EST}(q, db) > 0 \wedge \\ & \mathsf{ESize}_{EST}(q, db) = \max_{db' \in DB} \mathsf{ESize}_{EST}(q, db')\} \end{aligned} \tag{1}$$

Equation 1 may seem targeted to identifying the databases containing the *highest* number of matching documents. However, Section 7.2 shows how we can define $\mathsf{ESize}_{EST}(q, db)$ so that $Chosen_{EST}(q, db)$ becomes the set of *all* of the databases potentially containing matching documents, when we present the *Bin* estimator. Instances of ESize_{EST} are given in Sections 3.4 and 7.2, while a different "matching" function is used in Section 7.1.

3.4 The *Ind* estimator

This section describes *Ind*, the estimator that we will use for most of our experiments. *Ind* (for "independence") is an estimator built upon the (possibly unrealistic) assumption that keywords appear in the different documents of a database following independent and uniform probability distributions. Under this assumption, given a database *db*, any n keyword field-designation pairs $t_1, \ldots, t_n$, and any document $d \in db$, the probability that d contains all of $t_1, \ldots, t_n$ is:

$$\frac{freq(t_1, db)}{DBSize(db)} \times \ldots \times \frac{freq(t_n, db)}{DBSize(db)}$$

So, according to *Ind*, the estimated number of documents in *db* that will satisfy the query "*find* $t_1 \wedge \ldots \wedge t_n$" is [SFV83]:

$$\mathsf{ESize}_{Ind}(find\ t_1 \wedge \ldots \wedge t_n, db) \;\; = \;\; \frac{\prod_{i=1}^{n} freq(t_i, db)}{DBSize(db)^{n-1}} \tag{2}$$

The $Chosen_{Ind}$ set is then computed with Equation 1. Thus, *Ind* chooses those databases with the *highest* estimates (as given by ESize_{Ind}).

To illustrate these definitions, let DB ={INSPEC, PSYCINFO} (INSPEC and PSYCINFO are databases that we will use in our experiments, see Section 5). Also, let:

$$q = find\ author\ D.\ Knuth \wedge title\ computer$$

	INSPEC	PSYCINFO
$DBSize(\_)$	1,416,823	323,952
$freq(author\ D.\ Knuth, \_)$	13	0
$freq(title\ computer, \_)$	24,086	2704

Figure 3: Information *Ind* needs for DB = {INSPEC, PSYCINFO} and q= "*find author D. Knuth* $\wedge$ *title computer.*"

Figure 3 shows the statistics available to *Ind*. From this, *Ind* computes: $\mathsf{ESize}_{Ind}(q, \text{INSPEC}) = \frac{13 \times 24{,}086}{1{,}416{,}823} \simeq 0.22$. Incidentally, the actual result size of the query q in INSPEC, $\mathsf{RSize}(q, \text{INSPEC})$, is one document.

Since "*D. Knuth*" is not an author in the PSYCINFO database, and due to the boolean semantics of the query representation, the result size of query q in the PSYCINFO database must be zero. This agrees with what Equation 2 predicts: $\mathsf{ESize}_{Ind}(q, \text{PSYCINFO}) = \frac{0 \times 2704}{323{,}952} = 0$. This holds in general for boolean queries: if $freq(t_i, db) = 0$ for some $1 \leq i \leq n$, then

$$\mathsf{ESize}_{Ind}(find\ t_1 \wedge \ldots \wedge t_n, db) = \mathsf{RSize}(find\ t_1 \wedge \ldots \wedge t_n, db) = 0$$

As we have seen, when all frequencies are non-zero, ESize_{Ind} can differ from RSize. Section 6.1 analyzes how well ESize_{Ind} approximates RSize.

To continue with our example, since DB ={INSPEC, PSYCINFO}, and INSPEC is the only database with a non-zero result-size estimate, as given by ESize_{Ind}, it follows that $Chosen_{Ind}(q, DB)$ = {INSPEC}. So, *Ind* chooses the only database in the pair that might contain some matching document for q. In fact, since $\mathsf{RSize}(q, \text{INSPEC}) = 1$, *Ind* succeeds in selecting the only database that actually contains a document matching query q.

4 Evaluation parameters

Let DB be a set of databases and q a query. In order to evaluate an estimator EST, we need to compare its prediction against what actually is $Right(q, DB)$, the "right subset" of DB to query. There are several notions of what the right subset means, depending on the semantics the query submitter has in mind. Section 5.3 examines some of these options. For example, $Right(q, DB)$ can be defined as the set of all the databases in DB that contain documents that match query q. Once we have defined the $Right$ set for a query q and a database set DB, we evaluate how well $Chosen_{EST}(q, DB)$ approximates $Right(q, DB)$. (In general, we will drop the parameters of functions when this will not lead to confusion. For example, we refer to $Right(q, DB)$ as $Right$, whenever q and DB are clear from the context.)

To evaluate $Chosen_{EST}$, we adapt the well-known *precision* and *recall* parameters from information-retrieval theory [SM83] to the text-database discovery framework. If we regard $Right$ as the set of "items" (databases in this context) that are relevant to a given query q, and $Chosen_{EST}$ as the set of items that is actually retrieved, we can define the following functions P^{EST}_{Right} and R^{EST}_{Right}, based upon the precision and recall parameters:

$$P^{EST}_{Right}(q, DB) = \begin{cases} \frac{|Chosen_{EST}(q,DB) \cap Right(q,DB)|}{|Chosen_{EST}(q,DB)|} & \text{if } |Chosen_{EST}(q, DB)| > 0 \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

$$R^{EST}_{Right}(q, DB) = \begin{cases} \frac{|Chosen_{EST}(q,DB) \cap Right(q,DB)|}{|Right(q,DB)|} & \text{if } |Right(q, DB)| > 0 \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

Database	$DBSize$	Area
INSPEC	1,416,823	Physics, Elect. Eng., Computer Sc., etc.
COMPENDEX	1,086,289	Engineering
ABI	454,251	Business Periodical Literature
GEOREF	1,748,996	Geology and Geophysics
ERIC	803,022	Educational Materials
PSYCINFO	323,952	Psychology

Figure 4: Summary of the characteristics of the six databases considered.

Intuitively, P is the fraction of selected databases that are *Right* ones, and R is the fraction of the *Right* databases that are selected. For example, suppose that the set of databases is $DB = \{A, B, C\}$, and that for a given query q, $Right(q, DB)$ is defined to be $\{A, B\}$ (this could be the case if only A and B contained documents matching query q, as in Example 1). Furthermore, if $Chosen_{EST}(q, DB) = \{B\}$, then $P^{EST}_{Right}(q, DB) = 1$, since the only chosen database, B, is in the *Right* set. On the other hand, $R^{EST}_{Right}(q, DB) = 0.5$, since only half of the databases in *Right* are included in $Chosen_{EST}$.

Note that $P^{EST}_{Right}(q, DB) = 1$ whenever $Chosen_{EST} = \emptyset$, to capture the fact that no database in $Chosen_{EST}$ is not in *Right*. Similarly, $R^{EST}_{Right}(q, DB) = 1$ if $Right = \emptyset$, since all of the *Right* databases are included in $Chosen_{EST}$.

