The Efficacy of *GlOSS* for the Text Database Discovery Problem

Luis Gravano[*] Héctor García-Molina[†] Anthony Tomasic[‡]

Abstract

The popularity of information retrieval has led users to a new problem: finding which text databases (out of thousands of candidate choices) are the most relevant to a user. Answering a given query with a list of relevant databases is the *text database discovery problem*. The first part of this paper presents a practical method for attacking this problem based on estimating the result size of a query and a database. The method is termed *GlOSS-Glossary of Servers Server*. The second part of this paper evaluates *GlOSS* using four different semantics to answer a user's queries. Real users' queries were used in the experiments. We also describe several variations of *GlOSS* and compare their efficacy. In addition, we analyze the storage cost of our approach to the problem.

1 Introduction

Information vendors such as Dialog and Mead Data Central provide content-indexed access to multiple databases. Dialog for instance has over three hundred databases. In addition, the advent of Archie, WAIS, World Wide Web, and other Internet tools has provided easy, distributed access to many more hundreds of text document databases. Thus, users are faced with the problem of finding the databases that are relevant to their information need (the user query). This paper presents a framework for (and analyzes a solution to) this problem, which we call the *text database discovery problem* (also referred to as the *resource discovery problem* in some generally more comprehensive contexts). [SEKN92] and [ODL93] provide surveys of proposed solutions to this problem.

The traditional information retrieval problem of finding documents relevant to a user query is studied by (a) describing an information retrieval model (consisting of a document representation, a query representation, and a matching algorithm) and (b) evaluating the model in terms of precision and recall [TC92, SB88]. This problem has a single underlying semantics of producing all (measured by recall) and only (measured by precision) the documents relevant to the query. We expand on this framework by motivating four different semantics for the database discovery problem.

Given a query and a set of databases, the user may be interested in (at least) four different *semantics* for the query:

- *Exhaustive Search.* The user is interested in the set of all databases which contain any *relevant* documents.
- *All-Best Search.* The user wants all the *best* databases for the query. The best databases are the ones which contain more relevant documents than any other database.

[*]Stanford University, Computer Science Dept., Margaret Jacks Hall, Stanford, CA 94305. E-mail: `gravano@cs.stanford.edu`

[†]Stanford University, Computer Science Dept., Margaret Jacks Hall, Stanford, CA 94305. 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. E-mail: `tomasic@cs.stanford.edu`

- *Only-Best Search.* The user is interested in any one best database (not necessarily in all of them).
- *Sample Search.* The user is interested in any one database which contains at least one relevant document (not necessarily in all such databases).

Intuitively, exhaustive search is the most computationally expensive and sample search is the least computationally expensive.

Example 1.1 *To illustrate the various semantics, consider the query* "find Knuth and computer" *and four databases A, B, C and D. Suppose that A and B have 10 documents relevant to the query, C has one relevant document, and D has zero relevant documents. Exhaustive Search would return* $\{A,B,C\}$ *as the answer, All-Best Search would return* $\{A,B\}$ *as the answer, Only-Best Search would return any subset of* $\{A, B\}$ *as the answer, and Sample Search would return any subset of* $\{A, B, C\}$ *as the answer. Some systems may directly present the answer to the user, other systems may then proceed with a second step and send the query to the database(s) and then present the relevant documents to the user. We leave this choice as a user interface implementation issue.*

One solution to the database discovery problem is to let the 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. The Veronica Service [Fos92] has recently added a search facility to Gopher. The Prospero File System [Neu92] lets users organize information available in the Internet through the definition (and sharing) of customized views (*virtual systems*) 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 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.

[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 information providers 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", that have, among other administrative data, a boolean query associated with each of them (a "generator rule"). Each broker indexes (not necessarily local) documents that satisfy the broker's 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] and [BC92] are other examples of this type of approach in which users query a "meta-information" database.

A "content based routing" system is used in [SDW$^+$] 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 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 Knuth* $\wedge$ *computer*, 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. This approach has been followed in [GS93].

Another approach to solving the database discovery problem is user transparent database selection in a way that guarantees exhaustive answers to the users' queries. For example, the Archie system [SEKN92] periodically obtains a recursive listing of the contents of all the available FTP sites in order to answer users' queries. One way to achieve complete answers would be to evaluate the query on all databases (if the databases are distributed, this implies broadcasting the query) and, given the resulting answer set sizes, returning the proper set of databases as an answer. However, the computational expense of broadcasting queries makes this approach infeasible in most situations[1]. Another alternative is to build a complete index of all the documents. By searching in this index, we could match exactly the query to all relevant documents. Unfortunately, this approach does not scale well either. For example, a full text index is usually the same size as the documents it references, so we would need to build a structure of the same size as all the documents in the network. A third strategy for exact matches is to construct some type of document or index hierarchy. For example, documents on a particular topic would be stored on a fixed database. If our query is for a *computer science* document, it would be routed to the Computer Science database. (This database could be physically distributed.) An alternative is to not partition the documents but the index. Thus, the *computer science* query would be routed to the Computer Science index machine. It could in turn provide a list of all the *computer science* documents that match, even though the documents could be on a variety of machines.

While some form of the hierarchical scheme may be attractive, it seems to require some type of network wide agreement on how to partition the documents or the indexes. Thus, as an alternative to these complete answer strategies, we wish to explore a methodology that yields, in general, not necessarily exhaustive answers to users' queries. Our approach is to have each database extract a "histogram" of its words occurrences. These histograms are used to estimate the result size of each query. For example, the Computer Science Library could report that the word *Knuth* occurs 180 times, the word *computer* 25,548 times, and so on. This information is orders of magnitude smaller than a full index (see Section 6.2 and Figure 25): For example, [CD90] reports statistics on a legal textual 9 Gigabyte database. The vocabulary of this database consists of approximately

[1]Dialog can evaluate a query over all databases and return the answer set sizes as the answer to the query.

A	B	C	D
Knuth 100	Knuth 10	Knuth 1	Knuth 10
computer 1000	computer 100	computer 100	computer 0
$d = 1000$	$d = 100$	$d = 200$	$d = 20$

Figure 1: Portion of the histograms for four databases. d is the database size in documents.

1,800,000 words. For each of these words we only need to keep its frequency, as opposed to the identities of the documents that have these words. Even though the histograms are small, they can still provide very useful information regarding what sites may have relevant documents. To illustrate estimation, consider the following example.

Example 1.2 *Figure 1 shows the portion of the histograms corresponding to four databases, A, B, C, and D, that is relevant to a query q=*find Knuth $\wedge$ computer. *Using this information, we can safely discard database D, since there will be no documents that are relevant to q in it: D does not contain any documents with the word "*computer*" in them. However, the remaining databases* may *contain relevant documents. So, for the Exhaustive Search semantics, we would return $\{A, B, C\}$ as the answer. The best databases are those whose estimates are maximal in the set of databases. Database A contains 1000 documents, 100 of which contain the word "*Knuth*". Therefore, the probability that a document in A contains the word "*Knuth*" is $\frac{100}{1000}$. Similarly, the probability that a document in A contains the word "*computer*" is $\frac{1000}{1000}$. Under the assumption that words appear independently in documents*[2]*, the probability that a document in database A has both the words "*Knuth*" and "*computer*" is $\frac{100}{1000} \times \frac{1000}{1000}$. Consequently, we can estimate the result size of query q in database A as $f(q, A) = \frac{100}{1000} \times \frac{1000}{1000} \times 1000 = 100$ documents. Similarly, $f(q, B) = \frac{10}{100} \times \frac{100}{100} \times 100 = 10$, $f(q, C) = \frac{1}{200} \times \frac{100}{200} \times 200 = 0.5$, and $f(q, D) = \frac{10}{20} \times \frac{0}{20} \times 20 = 0$ (as we explained above). Thus, A is the estimated best database and the All-Best semantic would return $\{A\}$. The Only-Best semantic would return $\{A\}$, and the Sample Semantic would return any database(s) with a non-zero estimate, so any subset of $\{A, B, C\}$ could be returned.*

Our approach is based on estimators of the result size of the queries. The contributions of this paper are:

- a formal framework for the text database discovery problem,
- the concept of a Glossary Of Servers Server (*GlOSS*) that diverts queries to appropriate information sources, based on previously collected frequency statistics about the sources,
- several estimators that may be used by *GlOSS* for making decisions, and
- an experimental evaluation of *GlOSS* according to different semantics for the queries, using real users' queries.

Section 2 introduces *GlOSS* and the concept of an *estimator*. In particular, Section 2.4 describes *Ind*, one estimator for *GlOSS* that we will evaluate in the rest of the paper. Estimators do not generally produce exact answers. For instance, following Example 1.2, it is possible that database A does not have any document at all containing *both* the word "*Knuth*" and the word "*computer*", even though we predicted that there would be 100 such documents. Therefore, given an estimation function, we would like to evaluate the efficacy of the estimation function with respect to the four semantics. Section 3 defines these evaluation criteria precisely. In order to assess the performance of

[2]We examine this assumption in Section 6.6.

GlOSS, we performed experiments using query traces from the FOLIO library information retrieval system at Stanford University. Section 4 describes these experiments, which involved six databases available through FOLIO. The experimental results are reported in Section 5. Section 6 examines the space requirements of *GlOSS*, and introduces variations to it, as well as a comparison of *Ind* with two other estimators for *GlOSS*, namely *Min* and *Binary*.

2 *GlOSS*: Glossary Of Servers Server

Consider a query q that we want to evaluate over a set of databases DB. *GlOSS* (**Gl**ossary **O**f **S**ervers **S**erver) selects a subset of DB consisting of "good candidate" databases for actually submitting q. To make this selection, *GlOSS* uses an *estimator*, which assesses how "good" each database in DB is with respect to the given query. In this paper, we study several such estimators for *GlOSS*.

2.1 Query representation

We will only consider *boolean "and"* queries, i.e. 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:

$$\textit{find author Knuth} \wedge \textit{subject computer}.$$

This query has two atomic subqueries:

- *author Knuth*, and
- *subject computer*

In *author Knuth*, *author* is the field designation, and *Knuth* the corresponding keyword.

The reason why we are considering only boolean queries so far is 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 4.1). Nevertheless, it should be stressed that the approach taken in this paper can be generalized to the vector space retrieval model [SM83]. 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 4.1) are "and" queries. However, as will be explained in Section 4, a limited form of "or" queries is implicit whenever the *subject* field designation is used (see Section 6.1).

2.2 Database histograms

GlOSS has available the following information:

- $DBSize(db)$, 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" 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 no information is found by the estimator about $freq(t, db)$, then $freq(t, db)$ will be assumed to be 0 (see Section 6.4).

2.3 Estimate of the result size of a query

Given $freq$ and $DBSize$ for a set of databases DB, *GlOSS* can use an estimator EST to compute a guess of the result size of a query q at any database $db \in DB$, $\mathsf{ESize}_{EST}(q, db)$. Ideally [3],

$$\mathsf{ESize}_{EST}(q, db) \approx \mathsf{RSize}(q, db) \tag{1}$$

where $\mathsf{RSize}(q, db)$ is the actual number of documents satisfying the query in the database. Once $\mathsf{ESize}_{EST}(q, db)$ has been defined, we can determine $Chosen_{EST}(q, DB)$ in the following way:

$$\begin{aligned} Chosen_{EST}(q, DB) &= \{db \in DB | \mathsf{ESize}_{EST}(q, db) = \max_{db' \in DB} \mathsf{ESize}_{EST}(q, db') \\ &\quad \wedge \mathsf{ESize}_{EST}(q, db) > 0\} \end{aligned} \tag{2}$$

So, ESize_{EST} completely determines an estimator EST. Equation 2 may seem targeted to identifying the databases containing the *most* relevant documents. However, Section 6.6 shows how $\mathsf{ESize}_{EST}(q, db)$ can be defined so that $Chosen_{EST}(q, db)$ becomes the set of *all* of the databases potentially containing relevant documents, when the *Binary* estimator is presented.

2.4 The *Ind* estimator

In this section, the estimator that we will use for most of our experiments, *Ind*, is described. *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, the estimated number of documents in db that will satisfy the query:

$$find\ t_1 \wedge \ldots \wedge t_n$$

is given, according to *Ind*, by:

$$\begin{aligned} \mathsf{ESize}_{Ind}(find\ t_1 \wedge \ldots \wedge t_n, db) &= \frac{\prod_{i=1}^{n} freq(t_i, db)}{DBSize(db)^n} \times DBSize(db) \\ &= \frac{\prod_{i=1}^{n} freq(t_i, db)}{DBSize(db)^{n-1}} \end{aligned} \tag{3}$$

The $Chosen_{Ind}$ set is then computed with Equation 2.

To illustrate this definition, let db =INSPEC (INSPEC is a database that will be used in our experiments, see Section 4). Also, let:

$$q = \textit{find author Knuth} \wedge \textit{subject computer}$$

The statistics available to *Ind* are:

[3]In fact, the *Binary* estimator presented in Section 6.6 is such that $\mathsf{ESize}_{Binary}(q, db)$ will not attempt to approximate $\mathsf{RSize}(q, db)$. Instead, $\mathsf{ESize}_{Binary}(q, db)$ will be one or zero, depending on whether db might contain documents relevant to query q or not.

- $DBSize(\text{INSPEC}) = 1,416,823$,
- $freq(\textit{author Knuth}, \text{INSPEC}) = 47$, and
- $freq(\textit{subject computer}, \text{INSPEC}) = 155,574$.

