Technical Report CUCS-013-04
Computer Science Department
Columbia University

When one Sample is not Enough: Improving Text Database Selection Using Shrinkage

Panagiotis G. Ipeirotis
Computer Science Department
Columbia University
pirot@cs.columbia.edu

Luis Gravano
Computer Science Department
Columbia University
gravano@cs.columbia.edu

ABSTRACT

Database selection is an important step when searching over large numbers of distributed text databases. The database selection task relies on statistical summaries of the database contents, which are not typically exported by databases. Previous research has developed algorithms for constructing an approximate content summary of a text database from a small document sample extracted via querying. Unfortunately, Zipf's law practically guarantees that content summaries built this way for any relatively large database will fail to cover many low-frequency words. Incomplete content summaries might negatively affect the database selection process, especially for short queries with infrequent words. To improve the coverage of approximate content summaries, we build on the observation that topically similar databases tend to have related vocabularies. Therefore, the approximate content summaries of topically related databases can complement each other and increase their coverage. Specifically, we exploit a (given or derived) hierarchical categorization of the databases and adapt the notion of "shrinkage" –a form of smoothing that has been used successfully for document classification– to the content summary construction task. A thorough evaluation over 315 real web databases as well as over TREC data suggests that the shrinkage-based content summaries are substantially more complete than their "unshrunk" counterparts. We also describe how to modify existing database selection algorithms to adaptively decide –at run-time– whether to apply shrinkage for a query. Our experiments, which rely on TREC data sets, queries, and the associated "relevance judgments," show that our shrinkage-based approach significantly improves state-of-the-art database selection algorithms, and also outperforms a recently proposed hierarchical strategy that exploits database classification as well.

1. INTRODUCTION

A large amount of information available on the web is stored in text databases that are "hidden" behind search interfaces. Often the contents of such databases are non-crawlable and are hence ignored by search engines such as Google. As an alternative to querying each database individually, *metasearchers* offer a unified interface for searching over potentially many text databases simultaneously. Thus, a metasearcher (1) selects the best databases to search for a given user query, (2) translates the query into an appropriate format for each database, and (3) obtains the query results from each database and merges them into a unified answer.

The *database selection* step is critical for efficiency, since a large number of databases might be available even for restricted domains. For example, Conrad et al. [6] report on an *operational* application in the legal domain that involves 15,000 *real* text databases. Database selection typically relies on short descriptions of the available databases [13, 23, 30, 33]. The *content summary* of a database usually includes the words that appear in the database and their document frequency, plus simple additional information such as the size of the database. Unfortunately, web databases tend to be "uncooperative" in that they do not export content summaries, which complicates the database selection step. Query-based sampling [2, 17, 24] has emerged as a promising approach for building (approximate) content summaries for uncooperative databases. Techniques following this approach use queries to extract a relatively small document sample from the databases, which is subsequently used to derive content summaries.

Unfortunately, content summaries constructed using document sampling suffer from a sparse-data problem. These summaries tend to include the most frequent words, but generally miss many other words that appear only in relatively few documents:

Example 1. Consider the medical bibliography database PubMed[1]. Query *[hemophilia]* returns 15,158 matches out of the approximately 12 million citations that PubMed hosts. In other words, 0.1% of the PubMed documents contain the word "hemophilia." A document sample of moderate size is then likely to miss this "rare" word, and so would a content summary built from such a document sample. In turn, a metasearcher might then not identify PubMed as a relevant database for query *[hemophilia]*, which is problematic. □

The task of (hierarchical) document classification suffers from a related data sparseness problem when training data

[1]http://www.ncbi.nlm.nih.gov/entrez/

is insufficient. Based on the observation that words follow related distributions over topically related documents, McCallum et al. [22] suggested "sharing" training examples across close topic categories that are organized in a hierarchy. Their *shrinkage* technique compensates for sparse training data for a category by using training examples from more general categories.

Interestingly, we can exploit shrinkage for building approximate database summaries as well: when multiple databases correspond to similar topic categories, they tend to have similar content summaries [17]. Therefore, the content summaries of these databases can mutually complement each other. For example, the content summary of PubMed in Example 1 can be augmented with words from other "Health"-related databases, which may contain the word "hemophilia" in their content summaries.