Different users will be interested in different semantics for the queries. One way to define different semantics is through the definition of *Right* (see Section 5.3). Even for a fixed *Right* set of databases, some users may be interested in emphasizing "precision" (databases not in *Right* should be avoided, even if this implies missing some of the "right" databases), while some others may want to emphasize "recall" (at least all of the databases in *Right* should be included in the answer to query q). Therefore, high values of P^{EST}_{Right} should be the target in the former case, and high values of R^{EST}_{Right} in the latter.

In this paper, we evaluate different estimators in terms of the average value, over a set of user queries, of the P and R parameters defined above, for different *Right* sets of databases.

5 Experimental framework

In order to evaluate the performance of different *GlOSS* estimators according to the P and R parameters of Section 4, we performed experiments using query traces from the FOLIO library information-retrieval system at Stanford University.

5.1 Databases and the INSPEC query trace

Stanford University provides on-campus access to its information-retrieval system FOLIO from terminals in libraries and from workstations via `telnet` sessions. FOLIO gives access to several databases. Figure 4 summarizes some characteristics of the six databases we chose for our experiments. Six is a relatively small number, given our interest in exploring hundreds of databases. However, we were limited to a small number of databases by their accessibility and by the high cost of our experiments. Thus, our results will have to be taken with caution, indicative of the *potential* benefits of this type of estimators.

A trace of all user commands for the INSPEC database was collected from 4/12/1993 to 4/25/1993. This set of commands contained 8392 queries. As discussed in Section 3.1, we only considered correctly formed "and" queries[3]. Also, we did not consider the so-called "phrase" queries (e.g., "*find titlephrase knowledge bases*"). The final set of queries, $TRACE_{INSPEC}$, has 6897 queries, or 82.19% of the original set.

5.2 Database-frequency-information construction

In order to perform our experiments, we evaluated each of the $TRACE_{INSPEC}$ queries in the six databases described in Figure 4. This is the data we need to build the different $Right$ sets (see Section 5.3) for each of the queries.

Also, to build the database word-frequency information needed by *GlOSS* (Section 3.2) we evaluated, for each query of the form *find* $t_1 \wedge \ldots \wedge t_n$, the n queries *find* $t_1, \ldots,$ *find* t_n in each of the six databases. Note that the result size of the execution of *find* t_i in database db is equal to $freq(t_i, db)$ as defined in Section 3. This is exactly the information an estimator EST needs to define $Chosen_{EST}$, for each query in $TRACE_{INSPEC}$[4]. It should be noted that this is just the way we gathered the data in order to perform our experiments. An actual implementation of such a system would require that each database communicate the number of postings for each word to *GlOSS*.

5.3 Different "right" sets of databases

Section 4 introduced the notion of the $Right$ set of databases for a given query. Different definitions for the $Right$ set determine different instantiations of the P and R parameters defined by Equations 3 and 4. To illustrate the issues involved in determining $Right$, consider the following example:

Example 5.1 Figure 5 shows three databases: A, B, and C. Consider a query q issued by a user. Each database produces a set of matching documents as the answer to q. Figure 5 shows that database A gives document 4 as the answer to q, database B, documents 5, 6, and 7, and database C, documents 8 and 9. Also, each database contains a set of documents that are relevant to the user that issued query q, that is, are actually of interest to the user. These documents may or may not match the answer to q. Thus, database A has three relevant documents: documents 1, 2, and 3, database B has one relevant document: document 5, and database C has two relevant documents: documents 8 and 9. Furthermore, assume that the user is interested in evaluating the query in one database only. The question is how to define the $Right$ set given this scenario. There are three alternatives:

- $Right = \{A\}$, since A is the database with the highest number of documents (three) relevant to the user's information need. However, the answer produced by database A when presented with query q consists of document 4 only, which is not a relevant document. Therefore, the user would not benefit from the fact that A contains the highest number of relevant documents among the three available databases, making this definition for $Right$ not very useful.

- $Right = \{C\}$, since C is the database that produces the highest number of relevant documents in the answer to query q. This is an interesting definition. However, we believe that it is unreasonable to expect a service like *GlOSS* to guess this type of $Right$ set of sites. Since the information kept by *GlOSS* about each database is necessarily much less detailed than that kept by the search engine at each database, it would be very hard for *GlOSS* to accurately guess the number of relevant documents in the answer to a query given by a database.

[3] A limited form of "or" queries is implicit whenever the "subject" index is used (see [GGMT94]).

[4] In fact, we are not retrieving all of the word frequencies, but only those that are needed for the queries in $TRACE_{INSPEC}$.

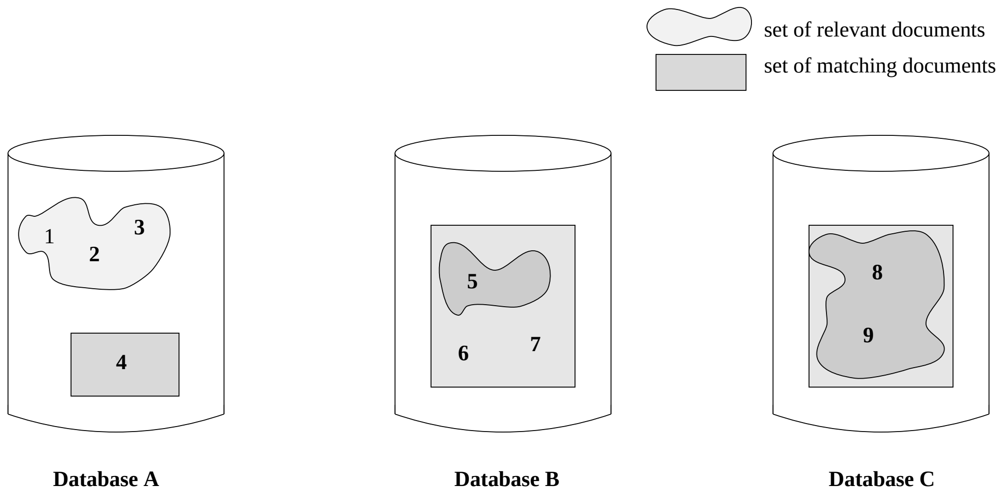

Figure 5: The documents relevant to a given query vs. the documents actually given as the answer to the query, for three different databases. Documents are represented by numbers in this figure.

- $Right = \{B\}$, since B is the database that produces the largest number of matching documents for q. Presumably, if the individual databases retrieve a reasonable approximation of the set of documents relevant to the given query, the *Right* database according to this definition would yield the highest number of useful documents. Also, the semantics of this definition are easily understood by the users, since they do not depend on relevance judgements, for example. □

In our first two definitions of the *Right* set, we will take the third approach illustrated in the example. That is, the goodness of a database db with respect to a query q will be determined by the number of documents that db returns when presented with q (i.e., the number of documents matching q in db). Our first definition for $Right(q, DB)$ is $Matching(q, DB)$, the set of all databases in DB containing at least one document that matches query q. More formally,

$$Right(q, DB) = Matching(q, DB) = \{db \in DB | \mathsf{RSize}(q, db) > 0\} \qquad (5)$$

There are (at least) two types of users that may specify $Matching(q, DB)$ as their right set of databases. One is users that want an exhaustive answer to their query. They are not willing to miss any of the matching documents. We will refer to these users as "recall-oriented" users. On the other hand, "precision-oriented" users may be in "sampling" mode: they simply want to obtain *some* matching documents without searching useless databases.