From this, *Ind* computes:

$$\begin{aligned} \mathsf{ESize}_{Ind}(q, \text{INSPEC}) &= \frac{47 \times 155,574}{1,416,823} \\ &\simeq 5.16 \end{aligned}$$

Incidentally, the actual result size of the query q in INSPEC (obtained by submitting the query) is:

$$\mathsf{RSize}(q, \text{INSPEC}) = 14$$

An interesting property of *Ind* is that if $freq(t_i, db) = 0$ for some $1 \leq i \leq n$, then

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

and so, *db* will not be included in $Chosen_{Ind}$. This is an important point, since in this case *db* cannot contain any document satisfying the given query, because of the boolean logic semantics of the query representation. So, *db* can be safely eliminated from the set of databases where to direct the query. To continue with our example, suppose that no database in a set DB other than INSPEC contains documents with *author Knuth*, i.e. $freq(\textit{author Knuth}, db) = 0$ for all $db \in DB - \{\text{INSPEC}\}$. Then, for each such database *db*, $\mathsf{ESize}_{Ind}(q, db) = 0$. Therefore,

$$\mathsf{ESize}_{Ind}(q, \text{INSPEC}) = \max_{db \in DB} \mathsf{ESize}_{Ind}(q, db)$$

and $Chosen_{Ind}(q, DB) = \{\text{INSPEC}\}$.

3 Evaluation Criteria

Let DB be a set of databases. In order to evaluate an estimator EST, we need to compare its prediction against what actually is the right subset of DB to query. There are two notions of what the "right subset" means: One is $Relevant(q, DB)$[4], the databases that contain documents that are judged to be "relevant" to q, in some predefined sense. A second notion of "right subset" is $Best(q, DB)$, those databases that yield the most relevant documents.

Once these sets have been defined for a query q and a database set DB, we can state four different criteria to evaluate $Chosen_{EST}(q, DB)$:

- *Exhaustive Search:* We are interested in obtaining *complete* answers to query q. We say that $Chosen_{EST}$ satisfies criterion C_{EX} for query q and set of databases DB if:

  $$C_{EX} : Relevant \subseteq Chosen_{EST}$$[5]

  In other words, we want to be certain to search all of the relevant databases.

[4]In general, we will drop the parameters of the functions when this will not lead to confusion. For example, we refer to $Relevant(q, DB)$ as *Relevant*, whenever q and DB are clear from the context.

[5]Note that $Relevant \subseteq Chosen_{EST}$ is a shorthand for $Relevant(q, DB) \subseteq Chosen_{EST}(q, DB)$.

- *All-Best Search:* We are interested in the *Best* databases for q. We say that $Chosen_{EST}$ satisfies criterion C_{AB} if:

$$C_{AB} : Best \subseteq Chosen_{EST}$$

  If $Best \subset Relevant$, we may not get an exhaustive answer to q (and so, $Chosen_{EST}$ may not satisfy C_{EX}). However, databases containing only a few documents will have been eliminated, permitting a more focused search.

- *Only-Best Search:* We are less demanding than with C_{AB}: we are just interested in searching (some of) the best databases for q. We might be missing some of these best databases, but we do not want to waste any time and resources by searching a non-optimal database. So, we say that $Chosen_{EST}$ satisfies criterion C_{OB} if:

$$C_{OB} : Chosen_{EST} \subseteq Best$$

- *Sample Search:* This criterion is the weakest: we just want an answer to our query from a relevant database. We do not want to carry fruitless searches in databases that do not contain any relevant documents. So, we say that $Chosen_{EST}$ satisfies criterion C_{SM} if:

$$C_{SM} : Chosen_{EST} \subseteq Relevant$$

Note that these four criteria correspond to the four different semantics for the database discovery problem, as described in Section 1. The set $Chosen_{EST}$ will be said to satisfy criterion C_{EX} *strictly* if $Chosen_{EST} = Relevant$. Analogous definitions follow for the other criteria.

Now, let C be any of the criteria above and Q be a fixed set of queries. Then,

$$Success(C, EST) = 100 \times \frac{|\{q \in Q | Chosen_{EST}(q, DB) \text{ satisfies } C\}|}{|Q|} \tag{4}$$

In other words, $Success(C, EST)$ is the percentage of Q queries for which EST produced the "right answer" under criterion C.

Following notions analogous to those used in Statistics, we define the *Alpha* and the *Beta* errors of EST for an evaluation criterion C as follows:

$$\begin{aligned} Alpha(C, EST) &= 100 - Success(C, EST) && (5) \\ Beta(C, EST) &= Success(C, EST) - && (6) \\ & \quad 100 \times \frac{|\{q \in Q | Chosen_{EST}(q, DB) \text{ satisfies } C \text{ strictly } \}|}{|Q|} \end{aligned}$$

So, $Alpha(C, EST)$ is the percentage of queries in Q for which the estimator gives the "wrong answer", i.e., the $Chosen_{EST}$ set does not satisfy criterion C at all. $Beta(C, EST)$ measures the percentage of queries for which the estimator satisfies the criterion, but not strictly. For the *Beta* queries, the estimator yields a correct but "overly conservative" (for C_{EX} and C_{AB}) or "overly narrow" (for C_{OB} and C_{SM}) answer. For example, consider an estimator, $TRIV$, that would always produce $\emptyset$ as the value for $Chosen_{TRIV}$. $TRIV$ would have $Success(C_{SM}, TRIV) = 100$ (and $Alpha(C_{SM}, TRIV) = 0$). However, *Beta* has a high value for conservative estimators: $Beta(C_{SM}, TRIV)$ would be quite high.

	C_{EX}	C_{SM}
$Success$	$R_{Relevant} = 1$	$P_{Relevant} = 1$
$Alpha$	$R_{Relevant} < 1$	$P_{Relevant} < 1$
$Beta$	$R_{Relevant} = 1$ and $P_{Relevant} < 1$	$P_{Relevant} = 1$ and $R_{Relevant} < 1$
$Success - Beta$	$R_{Relevant} = P_{Relevant} = 1$	$P_{Relevant} = R_{Relevant} = 1$

Figure 2: Summary of the relationship between the $Success$, $Alpha$, and $Beta$ functions and $P_{Relevant}$ and $R_{Relevant}$, for criteria C_{EX} and C_{SM}.

Now we will relate $Success$, $Alpha$, and $Beta$ to the well-known *precision* and *recall* parameters [SB88]. Consider, for example, criterion $C_{EX} : Relevant \subseteq Chosen_{EST}$. If we regard $Relevant$ 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 $P_{Relevant}$ and $R_{Relevant}$, based upon the *precision* and *recall* parameters[6]:

$$P_{Relevant}(q, DB) = \frac{|Chosen_{EST}(q, DB) \cap Relevant(q, DB)|}{|Chosen_{EST}(q, DB)|} \quad (7)$$

$$R_{Relevant}(q, DB) = \frac{|Chosen_{EST}(q, DB) \cap Relevant(q, DB)|}{|Relevant(q, DB)|} \quad (8)$$

Having $R_{Relevant}(q, DB) = 1$ is equivalent to having $Relevant(q, DB) \subseteq Chosen_{EST}(q, DB)$. Therefore, EST satisfies criterion C_{EX} for query q and set of databases DB if and only if $R_{Relevant}(q, DB) = 1$. This is shown in Figure 2, in the "$Success$" row, under C_{EX}. That is, $Success(C_{EX}, EST)$ gives the fraction of the queries for which the condition shown ($R_{Relevant} = 1$) is true.

Figure 2 gives the rest of the relationships. In particular, assume now that $R_{Relevant}(q, DB) = 1$ for some query q. Then, it is also the case that $P_{Relevant}(q, DB) < 1$ if and only if $Relevant(q, DB) \subset Chosen_{EST}(q, DB)$. Therefore, given that q satisfies criterion C_{EX} (or equivalently, $R_{Relevant}(q, DB) = 1$), q will add to $Beta(q, DB)$ if and only if $P_{Relevant}(q, DB) < 1$.

Now, consider criterion $C_{SM} : Chosen_{EST} \subseteq Relevant$. It follows from the definition of $P_{Relevant}$ that a query q will "contribute" to $Success(C_{SM}, EST)$ if and only if $P_{Relevant}(q, DB) = 1$. The conditions on $P_{Relevant}$ and $R_{Relevant}$ for $Beta$ are analogous to those for C_{EX} with the roles of $P_{Relevant}$ and $R_{Relevant}$ interchanged.

Analogously, Figure 3 summarizes the conditions for criteria C_{AB} and C_{OB}. These criteria involve the $Best$ set as opposed to the $Relevant$ set in criteria C_{EX} and C_{SM}. Therefore, the P_{Best} and R_{Best} functions are needed:

$$P_{Best}(q, DB) = \frac{|Chosen_{EST}(q, DB) \cap Best(q, DB)|}{|Chosen_{EST}(q, DB)|} \quad (9)$$

$$R_{Best}(q, DB) = \frac{|Chosen_{EST}(q, DB) \cap Best(q, DB)|}{|Best(q, DB)|} \quad (10)$$

As a final comment, notice that criteria C_{EX} and C_{AB} can be regarded as emphasizing recall over precision: these two criteria are satisfied whenever their "target sets" ($Relevant$ and $Best$, respectively) are included in $Chosen_{EST}$. On the other hand, C_{OB} and C_{SM} can be thought of as emphasizing precision over recall: even when $Chosen_{EST}$ is not a "complete" answer, success is achieved if no "useless" databases are included in $Chosen_{EST}$. Although $Success$, $Alpha$, and $Beta$ can be defined in terms of P and R (Figures 2 and 3), these definitions depend on whether

[6]Provided $Chosen_{EST}(q, DB)$ and $Relevant(q, DB)$ are non empty.

	C_{AB}	C_{OB}
Success	$R_{Best} = 1$	$P_{Best} = 1$
Alpha	$R_{Best} < 1$	$P_{Best} < 1$
Beta	$R_{Best} = 1$ and $P_{Best} < 1$	$P_{Best} = 1$ and $R_{Best} < 1$
Success − *Beta*	$R_{Best} = P_{Best} = 1$	$P_{Best} = R_{Best} = 1$

Figure 3: Summary of the relationship between the *Success*, *Alpha*, and *Beta* functions and P_{Best} and R_{Best}, for criteria C_{AB} and C_{OB}.

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.

the criteria emphasize recall or precision. Therefore, we will report the experimental results in terms of *Success*, *Alpha*, and *Beta* for the sake of clarity.

4 Experimental framework

In order to evaluate the performance of *Ind* (see Section 2.4) [7] according to the Section 3 criteria, we performed experiments using query traces from the FOLIO library information retrieval system at Stanford University.

4.1 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 chosen 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. Out of this set, 34 queries were eliminated, since they contained the character "*" apparently meant as a wildcard, while FOLIO generally does not consider it as such[8]. The "real" wildcard queries were kept. A total of 48 other queries were eliminated since they appeared truncated in the trace.

As discussed in Section 2.1, we only considered "and" queries [9]. Thus, we deleted from the trace those queries involving logical "or"s and "not"s. Also, we did not consider the so called

[7]We will refer indistinctively to both the estimator EST and its corresponding $Chosen_{EST}(q, DB)$ set as satisfying the criteria of Section 3, for a query q and a set of databases DB.

[8]The actual wildcard character is "#".

[9]In fact, a limited form of "or" queries is implicit whenever the "subject" index is used (see Section 6.1).

Source of the trace	INSPEC database
Raw trace queries	8392
Experiment queries	6897
Percentage experiment queries of raw queries	82.19

Figure 5: Characteristics of the set $TRACE_{INSPEC}$ of queries used in the experiments.

"phrase" queries (e.g. "*find titlephrase 'knowledge bases'*") and those queries that are extensions of previous queries[10].

The final set of queries, $TRACE_{INSPEC}$, has 6897 queries, out of the original 8392, or 82.19 % of the original set. Figure 5 summarizes some of this information.

4.2 Database histogram 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 *Best* and *Relevant* sets for each of the queries.

Also, to build the database histograms needed by *GlOSS* (Section 2.2) 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 2. This is exactly the information an estimator EST needs to define $Chosen_{EST}$, for each query in $TRACE_{INSPEC}$[11]. 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 regularly report the length of each inverted list to the estimator, so that the corresponding histogram is built.

4.3 Definition of the experiments

Section 3 introduced the notion of the *Relevant* and the *Best* sets of databases. Different instantiations of the four criteria of Section 3 are obtained for different definitions of *Relevant* and *Best*. In what follows, we give two definitions for each of these two sets.

- Ideally, a "relevant" database is one where the user that submitted the query would find documents of interest. Unfortunately, we have no way to know this. One way to attack this problem is to consider as relevant any database with documents satisfying the user's query. However, this does not necessarily solve the problem: For example, a database might contain a document written by a psychologist named *Knuth* on how *computers* can alienate people. This document may not be relevant to the issuer of the query *find Knuth* $\wedge$ *computers*. However, if we have no additional information on what relevant is, it is fair to simply look at databases with matching documents. Therefore, in our first definition, a database in DB will belong to $Relevant(q, DB)$ if and only if it has at least one document satisfying the query q. $Best(q, DB)$ is the set of those relevant databases that have the highest number of documents satisfying query q. More formally,

$$Relevant(q, DB) \quad = \quad \{db \in DB | \mathsf{RSize}(q, db) > 0\} \tag{11}$$

[10]In FOLIO, one can issue a query, say, *find author Knuth*, and after the query has been executed, one can extend it by entering *and subject computer*, for example.

[11]In fact, we are not building the complete histograms, but only those parts that are needed for the queries in $TRACE_{INSPEC}$.

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

Figure 6: Basic configuration of the experiments.

$$\begin{aligned} Best(q, DB) &= \{db \in DB | \mathsf{RSize}(q, db) > 0 \wedge \\ &\qquad \mathsf{RSize}(q, db) = \max_{db' \in DB} \mathsf{RSize}(q, db')\} \end{aligned} \tag{12}$$

We will refer to the instances of the four evaluation criteria (see Section 3) corresponding to these definitions as C_{EX}, C_{AB}, C_{OB}, and C_{SM}.