To apply shrinkage to our problem, each database should be categorized into a topic hierarchy. This categorization might be an existing one (e.g., if the databases are classified in a web directory) or, alternatively, can be derived automatically (e.g., using recently introduced query-probing techniques [14]). As a key contribution of this paper, we introduce the first technique to build (high-quality) database content summaries using shrinkage. As another contribution of this paper, we show how to use shrinkage during database selection in an adaptive, query-specific way. Our extensive experimental evaluation of our techniques is two-fold. First we compare the quality and completeness of the shrinkage-based content summaries against their "unshrunk" counterparts, for the most prominent summary-construction algorithms from the literature. Our experiments involve text databases from the TREC ("Text REtrieval Conference") testbed, as well as 315 real web databases. Second, we evaluate the impact of shrinkage on database selection accuracy, for several state-of-the-art database selection algorithms. Our experiments involve the text databases and queries from the TREC testbed, together with the "relevance judgments" associated with the queries and the database documents. In short, our evaluation shows that the new content summaries are highly complete, and correspondingly help improve the accuracy of database selection substantially. We compare our method against state-of-the-art database selection algorithms, including a recently introduced hierarchical algorithm that also exploits database classification information to compensate for incomplete summaries [17]. Our experiments show a significant improvement in performance, achieved *efficiently* just by exploiting the database classification information and without increasing the document-sample size.

The rest of the paper is organized as follows. Section 2 gives the necessary background. Section 3 explains the use of database classification in conjunction with shrinkage to improve content summary coverage. Section 4 describes an adaptive database selection algorithm that uses the shrunk content summaries. Sections 5 and 6 describe the experimental settings and results. Finally, Sections 7 and 8 describe related work and conclude the paper.

2. BACKGROUND

This section introduces the notation and necessary background for this paper. Section 2.1 briefly summarizes how existing database selection algorithms work, stressing their reliance on database "content summaries." Then, Section 2.2 describes how content summaries of "uncooperative" databases can be approximated via query probing, and identifies opportunities for improving the coverage of such approximate summaries.

D_1, with $\|D_1\|$=51,500	
w	$p(w\|D_1)$
algorithm	$1.4 \cdot 10^{-1}$
blood	$1.9 \cdot 10^{-5}$
hypertension	$3.8 \cdot 10^{-5}$
...	...

D_2, with $\|D_2\|$=25,730	
w	$p(w\|D_2)$
algorithm	$2 \cdot 10^{-4}$
blood	$4.2 \cdot 10^{-1}$
hypertension	$3.2 \cdot 10^{-1}$
...	...

Table 1: A fragment of the content summaries of two databases.

2.1 Database Selection Algorithms

Database selection is a crucial step in the metasearching process. For efficiency, a metasearcher should direct each query only to a relatively small number of databases. We now briefly outline how typical database selection algorithms work and how they depend on database "content summaries" to make decisions.

Definition 1. The *content summary* $S(D)$ of a database D consists of:

- The actual number of documents in D, $|D|$, and
- For each word w, the fraction of D documents that include w, or $p(w|D) = \frac{|d \in D: d \text{ contains word } w|}{|D|}$. $\diamond$

Table 1 shows a small fraction of the content summaries of two hypothetical text databases. The first database, D_1, contains articles about *Computer Science*. According to the associated content summary, this database consists of 51,500 documents, and the word "algorithm" appears in a large fraction of them. In contrast, this word appears in a small fraction of the documents in database D_2, with articles about *Health*.

Given database content summaries, a database selection algorithm estimates the relevance of each database for a given query (e.g., in terms of the number of matches that each database is expected to produce for the query):

Example 2. Consider the query *[blood hypertension]* and the two databases D_1 and D_2 of Table 1. A database selection algorithm may infer that D_2 is a promising database for the query, since each query word appears in a large fraction of the documents in D_2. In contrast, D_1 will probably be deemed not as relevant, since it contains only up to a handful of documents with each query word. $\square$

Database selection algorithms work best when the content summaries are accurate and up to date. The most desirable scenario is when each database exports these content summaries directly and reliably (e.g., using a protocol such as STARTS [12]). Unfortunately, no protocol is widely adopted for web-accessible databases. Hence, other solutions have been proposed to automate the construction of content summaries from databases that cannot or are not willing to export such information. We briefly review such approaches next.

2.2 Query-Based Sampling

Callan and Connell [2] presented an algorithm for building (approximate) content summaries of text databases that do not export any "metadata" about their contents. This algorithm starts by extracting a small document sample from a given database D via single-word queries. The document sample is then treated as a database and its content summary is used to approximate that of D's. A sample of about 300 documents is generally regarded as sufficient to create a "good" summary [2]. Alternative techniques [17, 24] also use query-based sampling with the same goal, but with different querying strategies. In this section, we assume for simplicity a generic (unspecified) document sampling and content summary approximation strategy, and refer to the resulting content summaries as follows:

DEFINITION 2. The *approximate content summary* $\hat{S}(D)$ of a database D consists of:

- An estimate $|\hat{D}|$ of the number of documents in D, and
- For each word w, an estimate $\hat{p}(w|D)$ of $p(w|D)$.