Our second definition for $Right(q, DB)$ is $Best(q, DB)$, the set of those databases that contain more matching documents than any other database. More formally,

$$\begin{aligned} Right(q, DB) &= Best(q, DB) \\ &= \{db \in DB | \mathsf{RSize}(q, db) > 0 \wedge \mathsf{RSize}(q, db) = \max_{db' \in DB} \mathsf{RSize}(q, db')\} \end{aligned} \qquad (6)$$

Again, users that define $Best(q, DB)$ as their right set of databases for query q might be classified as being "recall oriented" or "precision oriented." "Recall-oriented" users want all of the best databases for their query. These users are willing to miss some databases, as long as they are not the best ones. That is, the users recognize that there are more databases that could be examined, but want to ensure that at least those having the highest payoff (i.e., the largest number of documents) are searched. On the other hand, "precision-oriented" users want to examine (some)

Database set (DB)	{INSPEC, COMPENDEX, ABI, GEOREF, ERIC, PSYCINFO}
Estimator	*Ind*
Query set	$TRACE_{INSPEC}$
Query sizes considered	All
ϵ_C	0
ϵ_B	0

Figure 6: Basic configuration of the experiments.

best databases. Due to limited resources (e.g., time, money) the users only want to submit their query at databases that will yield the highest payoff.

Our third definition for $Right(q, DB)$, $Matching_I(q, DB)$, is specific for the case INSPEC $\in DB$, and for queries $q \in TRACE_{INSPEC}$. (This definition will be useful in the experiments we describe starting in Section 6.2.) In this case, we assume that INSPEC is the right database to search, regardless of the number of matching documents in the other databases, because the users issued the $TRACE_{INSPEC}$ queries to the INSPEC database, and perhaps they knew what the right database to search was. This is somewhat equivalent to regarding each query $q \in TRACE_{INSPEC}$ as augmented with the extra conjunct "$\wedge$ *database* INSPEC." So, our third definition for *Right* is:

$$
\begin{aligned}
Right(q, DB) &= Matching_I(q, DB) \\
&= \begin{cases} \{\text{INSPEC}\} & \text{if INSPEC} \in DB \ \wedge \ \mathsf{RSize}(q, \text{INSPEC}) > 0 \\ \emptyset & \text{otherwise} \end{cases}
\end{aligned} \tag{7}
$$

5.4 Configuration of the experiments

There are a number of parameters to our experiments. Figure 6 shows an assignment of values to these parameters that will determine the *basic configuration*. In later sections, some of these parameters will be changed, to produce alternative results. The parameters ϵ_C and ϵ_B will be defined in Section 7.1.

6 *Ind* results

In this section, we evaluate *Ind* by first studying how well it can predict the result size of a query and a database (Section 6.1). After this, we analyze *Ind*'s ability to distinguish between two databases (Section 6.2) and then we generalize the experiments to include six databases (Section 6.3). Finally, we repeat some of the experiments for a different set of queries to see how dependent our results are on the query trace used (Section 6.4).

6.1 *Ind* as a predictor of the result size of the queries

The key to *Ind* is its estimation function $\mathsf{ESize}_{Ind}(q, db)$, which predicts how many documents matching query q database db has. Before seeing how accurate *Ind* is at selecting a good subset of databases, let us study its estimation function ESize_{Ind}. An important question is whether ESize_{Ind} is a good predictor of the result size of a query in absolute terms, that is, whether the

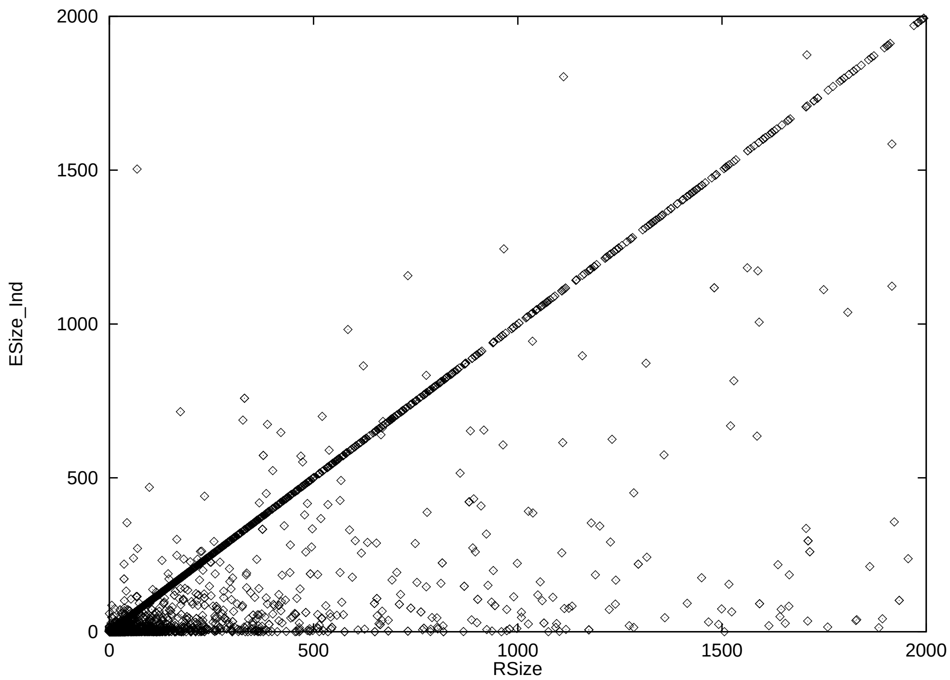

Figure 7: *Ind* as an estimator of the result size of the queries.

following holds:

$$\mathsf{ESize}_{Ind}(q, db) \quad \approx \quad \mathsf{RSize}(q, db)$$

If we analyze the data we collected, as explained in Section 5, the answer is no, unfortunately. In general, *Ind* tends to underestimate the result size of the queries. The more conjuncts in a query, the worse this problem becomes. Figure 7 shows a plot of the pairs:

$$< \mathsf{RSize}(q, \text{INSPEC}), \mathsf{ESize}_{Ind}(q, \text{INSPEC}) >$$

for the queries in $TRACE_{INSPEC}$ (see Section 5). The accumulation of points on the $y = x$ axis corresponds to the one-atomic-subquery queries (e.g., "*find author Knuth*"), for which $\mathsf{ESize}_{Ind} = \mathsf{RSize}$ (this follows from Equation 2).

Nevertheless, *Ind* will prove to be good at discriminating between useful and less useful databases according to the P and R parameters of Section 4. The reason for this is that even though $\mathsf{ESize}_{Ind}(q, db)$ will in general not be a good approximation of $\mathsf{RSize}(q, db)$, it is usually the case that $\mathsf{ESize}_{Ind}(q, db') < \mathsf{ESize}_{Ind}(q, db)$ if database db contains more documents matching query q than database db' does.