- The second definition 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 5.2). In this case, we assume that INSPEC is the "right" database, regardless of the number of matching documents in the other databases. This is so because the queries in $TRACE_{INSPEC}$ were issued by the users 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, we can define $Relevant_{INSPEC}$ and $Best_{INSPEC}$[12] as:

$$\begin{aligned} &Relevant_{INSPEC}(q, DB) = \\ &= \begin{cases} \{\text{INSPEC}\} & \text{if INSPEC} \in DB \wedge \\ & \mathsf{RSize}(q, \text{INSPEC}) > 0 \\ \emptyset & \text{otherwise} \end{cases} \\ &= Best_{INSPEC}(q, DB) \end{aligned} \tag{13}$$

Since $Relevant_{INSPEC} = Best_{INSPEC}$ in this case, the corresponding instances of the four criteria of Section 3 collapse into two criteria, which will be referred to as $C^{INSPEC}_{EX/AB}$ and $C^{INSPEC}_{OB/SM}$.

4.4 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 *threshold*, ϵ_C, and ϵ_B will be defined in Sections 6.4 and 6.5.

[12]We use the subindex $INSPEC$ to differentiate this second definition for *Best* and *Relevant* from the first one given above.

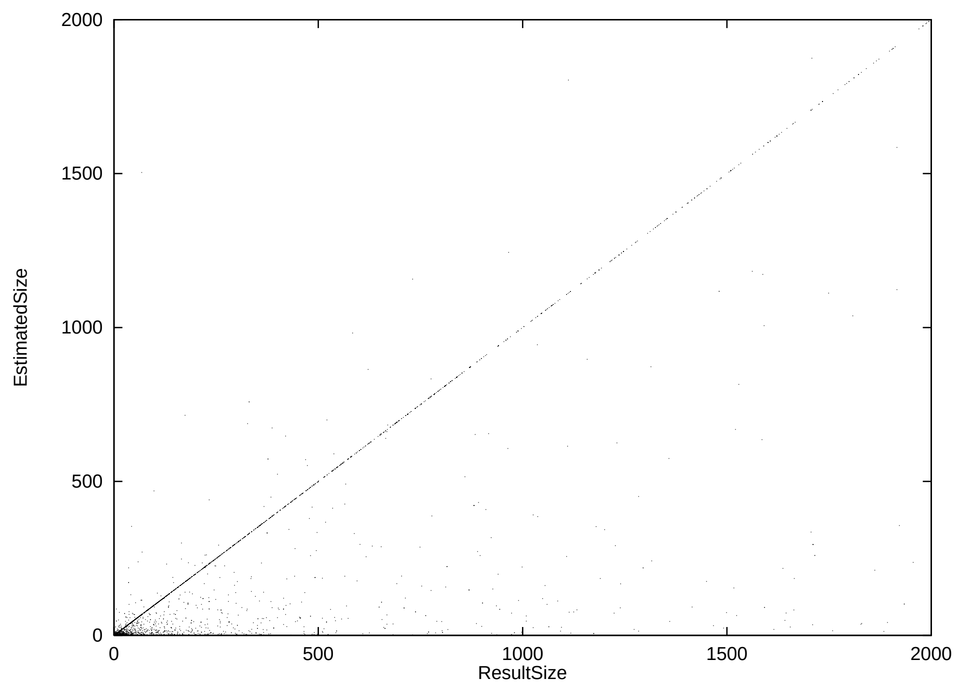

Figure 7: *Ind* as an estimator of the result size of the queries.

5 *Ind* results

5.1 *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, i.e., whether the following holds:

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

If we analyze the data we collected, as explained in Section 4, 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:

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

for the queries in $TRACE_{INSPEC}$ (see Section 4). The accumulation of points on the $y = x$ axis corresponds to the one term queries (e.g. *find author Knuth*), for which $\mathsf{ESize}_{Ind} = \mathsf{RSize}$.

Nevertheless, *Ind* will prove to be good at discriminating between useful and less useful databases for some of the criteria of Section 3. 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 that are relevant to query q than database db' does.

5.2 Evaluating *Ind* over pairs of databases

In this section, we report some results for the basic configuration (see 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 *Relevant* and *Best*. The first row, for instance, represents queries where both INSPEC and PSYCINFO had matching documents (*Relevant* ={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}, *Relevant* ={INSPEC, PSYCINFO}, and *Ind* correctly selected INSPEC as its prediction ($Chosen_{Ind}$ ={INSPEC}). In the same row, there were 26 other queries where *Ind* picked a relevant database (PSYCINFO) but not the best one.

The two matrices of Figures 8 and 9 show that $Chosen_{Ind} = \emptyset$ only if $Relevant = \emptyset$, i.e. as long as there is at least one database that might contain relevant documents, $Chosen_{Ind}$ will be non-empty. (This property will not hold with the modification to *GlOSS* of Section 6.4.) 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 very 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 6.5, where we relax the definition of $Chosen_{Ind}$ and *Best*.

Figures 10 and 11 report the values of *Success*, *Alpha*, and *Beta* for the six different criteria of Section 4.3. Consider for example the second row of the matrix in Figure 10. This row corresponds to criterion C_{AB} (see Section 4.3). *Ind* satisfies criterion C_{AB} for a query q if $Best(q, DB) \subseteq Chosen_{Ind}(q, DB)$, i.e. if $Chosen_{Ind}(q, DB)$ contains all of the best databases for q. From the table, we see that $Success(C_{AB}, Ind) = 99.04\%$. This means that in 99.04% of the cases *Ind* gave the correct answer (according to C_{AB}), that is, the $Chosen_{Ind}$ set of databases included the *Best* set of databases we were after. In 0.96% of the queries ($Alpha(C_{AB}, Ind)$) we got the "wrong" answer, i.e., the $Chosen_{Ind}$ set did not contain some of the best databases. Out of the successful cases (99.04%), sometimes *Ind* gives exactly the set of best databases, while in other cases it gives a larger set. The value $Beta(C_{AB}, Ind) = 7.29\%$ tells us how many queries were in the latter case. Finally, $Success(C_{AB}, Ind) - Beta(C_{AB}, Ind) = 91.75\%$ gives us the number of queries in the former case.

As we have just seen, for the case DB ={INSPEC, PSYCINFO}, $Success(C_{AB}, Ind) = 99.04\%$. The reason for such a high value 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 (90.94%) value for $Success(C_{AB}, Ind)$.

The values for $Success(C_{EX}, Ind)$ are much 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 relevant documents. Therefore, some relevant databases may be missed. Section 6.6 introduces a different estimator for *GlOSS*, *Binary*, aimed at maximizing *Success* for criterion C_{EX}. Notice that $Success(C_{EX}, Ind)$ is particularly low (21.40%) for the pair {INSPEC, COMPENDEX}, since for most of the queries, there are relevant documents in both databases (see the rows of Figure 9 corresponding to *Relevant* ={INSPEC, COMPENDEX}).

Note that it is always the case that $Success(C_{EX}, Ind) \leq Success(C_{AB}, Ind)$. The reason for this is that since $Best \subseteq Relevant$, if $Relevant \subseteq Chosen_{Ind}$ (criterion C_{EX}) then $Best \subseteq Chosen_{Ind}$ (criterion C_{AB}).

The values for $Success(C_{OB}, Ind)$ and $Success(C_{SM}, Ind)$ are relatively high for both pairs of

		$Chosen_{Ind}$			
Best	*Relevant*	{I}	{P}	{I, P}	$\emptyset$
{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
$\emptyset$	$\emptyset$	462	41	0	483

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

		$Chosen_{Ind}$			
Best	*Relevant*	{I}	{C}	{I, C}	$\emptyset$
{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
$\emptyset$	$\emptyset$	319	173	0	410

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

databases, showing that in most cases $Chosen_{Ind}$ consists only of relevant databases (criterion C_{SM}), and in many of these cases, $Chosen_{Ind}$ consists only of "best" databases (criterion C_{OB}). Furthermore, it is always the case that $Success(C_{OB}, Ind) \leq Success(C_{SM}, Ind)$. To see why note that $Chosen_{Ind} \subseteq Best$ (criterion C_{OB}) implies $Chosen_{Ind} \subseteq Relevant$ (criterion C_{SM}), since $Best \subseteq Relevant$.

Recall that $Beta(C_{OB/SM}^{INSPEC}, Ind)$ computes how many times the criterion $Chosen_{Ind} \subseteq Relevant_{INSPEC}$ is not met strictly. Since $Relevant_{INSPEC}$ can only be $\emptyset$ or {INSPEC}, then the criterion is not met strictly only if $Chosen_{Ind} = \emptyset$ and $Relevant_{INSPEC}$ = {INSPEC}. However, this can never happen because if $Chosen_{Ind} = \emptyset$, then $Relevant = \emptyset$, and hence $Relevant_{INSPEC}$ must also be empty (see Section 4.3).

Also, notice that for both criteria C_{EX} and C_{SM} the $Success - Beta$ entries are identical. This is because they have the same "target" set, namely the *Relevant* set of databases. That is, for both criteria, $Success - Beta$ measures the fraction of queries for which $Chosen_{Ind} = Relevant$. Therefore,

$$\begin{aligned} &Success(C_{EX}, Ind) - Beta(C_{EX}, Ind) = \\ &= \quad Success(C_{SM}, Ind) - Beta(C_{SM}, Ind) \end{aligned}$$

The same is true for C_{AB} and C_{OB}, with target set *Best*, and for $C_{EX/AB}^{INSPEC}$ and $C_{OB/SM}^{INSPEC}$, with target set $Relevant_{INSPEC}$ ($= Best_{INSPEC}$).

Finally, Figure 12 summarizes the evaluation criteria results for each of the 15 different pairs of databases that can be obtained from {INSPEC, COMPENDEX, ABI, GEOREF, ERIC, PSYCINFO}. Note that the two pairs of databases that we analyzed in more depth above, {INSPEC, PSYCINFO} and {INSPEC, COMPENDEX}, are among the "best" and the "worst", respectively, for *Ind*, among all possible pairs. In general, we can observe that the performance of *Ind* varies for the

Criteria	$Success$	$Alpha$	$Beta$	$Success - Beta$
C_{EX}	56.97	43.03	7.29	49.67
C_{AB}	99.04	0.96	7.29	91.75
C_{OB}	91.87	8.13	0.12	91.75
C_{SM}	92.40	7.60	42.73	49.67
$C_{EX/AB}^{INSPEC}$	96.07	3.93	8.08	87.99
$C_{OB/SM}^{INSPEC}$	87.99	12.01	0	87.99

Figure 10: Evaluation criteria for DB ={INSPEC, PSYCINFO} and *Ind* as the estimator.

Criteria	$Success$	$Alpha$	$Beta$	$Success - Beta$
C_{EX}	21.40	78.60	7.13	14.27
C_{AB}	90.94	9.06	7.13	83.80
C_{OB}	86.24	13.76	2.44	83.80
C_{SM}	91.91	8.09	77.64	14.27
$C_{EX/AB}^{INSPEC}$	84.40	15.60	10.25	74.15
$C_{OB/SM}^{INSPEC}$	74.15	25.85	0	74.15

Figure 11: Evaluation criteria for DB ={INSPEC, COMPENDEX} and *Ind* as the estimator.

different pairs of databases. $Success(C_{AB}, Ind)$ is always relatively high (above 90%) regardless of the pair. In contrast, the pair {INSPEC, COMPENDEX} reaches the lowest mark for $Success(C_{EX}, Ind)$, 21.40%. The reason for this, as was pointed out above, is that INSPEC and COMPENDEX have overlapping domains, and *Ind* is not designed to return all of the relevant databases, but only the most promising ones. Finally, the *Success* values for the rest of the criteria are relatively high for all of the pairs of databases.

5.3 Evaluating *Ind* over six databases

In this section we report some results for the basic configuration, as defined in Figure 6. So, DB ={INSPEC, COMPENDEX, ABI, GEOREF, ERIC, PSYCINFO}, and *Ind* is used as the estimator. Figure 13 summarizes the results corresponding to our six different evaluation criteria. This figure shows that the same phenomena described in Section 5.2 prevail, although in general the success rates are lower. The reason for this is that we are now considering six databases instead of just two. For example, $Success(C_{EX}, Ind)$ is notoriously lower. As was mentioned above, this can be explained by the fact that *Ind* chooses only the most promising databases, not all of the ones that might contain relevant documents (see Section 6.6). Still, $Success(C_{AB}, Ind)$ is relatively high (88.95%), showing *Ind*'s ability to predict what the best databases are. Also, the *Success* figures for C_{OB} and C_{SM} are high (84.38% and 91.26%, respectively), making *Ind* useful for exploring *some* of the relevant/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 might contain some document relevant to query q. Therefore, for 91.26% of the queries, *Ind* chooses databases that actually are relevant, provided there are any, what makes *Ind* particularly good for the C_{SM} semantics. However, *Ind* fails in many cases to isolate *all* of the relevant databases (thus making $Beta(C_{SM}, Ind)$ high: 80.64%).

Figure 14 shows the average values of $P_{Relevant}$, $R_{Relevant}$, P_{Best}, R_{Best}, $P_{Relevant_{INSPEC}}$, and $R_{Relevant_{INSPEC}}$ (see Equations 7 through 10) obtained for the basic configuration of the experiments. Note that the average $P_{Relevant}$ equals $Success(C_{SM}, Ind)/100$. This is not surprising, since

	C_{EX}	C_{AB}	C_{OB}	C_{SM}	$C^{INSPEC}_{EX/AB}$	$C^{INSPEC}_{OB/SM}$
ERIC, INSPEC	54.13	98.71	91.50	92.21	93.63	85.33
ERIC, COMPENDEX	54.75	98.58	88.79	89.55	N/A	N/A
ERIC, PSYCINFO	61.68	97.36	73.90	74.86	N/A	N/A
ERIC, GEOREF	61.52	97.83	73.70	74.45	N/A	N/A
ERIC, ABI	60.04	97.29	75.92	76.76	N/A	N/A
INSPEC, COMPENDEX	21.40	90.94	86.24	91.91	84.40	74.15
INSPEC, PSYCINFO	56.97	99.04	91.87	92.40	96.07	87.99
INSPEC, GEOREF	51.52	98.78	91.13	91.84	87.62	78.89
INSPEC, ABI	50.99	97.88	90.95	92.10	94.26	85.76
COMPENDEX, PSYCINFO	57.30	98.83	88.84	89.40	N/A	N/A
COMPENDEX, GEOREF	52.02	98.45	88.14	88.95	N/A	N/A
COMPENDEX, ABI	51.21	98.13	88.34	89.47	N/A	N/A
PSYCINFO, GEOREF	63.06	98.06	73.18	73.95	N/A	N/A
PSYCINFO, ABI	63.03	97.71	76.95	77.83	N/A	N/A
GEOREF, ABI	61.97	97.51	75.47	76.50	N/A	N/A

Figure 12: $Success(C, Ind)$, for the different criteria C and 15 pairs of databases.