The $\hat{S}(D)$ estimates are computed from a sample of the documents in D. □

Later, in Sections 5 and 6 we describe a variety of specific sampling and content summary construction strategies (e.g., [2, 17]) from the literature.

Unfortunately, all efficient techniques for building content summaries via document sampling suffer from the "sparse data" problem, since many words in any text database tend to occur in relatively few documents. Therefore, any document sample of reasonably small size will necessarily miss many words that occur in the associated database a small number of times. The absence of these words from the content summaries can negatively affect the performance of database selection algorithms for queries that mention such words. To alleviate this sparse-data problem, we exploit the observation that incomplete content summaries of topically related databases can be used to complement each other, as discussed next.

3. IMPROVING CONTENT SUMMARIES

As argued above, content summaries derived from relatively small document samples are generally incomplete. In this section we show how we can exploit database category information to improve the quality of the database summaries. Specifically, Section 3.1 presents an overview of our general approach, which builds on the shrinkage ideas from document classification [22], while Section 3.2 explains the details of our method.

3.1 Overview of our Approach

Section 2.2 briefly reviewed sampling-based techniques for building content summaries from "uncooperative" text databases. As discussed, low-frequency words tend to be absent from those summaries. Additionally, other words might be disproportionately represented in the document samples. One way to alleviate these problems is to increase the sample size. Unfortunately, this solution might be impractical, since it would involve extensive querying of the (remote) databases. Even more importantly, increases in document sample size do not tend to result in comparable improvements in content summary quality [2]. An interesting challenge is then to improve the quality of the approximate content summaries without necessarily increasing the document sample size.

This challenge has a counterpart in the problem of hierarchical document classification. Document classifiers rely

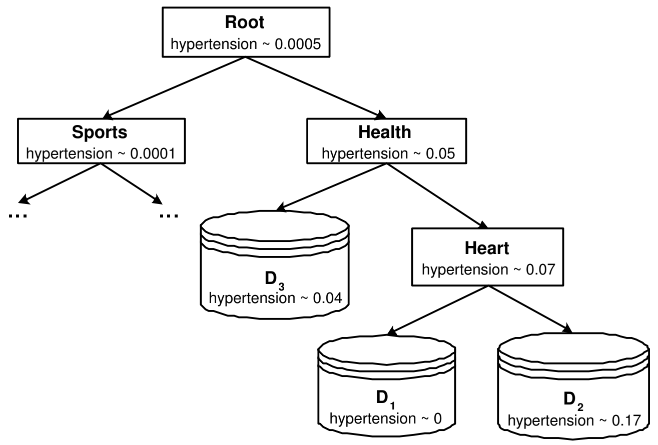


Figure 1: A fraction of a classification hierarchy and content summary statistics for word "hypertension."

on training data to associate words with categories. Often, only limited training data is available, which might lead to poor classifiers. Classifier quality can be increased with more training data, but creating large numbers of training examples might be prohibitively expensive. As a less expensive alternative, McCallum et al. [22] suggested "sharing" training data across related topic categories. Specifically, their *shrinkage* approach compensates for sparse training data for a category by using training examples for more general categories. For example, the training documents for the "Heart" category can be augmented with those from the more general "Health" category. The intuition behind this approach is that the word distribution in "Health" documents is hopefully related to the word distribution in the "Heart" documents.

We can apply the same shrinkage principle to our problem, which requires that databases be categorized into a topic hierarchy. This categorization might be an existing one (e.g., if the databases are classified under Open Directory[2] or InvisibleWeb[3]). Alternatively, databases can be classified automatically using, for example, the query-probing technique in [14]. Regardless of how databases are categorized, we can exploit this categorization to improve content summary coverage. The key intuition behind the use of shrinkage in this context is that databases under similar topics tend to have related content summaries, as observed in [17]. Hence, we can use the approximate content summaries for similarly classified databases to complement each other, as illustrated in the following example.