6.2 Evaluating *Ind* over pairs of databases

In this section, we report some results for the basic configuration (Figure 6), but with DB, the set of available databases, set to just two databases. Figures 8 and 9 show two matrices classifying the 6897 queries in $TRACE_{INSPEC}$ for the cases DB ={INSPEC, PSYCINFO} and DB ={INSPEC, COMPENDEX}. The sum of all of the entries of each matrix equals 6897. Consider for example Figure 8, for DB ={INSPEC, PSYCINFO}. Each row of the matrix represents an outcome for $Matching$ and $Best$. The first row, for instance, represents queries where both INSPEC and PSYCINFO had matching documents ($Matching$ ={INSPEC, PSYCINFO}) but where INSPEC had the most matching documents ($Best$ ={INSPEC}). On the other hand, each column represents the prediction made by *Ind*. For example, the number 2678 means that for 2678 of the queries in $TRACE_{INSPEC}$, $Best$ ={INSPEC}, $Matching$ ={INSPEC, PSYCINFO}, and *Ind* selected INSPEC as its prediction ($Chosen_{Ind}$ ={INSPEC}). In the same row, there were 26 other queries where *Ind* picked a matching database (PSYCINFO) but not the best one. In the first two rows, we see that for most of the queries (5614 out of 6897), INSPEC was the best database. This is not surprising, since the queries used in the experiments were originally issued by users to the INSPEC database.

The two matrices of Figures 8 and 9 show that $Chosen_{Ind} = \emptyset$ only if $Matching = \emptyset$. From Equations 1 and 2 it follows that this relationship holds in general, that is, as long as there is at least one database that contains matching documents, $Chosen_{Ind}$ will be non-empty. Also, note that very few times (15 for {INSPEC, PSYCINFO} and 92 for {INSPEC, COMPENDEX}) does *Ind* determine a tie between the two databases (and so, $Chosen_{Ind}$ consists of both databases). This is so since it is unlikely that $\mathsf{ESize}_{Ind}(q, db_1)$ will be exactly equal to $\mathsf{ESize}_{Ind}(q, db_2)$ if $db_1 \neq db_2$. With the current definition of $Chosen_{Ind}$, if for some query q and databases db_1 and db_2 it is the case that, say, $\mathsf{ESize}_{Ind}(q, db_1) = 9$ and $\mathsf{ESize}_{Ind}(q, db_2) = 8.9$, then $Chosen_{Ind}(q, \{db_1, db_2\}) = \{db_1\}$. We might want in such a case to include db_2 also in $Chosen_{Ind}$. We address this issue in Section 7.1, where we relax the definition of $Chosen_{Ind}$ and $Best$.

Figures 10 and 11 report the values of the P and R parameters for the three different target sets defined in Section 5.3. For example, in the second row of Figure 10, R^{Ind}_{Best}= 0.9910. This means that for the average query, $Chosen_{Ind}$ includes 99.10% of the $Best$ databases when DB ={INSPEC, PSYCINFO}. Therefore, for most of the $TRACE_{INSPEC}$ queries, $Best \subseteq Chosen_{Ind}$: from Figure 8, $Best \subseteq Chosen_{Ind}$ for 6831 queries. Also, for 6328 queries, $Chosen_{Ind}$ was exactly equal to $Best$. The reason for such high values is that INSPEC and PSYCINFO cover very different topics (see Figure 4). Therefore, for each query there is likely to be a clear “winner” (generally INSPEC for the queries in $TRACE_{INSPEC}$). On the other hand, INSPEC and COMPENDEX cover somewhat overlapping areas, thus yielding a lower (0.9216) value for R^{Ind}_{Best} (see Figure 11), for example.

The values for $R^{Ind}_{Matching}$ are lower in both the PSYCINFO and COMPENDEX cases: this is not surprising since *Ind* chooses the *most* promising databases, not all of the ones potentially containing matching documents. Therefore, some matching databases may be missed. Section 7.2 introduces a different estimator for *GlOSS*, *Bin*, aimed at optimizing the case $Right = Matching$. Notice that $R^{Ind}_{Matching}$ is particularly low (0.6022) for the pair {INSPEC, COMPENDEX}, since for most of the queries, there are matching documents in both databases (see the rows of Figure 9 corresponding to $Matching$ ={INSPEC, COMPENDEX}), and very rarely does *Ind* choose more than one database, as explained above.

From Figure 10, P^{Ind}_{Best}= 0.9187, showing that for each query, an average of 91.87% of the databases in $Chosen_{Ind}$ are among the $Best$ databases. So, for most of the queries, $Chosen_{Ind} \subseteq Best$: from Figure 8, $Chosen_{Ind} \subseteq Best$ for 6336 queries. In general, the values for P^{Ind}_{Best} and $P^{Ind}_{Matching}$ are relatively high for both pairs of databases, showing that in most cases $Chosen_{Ind}$ consists only of matching databases (high $P^{Ind}_{Matching}$) and in many of these cases, $Chosen_{Ind}$ consists only of “best” databases (high P^{Ind}_{Best}). Furthermore, it is always the case that $P^{Ind}_{Best}(q, DB) \leq P^{Ind}_{Matching}(q, DB)$, since $Best(q, DB) \subseteq Matching(q, DB)$.

Finally, note that the values of $P^{Ind}_{Matching_I}$ and $R^{Ind}_{Matching_I}$ are higher for the {INSPEC, PSYCINFO} pair than for the {INSPEC, COMPENDEX} pair: for the {INSPEC, PSYCINFO} pair, INSPEC is almost always clearly the best database (see Figure 8), whereas this is true to a lesser extent for the {INSPEC, COMPENDEX} pair (see Figure 9).

Best	*Matching*	$Chosen_{Ind}$			
		{I}	{P}	{I, P}	∅
{I}	{I, P}	2678	26	0	0
{I}	{I}	2894	16	0	0
{P}	{I, P}	11	224	0	0
{P}	{P}	5	34	0	0
{I, P}	{I, P}	3	5	15	0
∅	∅	462	41	0	483

Figure 8: Results corresponding to DB = {INSPEC (I), PSYCINFO (P)} and *Ind* as the estimator.

Best	*Matching*	$Chosen_{Ind}$			
		{I}	{C}	{I, C}	∅
{I}	{I, C}	4053	247	0	0
{I}	{I}	382	43	0	0
{C}	{I, C}	144	743	0	0
{C}	{C}	23	100	0	0
{I, C}	{I, C}	125	43	92	0
∅	∅	319	173	0	410

Figure 9: Results corresponding to DB = {INSPEC (I), COMPENDEX (C)} and *Ind* as the estimator.

Right	P^{Ind}_{Right}	R^{Ind}_{Right}
Matching	0.9240	0.7833
Best	0.9187	0.9910
$Matching_I$	0.8810	0.9607

Figure 10: Parameters P and R for DB ={INSPEC, PSYCINFO} and *Ind* as the estimator.

Right	P^{Ind}_{Right}	R^{Ind}_{Right}
Matching	0.9191	0.6022
Best	0.8624	0.9216
$Matching_I$	0.7482	0.8440

Figure 11: Parameters P and R for DB ={INSPEC, COMPENDEX} and *Ind* as the estimator.

$Right$	P^{Ind}_{Right}	R^{Ind}_{Right}
$Matching$	0.9126	0.4044
$Best$	0.8438	0.9010
$Matching_I$	0.5966	0.7012

Figure 12: Parameters P and R for the basic configuration of the experiments.