Criteria	$Success$	$Alpha$	$Beta$	$Success - Beta$
C_{EX}	17.50	82.50	6.89	10.61
C_{AB}	88.95	11.05	6.89	82.06
C_{OB}	84.38	15.62	2.32	82.06
C_{SM}	91.26	8.74	80.64	10.61
$C^{INSPEC}_{EX/AB}$	70.12	29.88	11.02	59.10
$C^{INSPEC}_{OB/SM}$	59.10	40.90	0	59.10

Figure 13: Evaluation criteria for DB ={INSPEC, COMPENDEX, ABI, GEOREF, ERIC, PSYCINFO}, using *Ind* as the estimator.

Set	P_{Set}	R_{Set}
$Relevant$	0.9126	0.4044
$Best$	0.8438	0.9010
$Relevant_{INSPEC}$	0.5966	0.7012

Figure 14: Average values of the different P and R functions for the basic configuration of the experiments.

Source of the trace	ERIC database
Raw trace queries	3050
Experiment queries	2404
Percentage experiment queries of raw queries	78.82

Figure 15: Characteristics of the set $TRACE_{ERIC}$ of queries used in the experiments.

the condition $P_{Relevant} = 1$ is equivalent to the condition for satisfying criterion C_{SM} (see Figure 2), and very few times does *Ind* choose more than one database in $Chosen_{Ind}$, as mentioned in Section 5.2. In particular, for only 96 out of the 6897 $TRACE_{INSPEC}$ queries does $Chosen_{Ind}$ consist of more than one database. Furthermore, 95 of these 96 queries are one-term queries, for which $Chosen_{Ind} = Best$ necessarily. So, if $Chosen_{Ind}(q, DB)$ is a singleton, either $Chosen_{Ind}(q, DB) \subseteq Relevant(q, DB)$, in which case $P_{Relevant}(q, DB) = 1$ and q contributes to $Success(C_{SM}, Ind)$, or $Chosen_{Ind}(q, DB) \cap Relevant(q, DB) = \emptyset$, $P_{Relevant}(q, DB) = 0$, and q does not contribute to $Success(C_{SM}, Ind)$. This explains why $P_{Relevant}$ equals $Success(C_{SM}, Ind)/100$. A similar explanation applies to P_{Best} being equal to $Success(C_{OB}, Ind)/100$, and to $P_{Relevant_{INSPEC}}$ being almost identical to $Success(C^{INSPEC}_{OB/SM}, Ind)/100$.

On the other hand, the average $R_{Relevant}$ is higher than $Success(C_{EX}, Ind)/100$. The reason for this is that *Ind* attempts to identify the most promising databases, not all of the ones containing relevant documents (this is why $Success(C_{EX}, Ind)$ is low). A query q such that $\emptyset \neq Chosen_{Ind}(q, DB) \subset Relevant(q, DB)$ will not contribute to $Success(C_{EX}, Ind)$, and, even though $R_{Relevant}(q, DB) \neq 1$, $R_{Relevant}(q, DB) = \frac{|Chosen_{Ind}(q,DB)|}{|Relevant(q,DB)|} > 0$, making the average $R_{Relevant}$ higher. The same is true, but to a more limited extent due to the higher *Success* values, for R_{Best} and $Success(C_{AB}, Ind)$. $R_{Relevant_{INSPEC}}$ and $Success(C^{INSPEC}_{EX/AB}, Ind)/100$ are equal, since $Relevant_{INSPEC}$ is either the empty set or the singleton {INSPEC}. Therefore, either $Relevant_{INSPEC}(q, DB) \subseteq Chosen_{Ind}(q, DB)$, in which case $R_{Relevant_{INSPEC}} = 1$ and q "contributes" to $Success(C^{INSPEC}_{EX/AB}, Ind)$, or $Relevant_{INSPEC}(q, DB) \not\subseteq Chosen_{Ind}(q, DB)$, in which case $Relevant_{INSPEC}(q, DB) \cap Chosen_{Ind}(q, DB) = \emptyset$, and so q does not contribute to $Success(C^{INSPEC}_{EX/AB}, Ind)$ and $R_{Relevant_{INSPEC}} = 0$. Therefore, $R_{Relevant_{INSPEC}}$ and $Success(C^{INSPEC}_{EX/AB}, Ind)/100$ must be equal.

5.4 Impact of using other traces

So far, all of our experiments were based on the set of 6897 queries from the INSPEC trace, namely, $TRACE_{INSPEC}$, as described in Section 4. To analyze how dependent the results are from the trace used, we ran our experiments using a different set of queries. These queries were issued by real users to the ERIC database, between 3/28/1993 and 4/10/1993. The trace was processed in the same way as the INSPEC trace (see Section 4). Figure 15 summarizes some information about the ERIC trace. We will refer to this set of queries as $TRACE_{ERIC}$.

Figure 16 shows the results for the different evaluation criteria, for the basic configuration (see

Criteria	$Success$	$Alpha$	$Beta$	$Success - Beta$
C_{EX}	28.62	71.38	9.28	19.34
C_{AB}	93.39	6.61	9.28	84.11
C_{OB}	84.94	15.06	0.83	84.11
C_{SM}	89.64	10.36	70.30	19.34
$C_{EX/AB}^{ERIC}$	68.76	31.24	14.23	54.53
$C_{OB/SM}^{ERIC}$	54.53	45.47	0	54.53

Figure 16: Evaluation criteria for DB ={INSPEC, COMPENDEX, ABI, GEOREF, ERIC, PSYCINFO} and *Ind* as the estimator (ERIC trace).

Figure 6) but using $TRACE_{ERIC}$. The results obtained differ only slightly from the ones reported in Figure 13 for $TRACE_{INSPEC}$.

5.5 Impact of the query size

It is interesting to classify the queries according to their number of keyword-field designation pairs and analyze *Ind*'s performance over each of these groups. Figures 17 and 18 show some of the results for the basic configuration (Figure 6) but split by query size. Results are reported for queries up to size four, since there are very few queries in $TRACE_{INSPEC}$ consisting of more than four terms. Figure 19 summarizes the number of $TRACE_{INSPEC}$ queries for the different query sizes.

As was mentioned in Section 5.1, *Ind* chooses exactly the *Best* set for queries of size one. This is why, for this query size, $Success(C_{AB}, Ind) = Success(C_{SM}, Ind) = 100\%$. Surprisingly, some criteria perform better as the number of terms per query grows, while others perform worse. For instance, the *Ind* performance according to C_{EX} gets better for larger queries: as pointed out before, *Ind* chooses only the databases that are predicted the best. The more terms a query contains, the smaller the *Relevant* subset of DB will tend to be, thus making $Chosen_{Ind}$ a better approximation of it. This is also true of C_{AB} (after an initial decline of the performance from query size 1 to query size 2, since for size 1, $Chosen_{Ind} = Best$).

The reverse effect takes place for C_{OB} and C_{SM}: the associated performance tends to get worse as *Best* and *Relevant* shrink, respectively: if a query contains, say, four terms, it is more likely that there will be no documents containing all of the four terms, even though there are documents containing each of the terms, but not all. So, it is more likely that $Chosen_{Ind}$ will contain some "non-relevant" or "non-best" database.

6 Improving *GlOSS*

In this section we study the space requirements of *GlOSS* and compare them with those of a full index of the set of databases. We also analyze ways of reducing the size of the database histograms kept by *GlOSS*. We then introduce slight variations to the definition of the $Chosen_{EST}$ and *Best* sets in order to make them more comprehensive, and present two new estimators, *Min* and *Binary*.

6.1 Eliminating the "subject" index

Before we compute the histogram sizes, we will analyze the way the "subject" index is treated in the six databases we considered. In all of these databases, "subject" is a compound index, built by merging together other "primitive" indexes. For example, in the INSPEC database, the

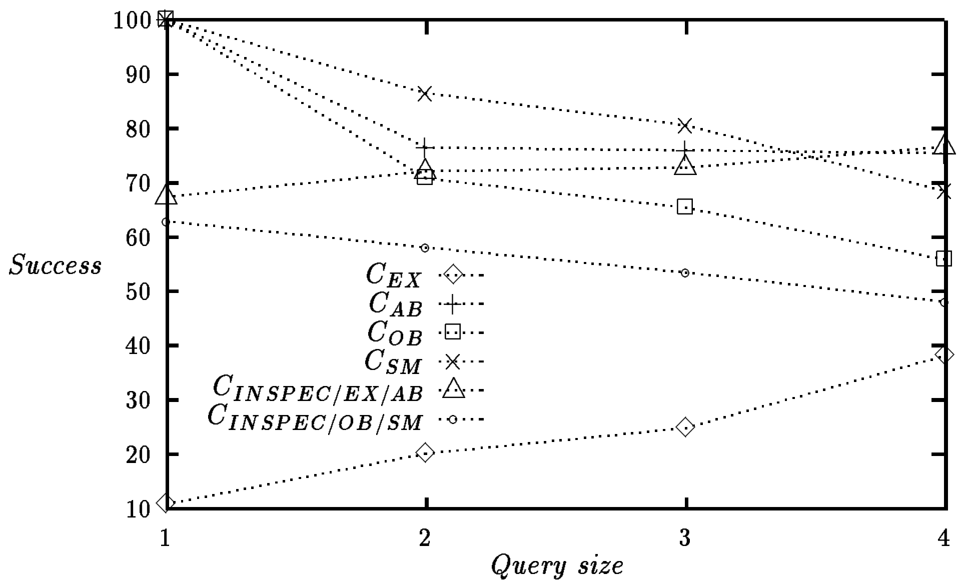

Figure 17: *Success*(C, Ind) as a function of the query size.

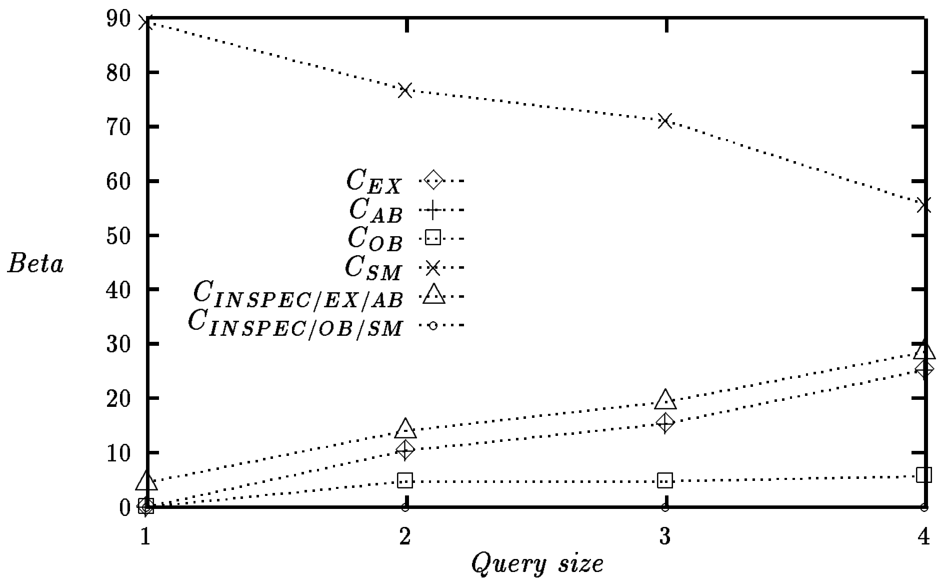

Figure 18: *Beta*(C, Ind) as a function of the query size.

Query size	Number of queries	% of total
1	3692	53.53
2	1720	24.94
3	1020	14.79
4	333	4.83
5 or more	132	1.92
Total	6897	100

Figure 19: Number of queries in $TRACE_{INSPEC}$ for the different query sizes.

Database	Indexes
INSPEC	title, abstract, thesaurus, organization, (other subjects)
COMPENDEX	title, abstract, thesaurus, conference, (other subjects)
ABI	title, abstract, thesaurus, organization, class, (places)
GEOREF	title, abstract, thesaurus
ERIC	title, abstract, thesaurus, organization, (geographic area), (other subjects)
PSYCINFO	title, abstract, thesaurus, (keyphrase)

Figure 20: The primitive indexes used to build the "subject" index in the six databases considered.

"subject" index is constructed by merging the "title", "abstract", "thesaurus", "organization", and "other subjects" indexes (see Figure 20). Therefore, a query "*find subject computers*" is in reality equivalent to the following "or" query:

$$\begin{aligned}&\textit{find title computers} \lor \textit{abstract computers} \lor \textit{thesaurus computers}\\ &\quad\lor \textit{organization computers} \lor \textit{other subjects computers}\end{aligned}$$

Figure 20 shows what indexes the "subject" index is built from for the six databases.

All of the experiments we reported so far treated "subject" as a primitive index, as though *GlOSS* kept the entries corresponding to the "subject" field designation in the database histograms. In this Section, we investigate whether this is actually necessary from a performance point of view. Given that *GlOSS* has the entries for the constituent indexes from which the "subject" index is formed, we could attempt to *estimate* the entries corresponding to the "subject" index using the entries for the primitive indexes. This way, we can save space by not having to store entries for the "subject" index. For example, given that we know *freq(title computers,* INSPEC), *freq(abstract computers,* INSPEC), and so on, can we estimate *freq(subject computers,* INSPEC) instead of storing this piece of information explicitly?