[GGMT93] reports experimental results for all the pairs of databases that can be obtained from {INSPEC, COMPENDEX, ABI, GEOREF, ERIC, PSYCINFO}. The two pairs of databases analyzed in this section, {INSPEC, PSYCINFO} and {INSPEC, COMPENDEX}, are among the best and the worst, respectively, for *Ind*, among all possible pairs: in general, the more unrelated the subject domains of the two databases considered were, the better *Ind* behaved in distinguishing the databases.

6.3 Evaluating *Ind* over six databases

In this section we report some results for the basic configuration, as defined in Figure 6. Figure 12 summarizes the results corresponding to the three definitions of the $Right$ set of Section 5.3. This figure shows that the same phenomena described in Section 6.2 prevail, although in general the values are lower. For example, $R^{Ind}_{Matching}$ is much lower (0.4044), since *Ind* chooses only the most promising databases, not all of the ones that might contain matching documents (see Section 7.2). Still, R^{Ind}_{Best} is high (0.9010), showing *Ind*'s ability to predict what the best databases are. Also, $P^{Ind}_{Matching}$ and P^{Ind}_{Best} are high (0.9126 and 0.8438, respectively), making *Ind* useful for exploring *some* of the matching/best databases. This is particularly significant for *Ind*: $Chosen_{Ind}(q, DB)$ will be non-empty as long as there is some database in DB that contains some document matching query q.

Another interesting piece of information that we gathered in our experiments is the fact that for only 96 out of the 6897 $TRACE_{INSPEC}$ queries does $Chosen_{Ind}$ consist of more than one database. Furthermore, 95 out of these 96 queries are one-atomic-subquery queries, for which $Chosen_{Ind} = Best$ necessarily (this follows from Equations 1 and 2). So, revisiting the results of Figure 12, since R^{Ind}_{Best}=0.9010, for most of the $TRACE_{INSPEC}$ queries not only does *Ind* narrow down the search space to one database (out of the six available ones), but it also manages to select the best database when there is one.

6.4 Impact of using other traces

So far, all of our experiments were based on the set of 6897 $TRACE_{INSPEC}$ queries. To analyze how dependent the results are on the trace used, we ran our experiments using a different set of queries. Real users issued these queries to the ERIC database between 3/28/1993 and 4/10/1993. We processed the trace in the same way as the INSPEC trace (see Section 5). The final set of queries, $TRACE_{ERIC}$, has 2404 queries, or 78.82% of the original 3050 query set.

Figure 13 shows the results for the different instances of the P and R parameters, for the basic configuration (Figure 6) but using $TRACE_{ERIC}$. The definition of the $Matching_E$ set of databases is analogous to that of $Matching_I$ (see Equation 7), using ERIC instead of INSPEC. The results obtained differ only slightly from the ones in Figure 12 for $TRACE_{INSPEC}$. This suggests that our results are not sensitive to the type of trace used.

$Right$	P^{Ind}_{Right}	R^{Ind}_{Right}
$Matching$	0.8960	0.4621
$Best$	0.8498	0.9384
$Matching_E$	0.5485	0.6876

Figure 13: Parameters P and R for the basic configuration, but using the $TRACE_{ERIC}$ queries.

7 Improving *GlOSS*

In this section we introduce variations to the definition of the $Chosen_{EST}$ and $Best$ sets in order to make them more flexible (Section 7.1), and present two new estimators, *Min* and *Bin* (Section 7.2).

7.1 Making $Chosen_{EST}$ and $Best$ more flexible

The definitions of $Chosen_{EST}$ and $Best$ given by Equations 1 and 6 are sometimes too "rigid." Consider the following example. Suppose $\{db_1, db_2\}$ is our set of databases, and let q be a query such that $\mathsf{RSize}(q, db_1) = 1,000$, and $\mathsf{RSize}(q, db_2) = 1,001$. According to Equation 6, $Best(q, DB) = \{db_2\}$. But this is probably too arbitrary, since both databases are almost identical regarding the number of matching documents they have for query q. Also, if an estimator EST predicts that the two databases contain a very similar number of documents satisfying a query, though not exactly equal, it might be preferable to choose both databases as the answer instead of picking the one with absolute highest estimated size.

In this section, we extend the definitions of $Chosen_{EST}$ and $Best$, through the introduction of two parameters, ϵ_B and ϵ_C. Parameter ϵ_B will make the definition of $Best$ looser, by letting databases with a number of documents close but not exactly equal to the maximum be considered as "best" databases also. Parameter ϵ_C changes the "matching" function (Section 3.3) of an estimator EST by making it able to choose databases that are close to the predicted optimal ones. The new definitions for $Chosen_{EST}$ and $Best$ are, for given $\epsilon_B, \epsilon_C \geq 0$:

$$Chosen_{EST}(q, DB) = \{db \in DB | \mathsf{ESize}_{EST}(q, db) > 0 \wedge \left| \frac{\mathsf{ESize}_{EST}(q, db) - hest}{hest} \right| \leq \epsilon_C\} \quad (8)$$

$$Best(q, DB) = \{db \in DB | \mathsf{RSize}(q, db) > 0 \wedge \left| \frac{\mathsf{RSize}(q, db) - hreal}{hreal} \right| \leq \epsilon_B\} \quad (9)$$

where

$$hest = \max_{db' \in DB} \mathsf{ESize}_{EST}(q, db') \text{ and } hreal = \max_{db' \in DB} \mathsf{RSize}(q, db').$$

Therefore, the larger ϵ_B and ϵ_C, the more databases will be included in $Best$ and $Chosen_{EST}$, respectively. Note that Equations 1 and 6 coincide with Equations 8 and 9 for $\epsilon_B = \epsilon_C = 0$. Also, if $\epsilon_C = 1$, *Ind* becomes the *Bin* estimator described in Section 7.2: $Chosen_{Ind}(q, DB)$ thus consists of all of the databases in DB that *might* contain some matching documents for query q.

Figures 14 and 15 show the average values of the P and R parameters, respectively, for the basic configuration of the experiments ($\epsilon_C = 0$), but for different values of ϵ_B. Thus, our *Ind* estimator remains fixed (since $\epsilon_C = 0$) and so do $Matching$ and $Matching_I$, since they do not depend on the parameter ϵ_B. This is why the curves corresponding to $P^{Ind}_{Matching}$, $R^{Ind}_{Matching}$, $P^{Ind}_{Matching_I}$, and $R^{Ind}_{Matching_I}$ are flat. On the other hand, the set of best databases, $Best$, varies as ϵ_B does. By varying ϵ_B alone, we are leaving the estimator fixed, and we change the semantics of our evaluation criteria, because we are modifying (i.e., making more flexible) our $Best$ "target" set.

In Figure 15 we see that parameter R^{Ind}_{Best} worsens as ϵ_B grows, since *Best* tends to contain more databases, while $Chosen_{Ind}$ remains fixed. This is exactly why P^{Ind}_{Best} (Figure 14) improves with higher values of ϵ_B. Note that for $\epsilon_B = 1$, $Best = Matching$, and so, $P^{Ind}_{Matching}$ and $R^{Ind}_{Matching}$ coincide with P^{Ind}_{Best} and R^{Ind}_{Best}, respectively.