There are different ways to estimate *freq(subject <w>, <db>)*, given the primitive indexes $index_1, index_2, \ldots, index_n$ the "subject" index is built from in database $< db >$. One such way is to take the maximum of the individual frequencies for the primitives indexes:

$$\textit{freq(subject <w>, <db>)} \approx \max_{i=1,\ldots,n} \textit{freq}(index_i < w >, <db>) \tag{14}$$

Note that this estimate constitutes a lower bound for the actual value of *freq(subject <w>, <db>)*.

Figure 21 shows the results obtained for the basic configuration (see Figure 6) but estimating the "subject" frequencies as in Equation 14, with one difference: only those indexes that actually appeared in $TRACE_{INSPEC}$ queries were considered. The other indexes are seldom used so it does not make sense for *GlOSS* to keep statistics on them. The indexes considered are the ones that are listed in Figure 23. The indexes that appear between parentheses in Figure 20 were ignored, since no information was kept about them by *GlOSS* in our experiments. For example, *freq(subject computers*, INSPEC) was estimated as:

$$\begin{aligned}\textit{freq(subject computers}, \text{INSPEC}) \approx\ & \max\{\textit{freq(title computers}, \text{INSPEC}),\\ & \textit{freq(abstract computers}, \text{INSPEC}),\\ & \textit{freq(thesaurus computers}, \text{INSPEC}),\\ & \textit{freq(organization computers}, \text{INSPEC})\}\end{aligned}$$

Criteria	$Success$	$Alpha$	$Beta$	$Success - Beta$	$Success$
C_{EX}	17.41	82.59	6.92	10.50	17.50
C_{AB}	88.23	11.77	6.93	81.30	88.95
C_{OB}	83.82	16.18	2.52	81.30	84.38
C_{SM}	91.21	8.79	80.72	10.50	91.26
$C^{INSPEC}_{EX/AB}$	70.18	29.82	10.89	59.29	70.12
$C^{INSPEC}_{OB/SM}$	59.29	40.71	0	59.29	59.10

Figure 21: Evaluation criteria for the basic configuration, but estimating the "subject" frequencies as the *maximum* of the frequencies of the primitives indexes. The last column shows the *Success* values for the basic configuration using the exact "subject" frequencies.

Criteria	$Success$	$Alpha$	$Beta$	$Success - Beta$	$Success$
C_{EX}	17.47	82.53	6.90	10.57	17.50
C_{AB}	88.30	11.70	6.92	81.38	88.95
C_{OB}	83.82	16.18	2.44	81.38	84.38
C_{SM}	91.33	8.67	80.76	10.57	91.26
$C^{INSPEC}_{EX/AB}$	69.91	30.09	10.98	58.94	70.12
$C^{INSPEC}_{OB/SM}$	58.94	41.06	0	58.94	59.10

Figure 22: Evaluation criteria for the basic configuration, but estimating the "subject" frequencies as the *sum* of the frequencies of the primitives indexes. The last column shows the *Success* values for the basic configuration using the exact "subject" frequencies.

Thus, the "other subjects" index is simply ignored in this estimate [13]. The last column in Figure 21 shows the *Success* figures for the basic configuration, using the exact frequencies for the "subject" index: there is very little change in performance if we estimate the "subject" frequencies as in Equation 14.

Alternatively, we can estimate *freq(subject <w>, <db>)* as:

$$\textit{freq(subject <w>, <db>)} \approx \sum_{i=1,\ldots,n} \textit{freq}(\textit{index}_i < w >, <\textit{db}>) \quad (15)$$

Note that this estimate constitutes an upper bound for the actual value of *freq(subject <w>, <db>)*.

Figure 22 shows the results obtained for the basic configuration (see Figure 6) but estimating the "subject" frequencies as in Equation 15, ignoring those indexes not appearing in $TRACE_{INSPEC}$ queries, as explained above.

Figures 21 and 22 show that either way we estimate the "subject" frequencies, the performance of *Ind* does not vary significantly from that obtained when using the exact frequencies for the "subject" indexes (see Figure 13). Therefore, when we compute the size of the *GlOSS* histograms in the next section, we will assume that "subject" histograms are not stored. Only primitive indexes appearing in $TRACE_{INSPEC}$ queries will be taken into account.

6.2 Characteristics of the histograms and full indexes

[13]In fact, the "other subjects" index does not appear in any of the queries of $TRACE_{INSPEC}$ since it can only be used in *browse* mode in the INSPEC database under FOLIO.

Field Designator	Full Index	*GlOSS* (*threshold=0*)
	# of postings	# of entries
Author	4108027	311632
Title	10292321	171537
Publication	6794557	18411
Abstract	74477422	487247
Thesaurus	11382655	3695
Conference	7246145	11934
Organization	9374199	62051
Class	4211136	2962
Numbers (ISBN, ...)	2445828	12637
Report Numbers	7833	7508
Totals	130340123	1089614

Figure 23: Characteristics of the histogram kept by *GlOSS* vs. those of a full index, for the INSPEC database.

As explained in Section 2.2, the estimators need to keep, for each database, the number of documents that satisfy each possible keyword-field designation pair. Figure 23 was generated using information of the corresponding INSPEC indexes obtained from Stanford's FOLIO library information retrieval system. The "# of entries" column reports the number of entries required for each of the INSPEC indexes appearing in the $TRACE_{INSPEC}$ queries. For example, there are $311,632$ different author surnames appearing in INSPEC (field designation "author"), and each will have an associated frequency in the histogram for INSPEC. A total of $1,089,614$ entries will be required for the histogram for the INSPEC database. Each of these entries will correspond to a term and its associated frequency (e.g. <*author Knuth, 47*>, meaning that there are 47 documents in INSPEC with *Knuth* as the author). In contrast, if we were to keep the complete inverted lists associated with the different indexes we considered, $130,340,123$ postings would have to be stored in the full index.

6.3 Storage cost estimates

In the following, we will roughly estimate the space requirements of a full index vs. those of the histogram kept by the estimators, for the INSPEC database. The figures we will produce should be taken just as an indication of the relative order of magnitude of the corresponding requirements.

Each of the *postings* of a full index will typically contain the following information:

- a field designation,
- a document identifier, and
- a count or position of the word in the document.

If we dedicate one byte for the field designation, three bytes for the document identifier, and one byte for the position in the document, we end up with five bytes per posting. Let us assume that, after compression, 2.5 bytes suffice per posting (compression of 50% is typical for inverted lists).

Each of the *frequencies* kept by an estimator will typically contain the following information:

Size of	Full Index	*GlOSS/threshold=0*
Vocabulary	3.13 MBytes	3.13 MBytes
Index	310.76 MBytes	2.60 MBytes
Total	313.89 MBytes	5.73 MBytes
% of Full Index	100	1.83

Figure 24: Estimated storage costs of a full index vs. the *GlOSS* histogram for the INSPEC database.

- a field designation,
- a database identifier, and
- the frequency itself.

Regarding the size of the frequencies themselves, only 1417 keyword-field designation pairs in INSPEC have more than 2^{16} documents containing them. Therefore, in the vast majority of the cases, two bytes suffice to store these frequencies, according to the INSPEC data we have available. So, we can encode the frequencies by letting one bit indicate whether the frequency is a two-byte or a three-byte frequency, for example. Since the number of keyword-field designation pairs that require three bytes is so small, to estimate storage costs we can assume that two bytes are dedicated per frequency. So, using one byte for the field designation and two bytes for the database identifier, we end up with five bytes per frequency of the histogram. Again, after compression we will assume that 2.5 bytes are required per frequency. Using the data from Figure 23 and our estimates for the size of each posting and histogram entry, we obtain the index sizes shown in Figure 24 ("Index" row).

The vocabulary itself will presumably be stored at the leaf level of some search structure. Let us assume that such a search structure is a B^+-tree. Given that the leaf nodes of the tree are reasonably big, the dominant cost of the search structure will be the word storage. We will focus on this cost in what follows. As a result of using B^+-trees, the vocabulary will appear in sorted order in the leaves of the tree. We can take advantage of this situation by compressing the keywords themselves, to reduce the amount of storage required for them. We will assume that after compression, five letters per word suffice, and each letter is stored in five bits. So, four bytes will suffice to store each keyword in a leaf of the B^+-tree. The vocabulary for INSPEC, ignoring the field designators and including only those indexes appearing in $TRACE_{INSPEC}$ queries, consists of $819,437$ words (since we are not considering the field designators here a keyword that is associated with more than one field designator appears only once in this vocabulary). Therefore, around $4 \times 819,437$ bytes, or 3.13 MBytes are needed to store the INSPEC vocabulary. This is shown in the "Vocabulary" row of Figure 24. So, the size of the histogram needed by *GlOSS* is only around 1.83% the size of the corresponding full index, for the INSPEC database.

So far, we have only focused on the space requirements of a single database, namely INSPEC. We will base the space requirement estimates for the six databases on the figures we have just computed for the INSPEC database, for which we have reliable index information. To do this, we will multiply the different values we calculated for INSPEC by a growth factor G (see Figure 4):

$$G = \frac{\sum_{db \in \{\text{INSPEC,COMPENDEX,ABI,GEOREF,ERIC,PSYCINFO}\}} DBSize(db)}{DBSize(\text{INSPEC})} \approx 4.12$$

Therefore, the number of postings required by a full index of the six databases is estimated as $G \times$ INSPEC number of postings $= 537,001,307$ postings, or around 1280.31 MBytes. The number

Size of	Full index	*GlOSS/threshold=0*
Vocabulary	11.59 MBytes	11.59 MBytes
Index	1280.31 MBytes	10.70 MBytes
Total	1291.90 MBytes	22.29 MBytes
% of Full index	100	1.73
$Success(C_{AB},\ \_)$	100	88.23
$Success(C_{AB},\ \_)$ - $Beta(C_{AB},\ \_)$	100	81.30

Figure 25: Storage estimates for *GlOSS* and a full index for the six databases. The entries for *GlOSS* in the last two rows correspond to the basic configuration (*Ind* as the estimator), but estimating the "subject" frequencies as the maximum of the frequencies of the primitive indexes.

of frequencies required by *GlOSS* for the six databases is estimated as $G\times$ INSPEC number of frequencies $= 4,489,210$ frequencies, or around 10.70 MBytes (see the "Index" row of Figure 25).

The space occupied by the index keywords of the six databases considered will be proportional to the size of their *merged* vocabularies. Using index information from Stanford's FOLIO library information retrieval system, we can determine that the size of the merged vocabulary of the six databases we considered is approximately 90% of the sum of the six individual vocabulary sizes. Therefore, we estimate the size of the merged vocabulary for the six databases as $G \times 0.90\times$ INSPEC vocabulary size = 3,038,472 words, or around 11.59 MBytes (see the "Vocabulary" row of Figure 25).

Figure 25 summarizes the storage estimates for *GlOSS* and a full index. Note that the *GlOSS* histogram is only 1.73% the size of the full index. This is even less than the corresponding figure we obtained above just for the INSPEC database (1.83%). The reason for this is the fact that the merged vocabulary size is only 90% of the sum of the individual vocabulary sizes. Although this 10% reduction "benefits" both *GlOSS* and the full index case, the impact on *GlOSS* is higher, since the vocabulary size is a much larger fraction of the total storage needed by *GlOSS* than it is for the full index.

The numbers of Figure 25 have been obtained using some very rough estimates and approximations, so they should be taken cautiously. However, we think they are useful to illustrate the low space requirements of *GlOSS*: around 22.29 MBytes would suffice to keep the histograms for the six databases we studied.

6.4 Pruning the histograms

To further reduce the amount of information that we keep about each database, we introduce the notion of a *threshold*. If a database *db* has fewer than *threshold* documents with a given keyword-field pair t, then this information will not be stored in the corresponding histogram. Therefore, it will be assumed that $freq(t, db)$ is 0 whenever this data is needed.

As a result of the introduction of *threshold*, the estimator may now conclude that some database *db* does not contain any relevant documents for a query:

$$find\ t_1 \wedge \ldots \wedge t_n$$

if $freq(t_i, db)$ is missing, for some i, while in fact *db* does contain relevant documents for the query. This situation was not possible before: if $freq(t_i, db)$ was missing from the information set of the estimator, then $freq(t_i, db) = 0$, and so, there could be no documents in *db* satisfying such a query. This is why $Beta(C^{INSPEC}_{OB/SM}, Ind)$ may now be greater than 0 for *threshold*> 0 (see Section 5.2).

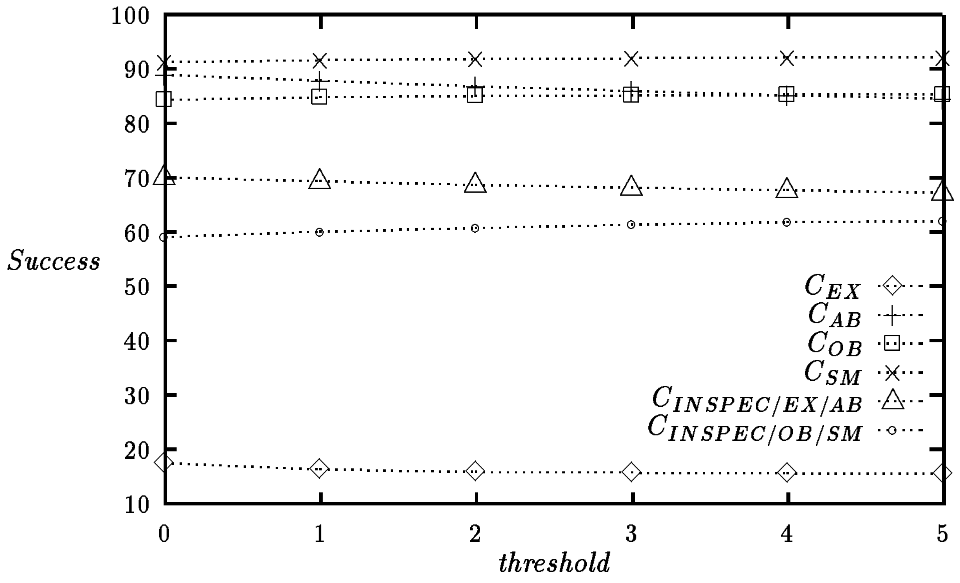

Figure 26: *Success*(C, Ind) as a function of *threshold*.

To see if *Ind*'s performance deteriorates by the use of this *threshold*, Figures 26, 27, 28, and 29 show some results for different values of *threshold*, for the basic configuration, and using the exact figures for the "subject" index entries. On the other hand, Figures 30, 31, 32, and 33 show some results for different values of *threshold*, for the basic configuration, but estimating the "subject" index entries as in Equation 14. These figures show that the performance for the different criteria is only slightly sensitive to (small) increases in *threshold*. In fact, the *Success* values for criterion C_{SM}, for example, tend to improve for higher values of *threshold*. The reason for this is that $Chosen_{Ind}$ does not include databases with $\mathsf{ESize}_{Ind} = 0$. By increasing *threshold*, the number of such databases will presumably increase, thus making $Chosen_{Ind}$ smaller, and more likely to satisfy $C_{SM} : Chosen_{Ind} \subseteq Relevant$. Analogous explanations apply to the improvement in *Success* for criteria C_{OB} and $C_{OB/SM}^{INSPEC}$.

The reason for introducing *threshold*s is to have to store less information for the estimator. Figure 34 reports the number of entries that would be left, for different field designators, in the histogram for the INSPEC database. Some field designators (e.g. "thesaurus") are not affected much by this pruning of the smallest entries, whereas the space requirements for some others (e.g. "author", "title", and "abstract") are reduced drastically. Adding together all of the indexes, the number of entries in the histogram for INSPEC decreases very fast as *threshold* increases: for *threshold*=1, for instance, 508,978 entries, or 46.71% of the original number of entries, are eliminated. Therefore, the size of the *GlOSS* histograms can be substantially reduced beyond the already small size estimated in Figure 25.

Another approach to reducing the size of the histograms kept by the estimators would be to classify each of the individual frequencies into one out of, say, eight classes: for example, the 0^{th} class would consist of all the zero frequencies, the 1^{st} class, of the frequencies from 1 to 10, and so on. So, instead of storing the frequencies to form the histograms, we would instead just store the class each frequency belongs to. If we have eight classes, then only three bits will be required per entry of each histogram, at the expense of having less information about the databases. We are

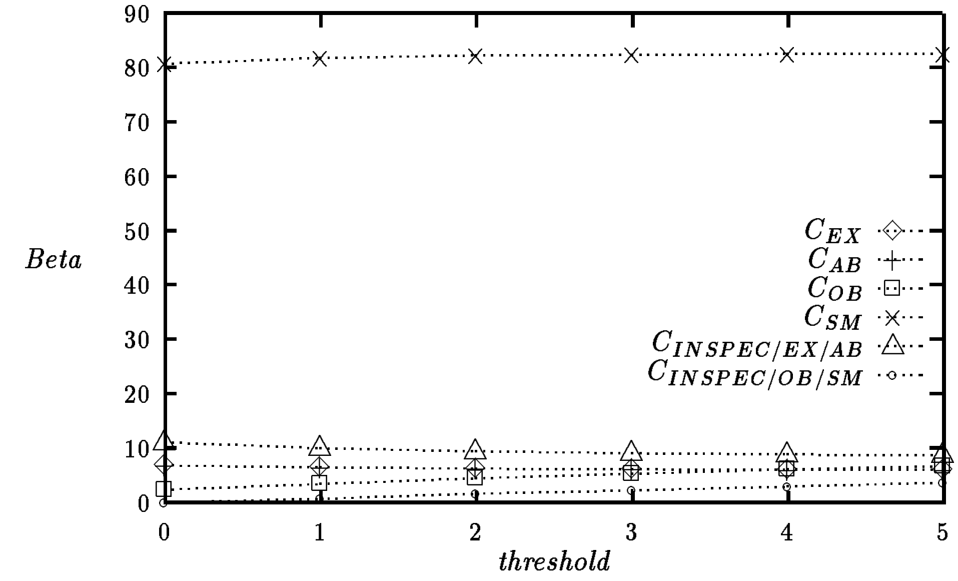

Figure 27: *Beta*(C, *Ind*) as a function of *threshold*.

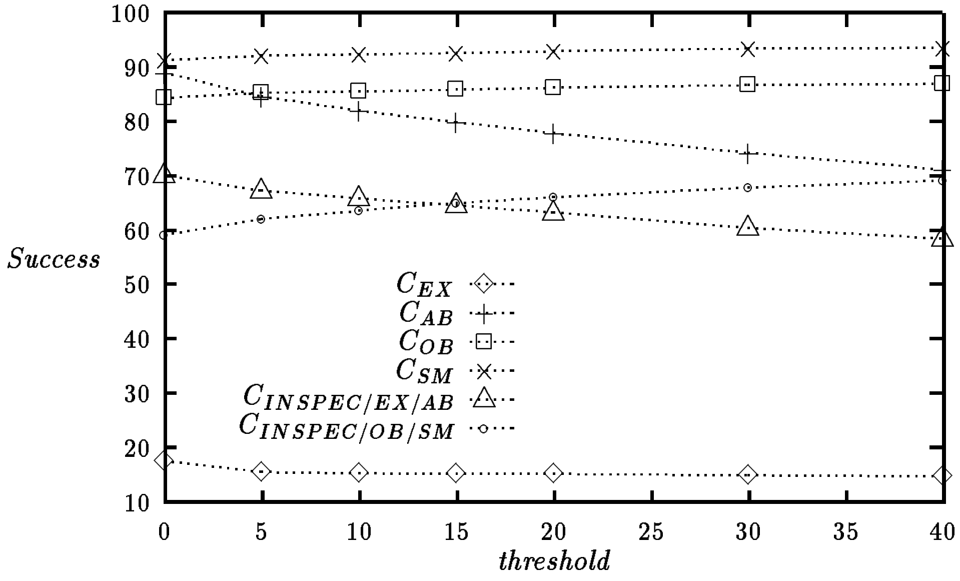

Figure 28: *Success*(C, *Ind*) as a function of higher values of *threshold*.

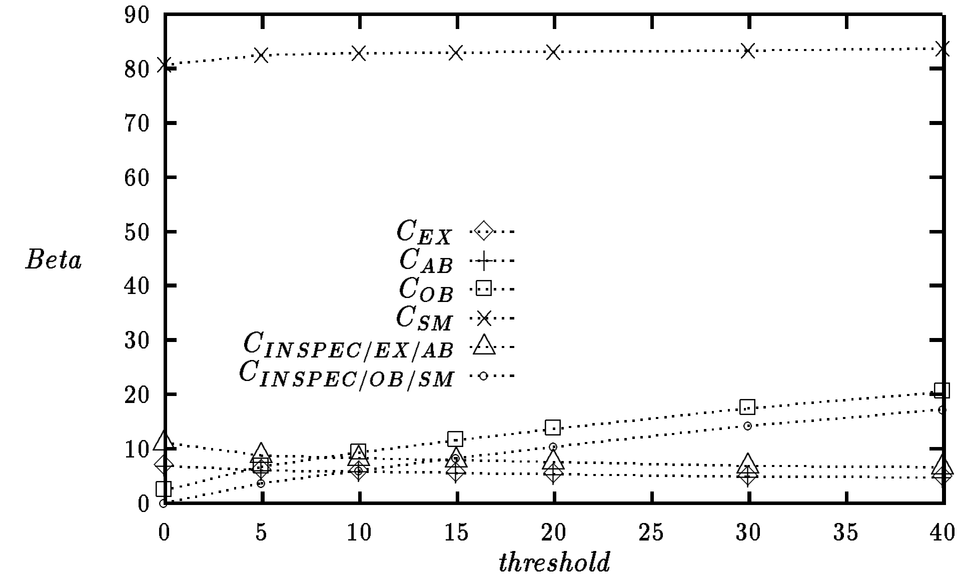

Figure 29: *Beta*(C, *Ind*) as a function of higher values of *threshold.*

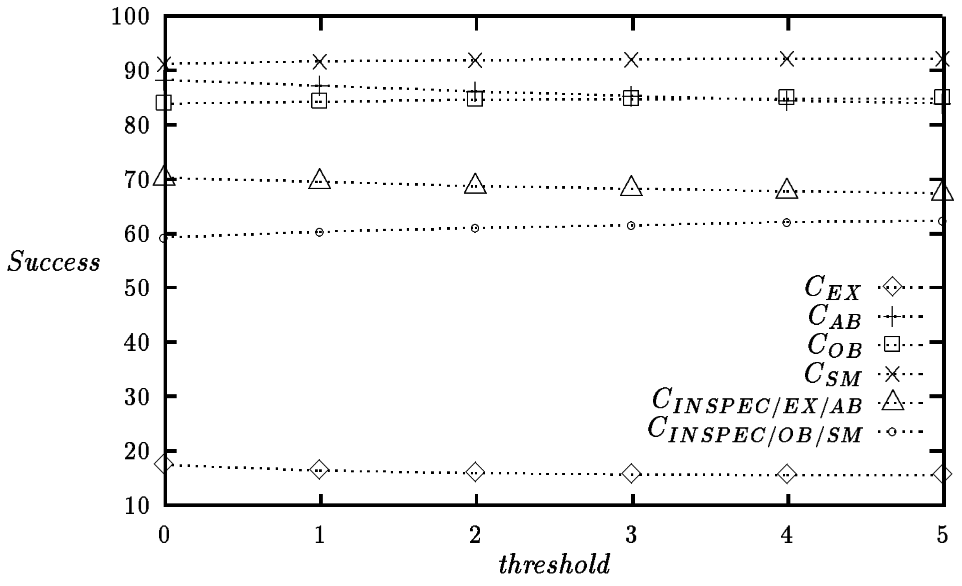

Figure 30: *Success*(C, *Ind*) as a function of *threshold.* The “subject” entries are estimated as the maximum of the entries corresponding to the “primitive” indexes.

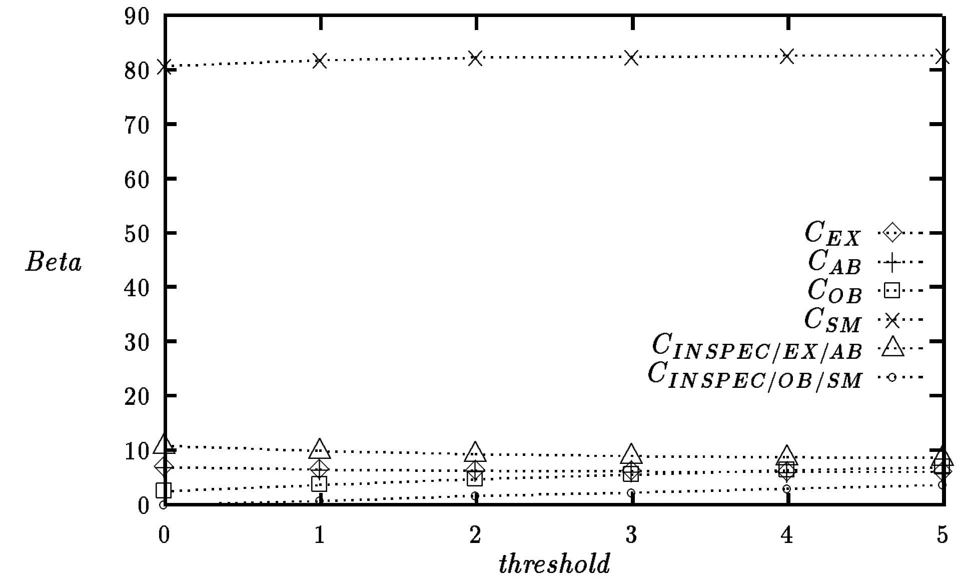

Figure 31: *Beta*(C, *Ind*) as a function of *threshold*. The “subject” entries are estimated as the maximum of the entries corresponding to the “primitive” indexes.

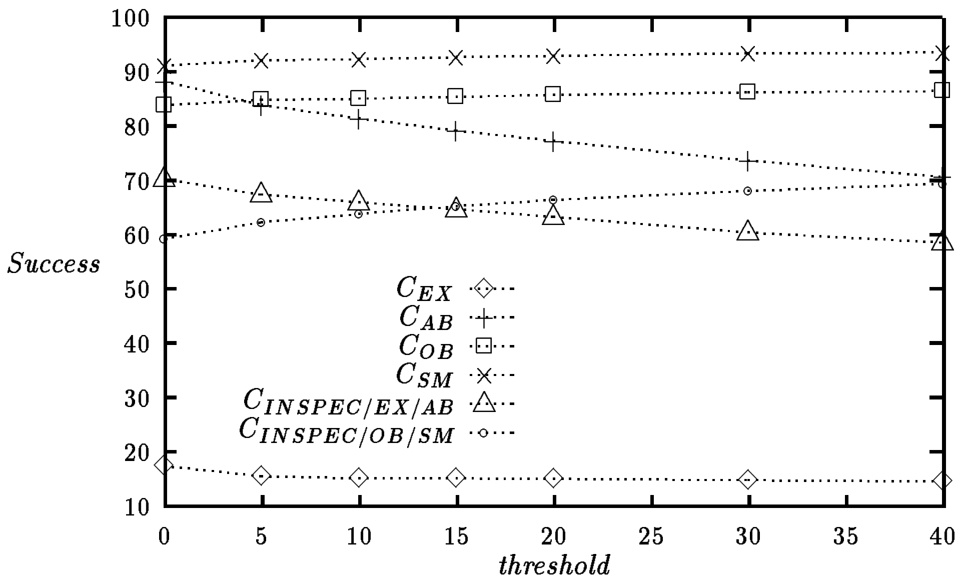

Figure 32: *Success*(C, *Ind*) as a function of higher values of *threshold*. The “subject” entries are estimated as the maximum of the entries corresponding to the “primitive” indexes.