As mentioned above, parameter ϵ_B is not a parameter of our estimator, but of the semantics of the queries. The submitter of a query does not give an ϵ_B value to *GlOSS*. Instead, higher values for ϵ_B yield more comprehensive *Best* sets. Therefore, parameter ϵ_B should be fixed according to the desired "meaning" for *Best*. For example, suppose that we are evaluating *Ind* for a user that wants to locate *Best* databases, but is willing to search at sites that have 90% or more of the number of matching documents than the overall *Best* sites have. Then, the experimental results that are relevant to this user are those obtained for $\epsilon_B = 0.1$.

Figures 16 and 17 show the average values of the P and R parameters, respectively, for the basic configuration of the experiments ($\epsilon_B = 0$), but for different values of ϵ_C. Here, the $Matching$ and $Matching_I$ sets do not change (they do not depend on ϵ_C), and neither does *Best* (since $\epsilon_B = 0$). *Ind* is affected, since ϵ_C is variable. Since $Chosen_{Ind}$ tends to cover more databases as ϵ_C grows, $R^{Ind}_{Matching}$, R^{Ind}_{Best}, and $R^{Ind}_{Matching_I}$ improve for higher values of ϵ_C. For $\epsilon_C = 1$, $R^{Ind}_{Matching} = R^{Ind}_{Best} = R^{Ind}_{Matching_I} = 1$, since $Chosen_{Ind}$ contains all of the potentially matching databases: as mentioned above, *Ind* becomes the *Bin* estimator (Section 7.2) for $\epsilon_C = 1$. This is also why P^{Ind}_{Best} and $P^{Ind}_{Matching_I}$ worsen as ϵ_C grows. Parameter $P^{Ind}_{Matching}$ remains basically unchanged for higher values of ϵ_C, but worsens for ϵ_C close to one, for the same reasons P^{Ind}_{Best} and $P^{Ind}_{Matching_I}$ get lower. Note that for $\epsilon_C = 1$, $P^{Ind}_{Best} \neq P^{Ind}_{Matching}$, since *Best* and *Matching* differ ($\epsilon_B = 0$).

From Figures 16 and 17 we can conclude that the value for ϵ_C should be set according to whether precision or recall should be emphasized (in the sense of Section 4). Users can set the value for ϵ_C to be used by *Ind* according to the query semantics they are interested in: in general, higher values for ϵ_C make the R parameters improve, while the P parameters worsen. However, when the *Right* set of databases is equal to the *Best* set, $\epsilon_C = 0$ is a good compromise to obtain both high P and high R values, since R^{Ind}_{Best} is already high for $\epsilon_C = 0$ (and so is P^{Ind}_{Best}).

7.2 Other estimators

So far, all of our experiments involved *Ind* as the estimator for *GlOSS*. In this section, we consider two other estimators, and compare their performance with that of *Ind*.

Ind is based upon the assumption that the occurrence of query keywords in documents follows independent and uniform probability distributions. We can build alternative estimators by departing from this assumption. For example, we can adopt the "opposite" assumption, and assume that the keywords that appear together in a user query are strongly correlated. So, we define another estimator for *GlOSS*, *Min* (for "minimum"), by letting:

$$\mathsf{ESize}_{Min}(find\ t_1 \wedge \ldots \wedge t_n, db) = \min_{i=1}^{n} freq(t_i, db) \qquad (10)$$

$\mathsf{ESize}_{Min}(q, db)$ is an upper bound of the actual result size of query q:

$$\mathsf{RSize}(q, db) \leq \mathsf{ESize}_{Min}(q, db)$$

$Chosen_{Min}$ follows from the definition of ESize_{Min}, using Equation 1.

If our goal is to maximize $R^{EST}_{Matching}$, then we should be very conservative in dropping databases from the $Chosen_{EST}$ set. With this motivation we define another estimator for *GlOSS*, *Bin* (for

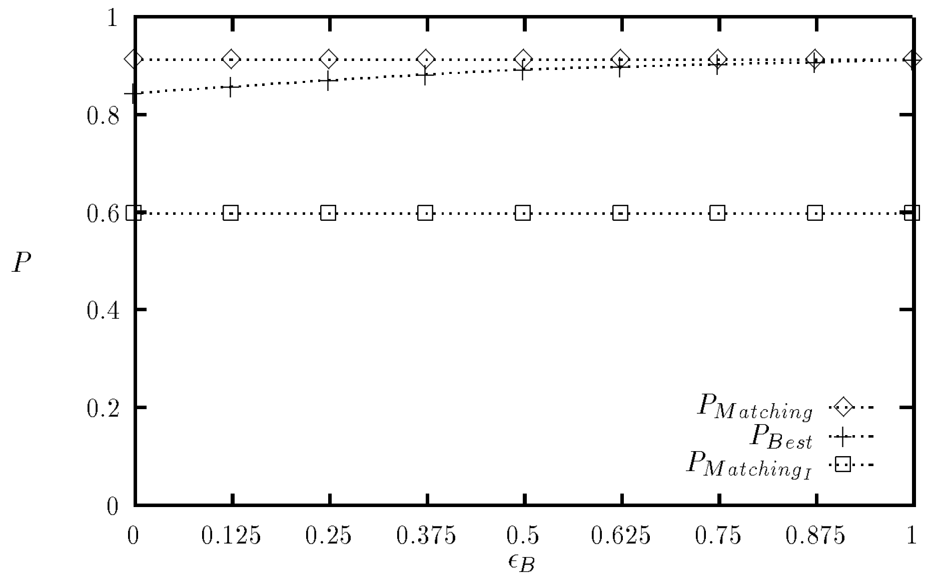

Figure 14: The average P parameters as a function of ϵ_B for the *Ind* estimator ($\epsilon_C = 0$).

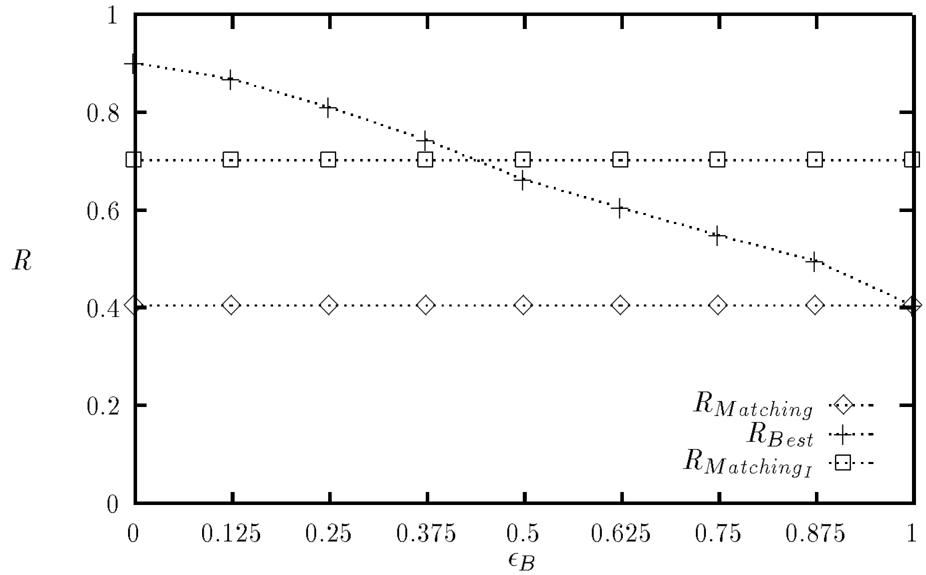

Figure 15: The average R parameters as a function of ϵ_B for the *Ind* estimator ($\epsilon_C = 0$).