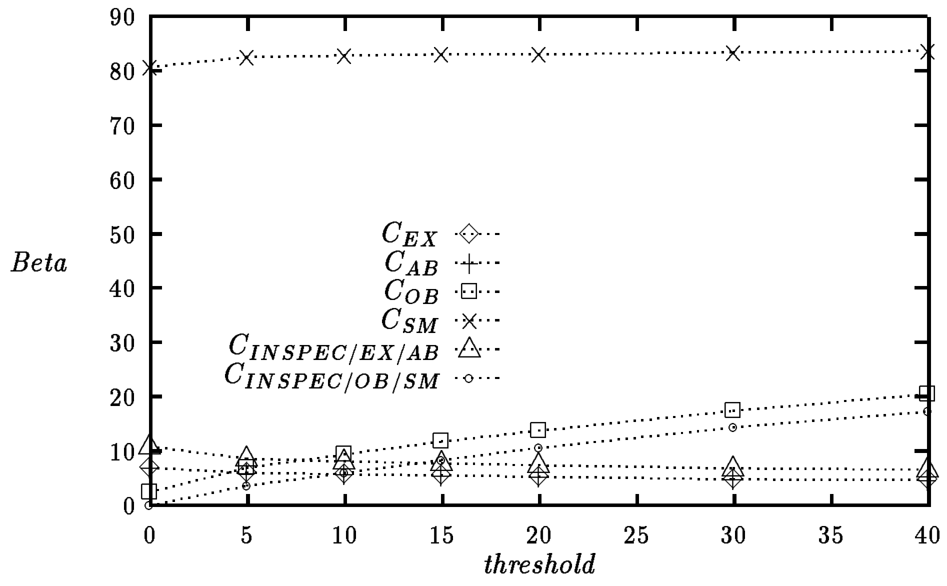

Figure 33: *Beta*(C, *Ind*) as a function of higher values of *threshold*. The "subject" entries are estimated as the maximum of the entries corresponding to the "primitive" indexes.

Field Designator	*threshold*					
	0	1	2	3	4	5
Author	311632	194769	150968	125220	107432	94248
Title	171537	85448	62759	51664	44687	40007
Publication	18411	11666	10042	9281	8832	8535
Abstract	487247	227526	163644	133323	115237	102761
Thesaurus	3695	3682	3666	3653	3641	3637
Conference	11934	10138	9887	9789	9702	9653
Organization	62051	34153	26518	22612	20121	18382
Class	2962	2953	2946	2937	2931	2926
Numbers (ISBN, ...)	12637	10199	10067	9946	9865	9779
Report Numbers	7508	102	37	22	14	12
Totals	1089614	580636	440534	368447	322462	289940
%	100	53.29	40.43	33.81	29.59	26.61
Success(C_{AB}, Ind)	88.23	87.12	86.07	85.28	84.44	83.82
Success(C_{AB}, Ind) - *Beta*(C_{AB}, Ind)	81.30	80.64	79.85	79.15	78.44	77.85

Figure 34: Number of entries left for the different thresholds and field designators in the INSPEC database. The last two rows correspond to the basic configuration, but estimating the "subject" frequencies as the maximum of the frequencies of the primitive indexes, as explained in this Section.

planning on exploring this direction shortly[14].

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

The definitions of $Chosen_{EST}$ and $Best$ given by Equations 2 and 12 are sometimes too inflexible. Consider the following example. Suppose $DB = \{db_1, db_2\}$ 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 12, $Best(q, DB) = \{db_2\}$. But this is probably too arbitrary, as both databases are almost identical regarding the number of relevant documents they have for query q. Also, if the two databases are predicted by an estimator EST to 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 affects the 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 $\epsilon_B, \epsilon_C \geq 0$:

$$\begin{aligned} Chosen_{EST}(q, DB) &= \{db \in DB \mid \mathsf{ESize}_{EST}(q, db) > 0 \wedge \\ &\quad \left|\frac{\mathsf{ESize}_{EST}(q, db) - hest}{hest}\right| \leq \epsilon_C\} && (16) \\ Best(q, DB) &= \{db \in DB \mid \mathsf{RSize}(q, db) > 0 \wedge \\ &\quad \left|\frac{\mathsf{RSize}(q, db) - hreal}{hreal}\right| \leq \epsilon_B\} && (17) \end{aligned}$$

where $hest = max_{db' \in DB}\mathsf{ESize}_{EST}(q, db')$ and $hreal = max_{db' \in DB}\mathsf{RSize}(q, db')$. Therefore, the larger ϵ_B and ϵ_C get, the more databases will tend to be included in $Best$ and $Chosen_{EST}$, respectively. Note that Equations 2 and 12 coincide with Equations 16 and 17 for $\epsilon_B = \epsilon_C = 0$. Also, if $\epsilon_C = 1$, Ind becomes the $Binary$ estimator described in Section 6.6: $Chosen_{Ind}(q, DB)$ thus consists of all of the databases in DB that *might* contain some relevant documents for query q.

To see the impact of introducing ϵ_B and ϵ_C, we repeated some of the experiments. Figures 35 and 36 are for the basic configuration, but with varying $\epsilon_B = \epsilon_C$ $(= \epsilon)$. As ϵ increases, $Chosen_{Ind}$ tends to contain more databases, and so does $Best$. On the other hand, $Relevant$, $Relevant_{INSPEC}$, and $Best_{INSPEC}$ do not vary for $\epsilon > 0$. This explains why $Success(C_{EX}, Ind)$ increases with ϵ: in the extreme case of $\epsilon = 1$, $Chosen_{Ind}$ consists of *all* of the databases that might contain some relevant document for the given query. Consequently, $Success(C_{EX}, Ind) = 100\%$ in this case. On the other hand, $Success(C_{AB}, Ind)$ improves slightly more slowly, since $Best$ also grows as ϵ does. For the same reason, $Success(C_{OB}, Ind)$ does not change as much as ϵ grows.

The values of $Success(C_{SM}, Ind)$ get worse as ϵ increases, since $Relevant$ does not depend on ϵ, and $Chosen_{Ind}$ tends to grow along with ϵ. For $\epsilon = 1$, $Best$ becomes equal to $Relevant$, so criterion C_{EX} coincides with criterion C_{AB}, and C_{OB} with C_{SM}. Note that, also for $\epsilon = 1$, $Beta(C_{SM}, Ind) = Beta(C_{OB}, Ind) = 0$. For this value of ϵ, $Relevant = Best$, as mentioned above. If for some given query, $Chosen_{Ind} \subseteq Relevant$ (and so, criteria C_{OB} and C_{SM} are satisfied), then this query satisfies these criteria strictly, i.e. $Chosen_{Ind} = Relevant$. This is so since for $\epsilon = 1$

[14]Note that the *Binary* estimator of Section 6.6 can be regarded as an extreme instance of this approach, in which there are only two classes: one containing all of the frequencies greater than zero, and the other one containing all of the frequencies equal to zero.

$Chosen_{Ind}$ contains all of the databases that might contain documents relevant to the given query, as pointed out above. Similarly, $Beta(C^{INSPEC}_{OB/SM}, Ind)$ is also 0 for $\epsilon = 1$.

Figures 37 and 38 show the performance of the different evaluation criteria for $\epsilon_C = 0$ and for different values of ϵ_B. So, our estimator remains fixed (since $\epsilon_C = 0$) and so do *Relevant* and $Relevant_{INSPEC}$, which do not depend on the parameter ϵ_B. 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) one of our "target" sets, namely *Best*. As a result of this, the values for C_{EX}, C_{SM}, $C^{INSPEC}_{EX/AB}$, and $C^{INSPEC}_{OB/SM}$ in Figures 37 and 38 remain constant as ϵ_B varies. On the other hand, $Success(C_{AB}, Ind)$ worsens as ϵ_B grows, since *Best* tends to contain more databases, while $Chosen_{Ind}$ remains fixed. This is exactly why $Success(C_{OB}, Ind)$ improves with higher values of ϵ_B. For $\epsilon_B = 1$, $Best = Relevant$, and so, criterion C_{EX} coincides with criterion C_{AB}, and criterion C_{OB} coincides with criterion C_{SM}. As mentioned above, parameter ϵ_B is not a parameter of our estimator, but of the semantics of the queries. Higher values for ϵ_B yield more comprehensive *Best* sets. Therefore, parameter ϵ_B should be fixed according to the desired "meaning" for *Best*.

Figures 39 and 40 show the performance of the different evaluation criteria for $\epsilon_B = 0$ and for different values of ϵ_C. So, the evaluation criteria remain fixed, since *Relevant* and $Relevant_{INSPEC}$ do not change (they do not depend on the ϵ), and neither does *Best* (since $\epsilon_B = 0$). *Ind* does vary, since ϵ_C is variable. Since $Chosen_{Ind}$ tends to cover more databases as ϵ_C grows, $Success(C_{EX}, Ind)$ and $Success(C_{AB}, Ind)$ improve for higher values of ϵ_C. For $\epsilon_C = 1$, $Success(C_{EX}, Ind) = Success(C_{AB}, Ind) = 100\%$, since $Chosen_{Ind}$ contains all of the potentially relevant databases. As mentioned above, *Ind* becomes the *Binary* estimator (Section 6.6) for $\epsilon_C = 1$. On the other hand, $Success(C_{OB}, Ind)$ and $Success(C_{SM}, Ind)$ worsen as ϵ_C grows, for the same reasons. Note that for $\epsilon_C = 1$, $Success(C_{OB}, Ind) \neq Success(C_{SM}, Ind)$, since *Best* and *Relevant* differ ($\epsilon_B = 0$). In general, $Best \subseteq Relevant$, and so, $Success(C_{OB}, Ind) \leq Success(C_{SM}, Ind)$ (see Section 5.2). From Figure 39 we can conclude that the value for ϵ_C should be between 0 and 0.125: C_{OB} and $C^{INSPEC}_{OB/SM}$ deteriorate very quickly for higher values of ϵ_C. If we focus on the individual criteria, the following values of ϵ_C should be used: to optimize the performance of C_{EX}, ϵ_C should be set to one (see Section 6.6). However, the best values for C_{OB} are obtained for $\epsilon_C = 0$. For criterion C_{AB}, the preferred value for ϵ_B should be zero. Even though *Success* increases for higher values of ϵ_B for this criterion, *Beta* also increases, and much faster. For criterion C_{SM}, the highest *Success* value is achieved for $\epsilon_B = 0$. However, the highest figure for $Success - Beta$ is obtained when $\epsilon_B = 1$.

6.6 Other estimators

So far, all of our experiments involved estimator *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 the keywords of a query in the documents of a database follow 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*, by letting:

$$\mathsf{ESize}_{Min}(find\ t_1 \wedge \ldots \wedge t_n, db) \ = \ \min_{i=1}^{n} freq(t_i, db) \tag{18}$$

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

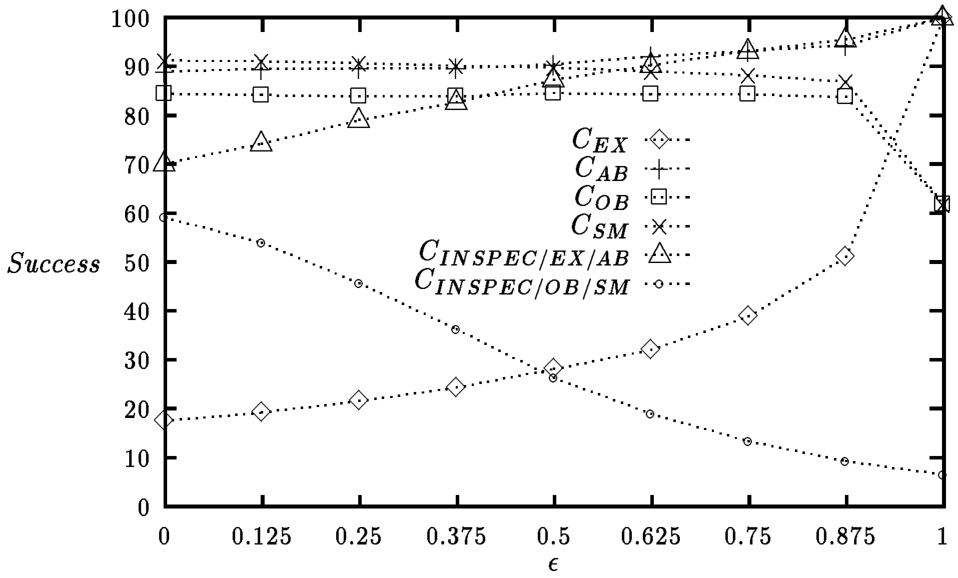

Figure 35: $Success(C, Ind)$ as a function of $\epsilon_B = \epsilon_C (= \epsilon)$.

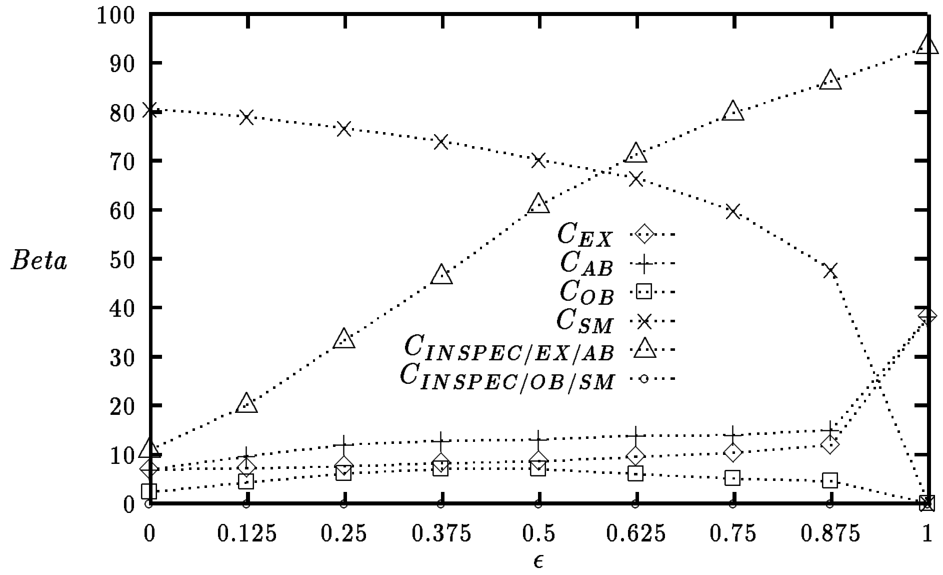

Figure 36: $Beta(C, Ind)$ as a function of $\epsilon_B = \epsilon_C (= \epsilon)$.