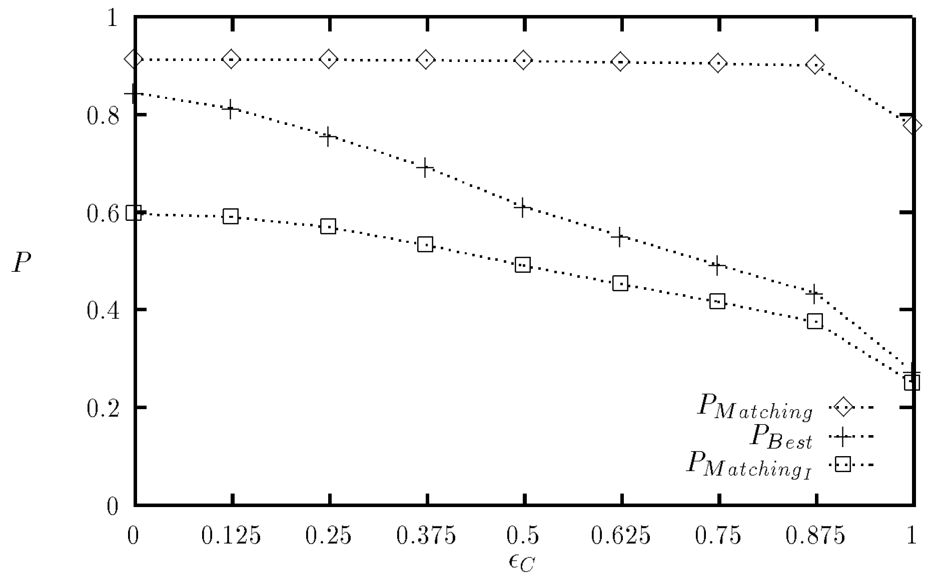

Figure 16: The average P parameters as a function of ϵ_C for the *Ind* estimator ($\epsilon_B = 0$).

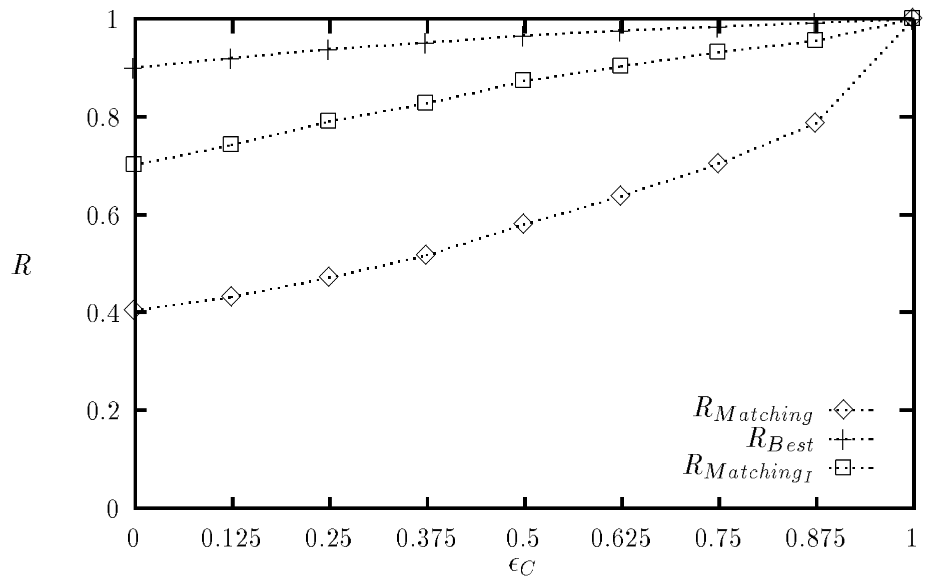

Figure 17: The average R parameters as a function of ϵ_C for the *Ind* estimator ($\epsilon_B = 0$).

$Right$	P^{Min}_{Right}	R^{Min}_{Right}	P^{Ind}_{Right}	R^{Ind}_{Right}
$Matching$	0.9077	0.4031	0.9126	0.4044
$Best$	0.8356	0.8938	0.8438	0.9010
$Matching_I$	0.6261	0.7316	0.5966	0.7012

Figure 18: The average P and R parameters for the basic configuration with *Min* as the estimator. The last two columns show the corresponding values for the basic configuration, using *Ind* as the estimator.

$Right$	P^{Bin}_{Right}	R^{Bin}_{Right}	P^{Ind}_{Right}	R^{Ind}_{Right}
$Matching$	0.7757	1	0.9126	0.4044
$Best$	0.2739	1	0.8438	0.9010
$Matching_I$	0.2494	1	0.5966	0.7012

Figure 19: The average P and R parameters for the basic configuration with *Bin* as the estimator. The last two columns show the corresponding values for the basic configuration, using *Ind* as the estimator.

"binary"):

$$\mathsf{ESize}_{Bin}(find\ t_1 \wedge \ldots \wedge t_n, db) \quad = \quad \begin{cases} 0 & \text{if } \exists i,\, 1 \leq i \leq n \mid freq(t_i, db) = 0 \\ 1 & \text{otherwise} \end{cases} \tag{11}$$

Again, $Chosen_{Bin}$ follows from the definition of ESize_{Bin}, using Equation 1. So, we are guaranteed that $R^{Bin}_{Matching} = R^{Bin}_{Best} = R^{Bin}_{Matching_I} = 1$ (at the expense of likely low values for the P parameters).

Figures 18 and 19 show the results obtained for the basic configuration (Figure 6) using the *Min* and *Bin* estimators, respectively. The results for *Min* are very similar to the corresponding results for *Ind*, with no significant differences. Note that the definition of $\mathsf{ESize}_{Min}(q, db)$ does not depend on the size of the *db* database, unlike the definition of $\mathsf{ESize}_{Ind}(q, db)$. This does not seem to have played an important role for the queries and databases we considered in the experiments, since the results we obtained for *Ind* and *Min* are very similar.

As expected, although *Bin* gets much higher values for the R parameters (in fact, $R^{Bin}_{Matching} = R^{Bin}_{Best} = R^{Bin}_{Matching_I} = 1$), it performs much worse for the P parameters than *Ind* and *Min*. For example, P^{Bin}_{Best} is very low: 0.2739. Note that $P^{Bin}_{Matching_I}$ is also low (0.2494), since *Bin* tends to produce overly conservative $Chosen_{Bin}$ sets, so as not to miss any of the databases with matching documents.

Consequently, a user might indicate what the query semantics are to *GlOSS*. *GlOSS* would then choose one of the estimators to answer the user query accordingly. Thus, if the user is interested in high values of the P parameters, then the *Ind* estimator would be used, whereas *Bin* would be the choice if high values of R are of interest. If, on the other hand, the user wants both high values of P and R, then *Ind* would be chosen for $Right = Best$, and *Bin* for $Right = Matching$.

8 Conclusions

In this paper we presented several estimators for *GlOSS*, a solution to the text-database discovery problem. We also developed a formal framework for this problem and defined different "right sets" of databases for evaluating a user's query. We used this framework to evaluate the effectiveness of the *GlOSS* estimators using real-user query traces. The experimental results we obtained, although involving only six databases, are encouraging. Furthermore, we believe that our results are independent of the query traces we used, since we obtained very similar results using two different query traces.

The storage cost of *GlOSS* is relatively low and was analyzed in [GGMT94]: a rough estimate suggested that 22.29 MBytes would be enough to keep all the data needed for the six databases we studied. In contrast, a full index of the six databases was estimated to require 1035.84 MBytes. Given its low space requirement, we can replicate *GlOSS* to increase its availability.

Our approach to solving the text-database discovery problem could also deal with information servers that would charge for their use. Since we are selecting what databases to search according to a quantitative measure of their "goodness" for a query (given by ESize_{EST}), we could easily incorporate this cost factor into the computation of ESize_{EST} so that, for example, given two equally promising databases, a higher value would be assigned to the least expensive of the two.

We are currently implementing a *GlOSS* server that will keep information on databases having WAIS [KM91] indexes. These databases can correspond to WAIS servers, or to World-Wide Web servers [BLCGP92] with WAIS indexes, for example. The *GlOSS* server will be available through World-Wide Web.

9 Acknowledgments

This research was sponsored by the Advanced Research Projects Agency (ARPA) of the Department of Defense under Grant No. MDA972-92-J-1029 with the Corporation for National Research Initiatives (CNRI). The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsement, either expressed or implied, of ARPA, the U.S. Government or CNRI. This work was supported by an equipment grant from Digital Equipment Corporation.
We would also like to thank Jim Davis and Daniela Rus for their helpful comments on an earlier version of this paper.

References

[BC92] Daniel Barbará and Chris Clifton. Information Brokers: Sharing knowledge in a heterogeneous distributed system. Technical Report MITL-TR-31-92, Matsushita Information Technology Laboratory, October 1992.

[BLCGP92] Tim Berners-Lee, Robert Cailliau, Jean-F. Groff, and Bernd Pollermann. World-Wide Web: The Information Universe. *Electronic Networking: Research, Applications and Policy*, 1(2), 1992.

[Cha88] Alice Y. Chamis. Selection of online databases using switching vocabularies. *Journal of the American Society for Information Science*, 39(3), 1988.

[DANO91] Peter B. Danzig, Jongsuk Ahn, John Noll, and Katia Obraczka. Distributed indexing: a scalable mechanism for distributed information retrieval. In *Proceedings of the* 14^{th} *Annual SIGIR Conference*, October 1991.

[DLO92] Peter B. Danzig, Shih-Hao Li, and Katia Obraczka. Distributed indexing of autonomous INTERNET services. *Computer Systems*, 5(4), 1992.

[DS93] Andrzej Duda and Mark A. Sheldon. Content routing in networks of WAIS servers, 1993. Unpublished material.

[ED92] Alan Emtage and Peter Deutsch. archie–an electronic directory service for the INTERNET. In *Proceedings of the Winter 1992 USENIX Conference*, January 1992.

[Fos92] Steve Foster. About the Veronica service, November 1992. Message posted in `comp.infosystems.gopher`.

[FW+93] Jim Fullton, Archie Warnock, et al. Release notes for freeWAIS 0.2, October 1993.

[FY93] David W. Flater and Yelena Yesha. An information retrieval system for network resources. In *Proceedings of the International Workshop on Next Generation Information Technologies and Systems*, June 1993.

[GGMT93] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. The efficacy of *GlOSS* for the text-database discovery problem. Technical Report STAN-CS-TN-93-002, Stanford University, November 1993. Available by anonymous `ftp` from `db.stanford.edu` in `/pub/gravano/1993/stan.cs.tn.93.002.ps`.

[GGMT94] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. The effectiveness of *GlOSS* for the text-database discovery problem. In *Proceedings of the 1994 ACM SIGMOD Conference*, May 1994. Also available by anonymous `ftp` from `db.stanford.edu` in `/pub/gravano/1994/stan.cs.tn.93.002.sigmod94.ps`.

[GS93] Ran Giladi and Peretz Shoval. Routing queries in a network of databases driven by a meta knowledge-base. In *Proceedings of the International Workshop on Next Generation Information Technologies and Systems*, June 1993.

[KM91] Brewster Kahle and Art Medlar. An information system for corporate users: Wide Area Information Servers. Technical Report TMC199, Thinking Machines Corporation, April 1991.

[MDT93] Anne Morris, Hilary Drenth, and Gwyneth Tseng. The development of an expert system for online company database selection. *Expert Systems*, 10(2):47–60, May 1993.

[MTD92] Anne Morris, Gwyneth Tseng, and Hilary Drenth. Expert systems for online business database selection. *Library Hi Tech*, 10(1-2):65–68, 1992.

[Neu92] B. Clifford Neuman. The Prospero File System: A global file system based on the Virtual System model. *Computer Systems*, 5(4), 1992.

[ODL93] Katia Obraczka, Peter B. Danzig, and Shih-Hao Li. INTERNET resource discovery services. *IEEE Computer*, September 1993.

[OM92] Joann J. Ordille and Barton P. Miller. Distributed active catalogs and meta-data caching in descriptive name services. Technical Report #1118, University of Wisconsin-Madison, November 1992.

[SA89] Patricia Simpson and Rafael Alonso. Querying a network of autonomous databases. Technical Report CS-TR-202-89, Dept. of Computer Science, Princeton University, January 1989.

[Sch90] Michael F. Schwartz. A scalable, non-hierarchical resource discovery mechanism based on probabilistic protocols. Technical Report CU-CS-474-90, Dept. of Computer Science, University of Colorado at Boulder, June 1990.

[Sch93] Michael F. Schwartz. INTERNET resource discovery at the University of Colorado. *IEEE Computer*, September 1993.

[SDW+94] Mark A. Sheldon, Andrzej Duda, Ron Weiss, James W. O'Toole, and David K. Gifford. A content routing system for distributed information servers, 1994. To appear in EDBT '94.

[SEKN92] Michael F. Schwartz, Alan Emtage, Brewster Kahle, and B. Clifford Neuman. A comparison of INTERNET resource discovery approaches. *Computer Systems*, 5(4), 1992.

[SFV83] G. Salton, E. A. Fox, and E. Voorhees. A comparison of two methods for boolean query relevance feedback. TR 83-564, Cornell University, July 1983.

[SM83] Gerard Salton and Michael J. McGill. *Introduction to modern information retrieval.* McGraw-Hill, 1983.

[WS93] Chris Weider and Simon Spero. Architecture of the WHOIS++ Index Service, October 1993. Working draft.

[ZC92] Sajjad Zahir and Chew Lik Chang. Online-Expert: An expert system for online database selection. *Journal of the American Society for Information Science*, 43(5):340–357, June 1992.