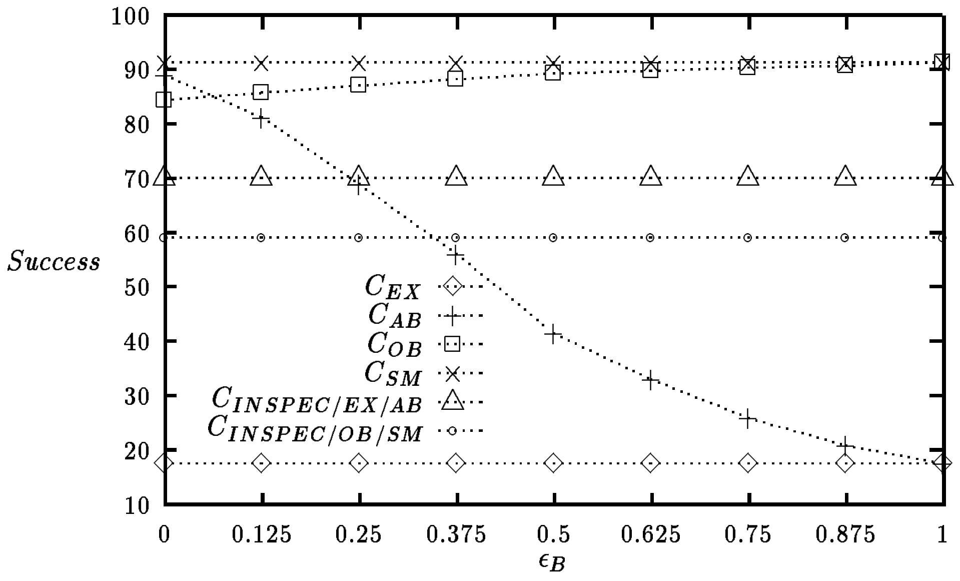

Figure 37: $Success(C, Ind)$ as a function of ϵ_B. $\epsilon_C = 0$.

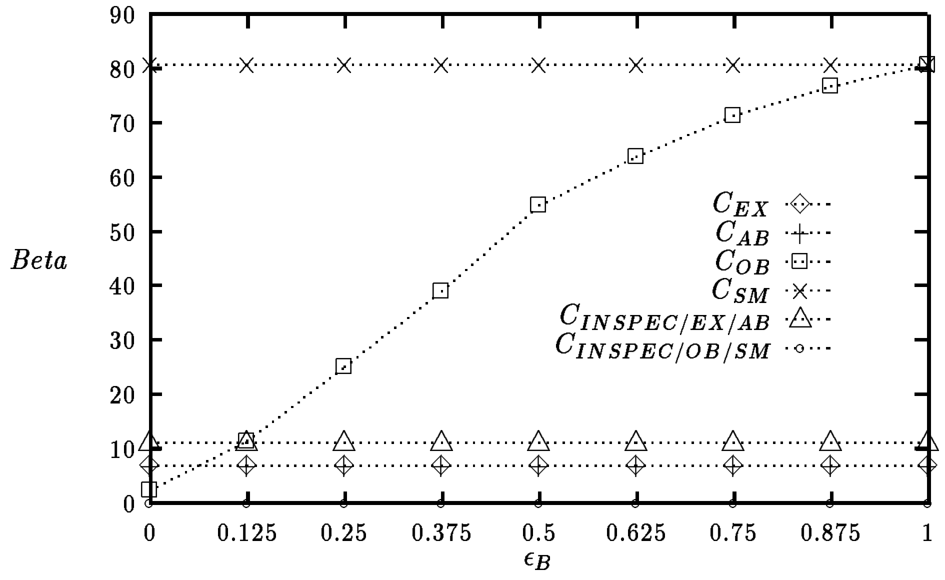

Figure 38: $Beta(C, Ind)$ as a function of ϵ_B. $\epsilon_C = 0$.

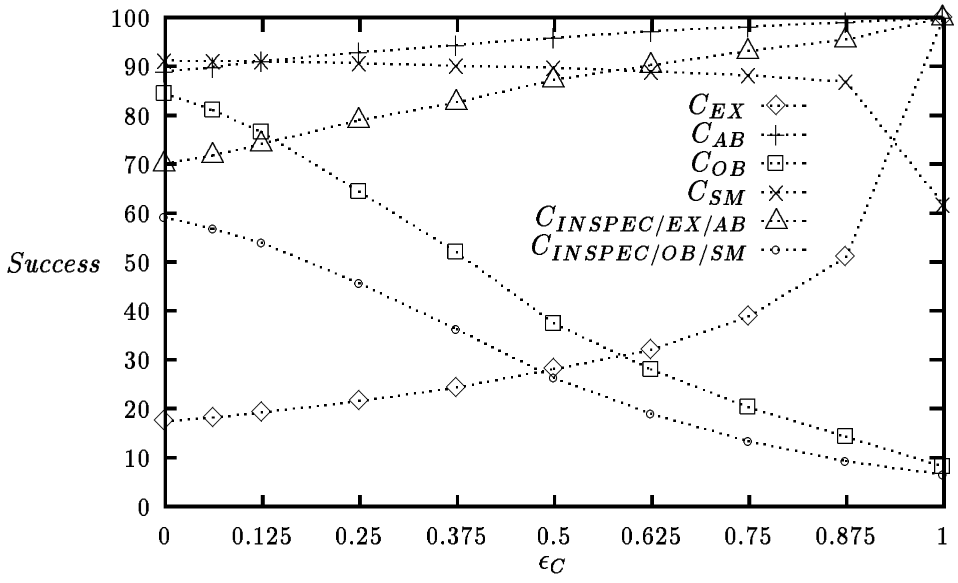

Figure 39: *Success*(C, *Ind*) as a function of ϵ_C. $\epsilon_B = 0$.

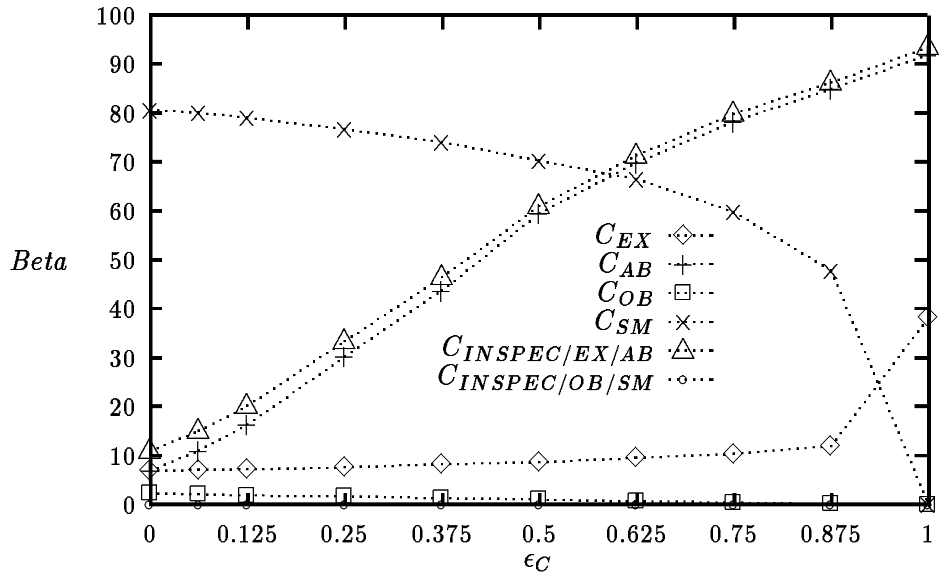

Figure 40: *Beta*(C, *Ind*) as a function of ϵ_C. $\epsilon_B = 0$.

Criteria	*Success*	*Alpha*	*Beta*	*Success* − *Beta*	*Success*
C_{EX}	17.62	82.38	6.92	10.70	17.50
C_{AB}	88.28	11.72	6.99	81.30	88.95
C_{OB}	83.50	16.50	2.20	81.30	84.38
C_{SM}	90.74	9.26	80.03	10.70	91.26
$C_{EX/AB}^{INSPEC}$	73.16	26.84	11.21	61.95	70.12
$C_{OB/SM}^{INSPEC}$	61.95	38.05	0	61.95	59.10

Figure 41: Evaluation criteria for the basic configuration with *Min* as the estimator. The last column shows the *Success* values for the basic configuration, using *Ind* as the estimator.

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

$Chosen_{Min}$ follows from the definition of ESize_{Min}, using Equation 2.

- If our goal is to maximize $Success(C_{EX}, EST)$, then we should be very conservative at dropping databases from the $Chosen_{EST}$ set. With this motivation we define another estimator for *GlOSS*, *Binary*:

$$\begin{aligned}&\mathsf{ESize}_{Binary}(find\ t_1 \wedge \ldots \wedge t_n, db) = \\ &= \begin{cases} 0 & \text{if } \exists i,\ 1 \leq i \leq n | freq(t_i, db) = 0 \\ 1 & \text{otherwise} \end{cases}\end{aligned} \tag{19}$$

  $Chosen_{Binary}$ follows from this definition, using Equation 2.

  So, if our *threshold* is 0 (see Section 6.4), we are guaranteed that $Success(C_{EX}, Binary) = 100$ (at the expense of a high $Beta(C_{EX}, Binary)$, probably). On the other hand, if our *threshold* is greater than 0, this is not necessarily the case, as was explained in Section 6.4.

Figures 41 and 42 show the results obtained for the basic configuration (see Figure 6) but using *Min* and *Binary* as the estimators, respectively. The results for *Min* are very similar to the corresponding results for *Ind*, with no significant differences (see Figure 13). Note that, unlike *Ind*, the definition of ESize_{Min} does not depend on the size of the corresponding database. 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 (see Figures 13 and 41).

As expected, *Binary* gets much higher success values for the C_{EX} criterion, but performs much worse for criterion C_{OB} than *Ind* and *Min*. Even though $Success(C_{AB}, Binary) = 100\%$, $Beta(C_{AB}, Binary)$ is also very high (91.92%). Note that *Beta* is also quite high for criteria C_{EX} and $C_{EX/AB}^{INSPEC}$, since *Binary* tends to produce overly conservative $Chosen_{Binary}$ sets, so as not to miss any of the target databases.

7 Conclusions

In this paper we presented *GlOSS*, a solution to the text database discovery problem. We also developed a formal framework for this problem, together with four different semantics to answer a

Criteria	*Success*	*Alpha*	*Beta*	*Success − Beta*	*Success*
C_{EX}	100	0	38.23	61.77	17.50
C_{AB}	100	0	91.92	8.08	88.95
C_{OB}	8.08	91.92	0	8.08	84.38
C_{SM}	61.77	38.23	0	61.77	91.26
$C_{EX/AB}^{INSPEC}$	100	0	93.42	6.58	70.12
$C_{OB/SM}^{INSPEC}$	6.58	93.42	0	6.58	59.10

Figure 42: Evaluation criteria for the basic configuration with *Binary* as the estimator. The last column shows the *Success* values for the basic configuration, using *Ind* as the estimator.

Criteria	*Success*	*Alpha*	*Beta*	*Success − Beta*
C_{EX}	100	0	38.23	61.77
C_{AB}	88.95	11.05	6.89	82.06
C_{OB}	84.38	15.62	2.32	82.06
C_{SM}	91.26	8.74	80.64	10.61

Figure 43: Evaluation criteria for the basic configuration using a *hybrid* estimator.

user's queries. We used this framework to evaluate the efficacy of various estimators. We considered different variations of *GlOSS* aimed at reducing *GlOSS*' storage cost, for example.

The experimental results we obtained, although involving only six databases, are encouraging. To illustrate this, consider the case in which the semantics the user is interested in (out of C_{EX}, C_{AB}, C_{OB}, and C_{SM}) is given as a parameter to a *hybrid* estimator for *GlOSS*. This estimator would choose among *Ind*, *Min*, and *Binary* according to the desired criterion. Following our experimental results, such a hybrid estimator should behave as *Ind* if the target semantics of the query is given by C_{AB}, C_{OB}, or C_{SM}, and as *Binary* for C_{EX}. The results for this hybrid estimator and the basic configuration are summarized in Figure 43. This figure shows that the *Success* values for our four criteria range from 84% to 100%.

To see if the results we got are too dependent on the particular query trace we considered, we executed some of the experiments using a trace from another database. The results we obtained are consistent with those that had been produced with the original set of queries.

The storage cost of *GlOSS* is relatively low (see Figure 25). A rough estimate suggested that 22.29 MBytes would be enough to keep all the data needed for the six databases we studied. Given this low space need, *GlOSS* itself can then be replicated to increase its availability.

The approach we took towards solving the text database discovery problem could also deal with the situation in which some information servers would charge for their use. Since we are selecting what databases to search according to a quantitative measure of their "goodness" for a given 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.

8 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.

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.

[CD90] D. Chapman and S. DeFazio. Statistical characteristics of legal document databases. S&PT Technical Report, Mead Data Central, January 1990.

[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.

[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.

[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. TMC199, Thinking Machines Corporation, April 1991.

[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.

[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.

[SB88] Gerard Salton and Chris Buckley. Parallel text search methods. *Communications of the ACM*, 31(2), February 1988.

[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$^+$] Mark A. Sheldon, Andrzej Duda, Ron Weiss, James W. O'Toole, and David K. Gifford. A content routing system for distributed information servers. 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.

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

[TC92] Howard R. Turtle and W. Bruce Croft. Uncertainty in information retrieval systems, September 1992. Presented at an NSF Workshop in Majorca.