EXAMPLE 3. Figure 1 shows a fraction of a hierarchical classification scheme with two databases D_1 and D_2 classified under the category "Heart," and one database D_3 classified under the (higher-level) category "Health." Assume that the approximate content summary of D_1 does not contain the word "*hypertension*," but that this word appears in many documents in D_1. ("*Hypertension*" might not have appeared in any of the documents sampled to build $\hat{S}(D_1)$.) In contrast, "*hypertension*" appears in a relatively large fraction of D_2 documents as reported in the content summary

[2]http://www.dmoz.org
[3]http://www.invisibleweb.com

of D_2, a database also classified under the "Heart" category. Then, by "shrinking" $\hat{p}(\textit{hypertension}|D_1)$ towards the value of $\hat{p}(\textit{hypertension}|D_2)$, we can capture more closely the actual (and unknown) value of $p(\textit{hypertension}|D_1)$. The new, "shrunk" value is in effect exploiting the documents sampled from both D_1 and D_2. $\square$

We expect databases under the same category to have similar content summaries. Furthermore, even databases classified under relatively general categories can help to improve the approximate content summary of a more specific database. Consider database D_3, classified under "Health" in the example in Figure 1. $\hat{S}(D_3)$ can help complement the content summary approximation of databases D_1 and D_2, which are classified under a subcategory of "Health," namely "Heart." Database D_3, however, is a more general database that contains documents in topics other than heart-related. Hence, the influence of $\hat{S}(D_3)$ on $\hat{S}(D_1)$ should perhaps be smaller than that of, say, $\hat{S}(D_2)$. In general, and just as for document classification [22], each category level might be assigned a different "weight" during shrinkage. We discuss this and other specific aspects of our technique next.

3.2 Using Shrinkage over a Topic Hierarchy

We now define more formally how we can use shrinkage for content summary construction. For this, we first extend the notion of content summaries to the categories of a classification scheme.

Creating Category Content Summaries

The content summary of a category C summarizes the contents of the databases classified under C[4].

Definition 3. Consider a category C and the set $db(C)$ of databases classified under C. The *approximate content summary* $\hat{S}(C)$ of category C contains, for each word w, an estimate $\hat{p}(w|C)$ of $p(w|C)$, where $p(w|C)$ is the probability that a randomly selected document from a database in $db(C)$ contains the word w. The $\hat{p}(w|C)$ estimates in $\hat{S}(C)$ are derived from the content summaries of databases in $db(C)$ as[5]:

$$\hat{p}(w|C) = \frac{\sum_{D \in db(C)} \hat{p}(w|D) \cdot |\hat{D}|}{\sum_{D \in db(C)} |\hat{D}|} \quad (1)$$

Creating Shrunk Content Summaries

Section 3.1 argued that mixing information from content summaries of topically related databases may lead to more complete approximate content summaries. We now formally describe how to use shrinkage for this purpose. In essence, we create a new content summary for each database D by "shrinking" the approximate content summary of D, $\hat{S}(D)$, so that it is "closer" to the content summaries $S(C_i)$ of each category C_i under which D is classified.

Definition 4. Consider a database D classified under categories $C_1, \ldots, C_m$ of a hierarchical classification scheme, with $C_i = \textit{Parent}(C_{i+1})$ for $i = 1, \ldots, m-1$. Let C_0 be a dummy category whose content summary $\hat{S}(C_0)$ contains the same estimate $\hat{p}(w|C_0)$ for every word w. Then, the *shrunk content summary* $\hat{\mathcal{R}}(D)$ of database D consists of:

- An estimate $|\hat{D}|$ of the number of documents in D, and
- For each word w, a shrinkage-based estimate $\hat{p}_{\mathcal{R}}(w|D)$ of $p(w|D)$, defined as:

$$\hat{p}_{\mathcal{R}}(w|D) = \lambda_{m+1} \cdot \hat{p}(w|D) + \sum_{i=0}^{m} \lambda_i \cdot \hat{p}(w|C_i) \quad (2)$$

for a choice of λ_i values such that $\sum_{i=0}^{m+1} \lambda_i = 1$ (see below).$\square$

As described so far, the $\hat{p}(w|C_i)$ values in the $\hat{S}(C_i)$ content summaries are not independent of each other: since $C_i = \textit{Parent}(C_{i+1})$, all the databases under C_{i+1} are also used to compute $\hat{S}(C_i)$ (Definition 3). To avoid this overlap, before estimating $\hat{\mathcal{R}}(D)$ we subtract from $\hat{S}(C_i)$ all the data used to construct $\hat{S}(C_{i+1})$. Also note that a simple version of Equation 2 is used for database selection based on language models [28]. Language model database selection "smoothes" the $\hat{p}(w|D)$ probabilities with the probability $\hat{p}(w|G)$ for a "global" category G. Our technique extends this principle and does multilevel smoothing of $\hat{p}(w|D)$, using the hierarchical classification of D. We now describe how to compute the λ_i weights used in Equation 2.

Calculating the Category Mixture Weights

We define the λ_i mixture weights from Equation 2 so as to make the shrunk content summaries $\hat{\mathcal{R}}(D)$ for each database D as "similar" as possible to *both* the starting summary $\hat{S}(D)$ and the summary $\hat{S}(C_i)$ of each category C_i under which D is classified. Specifically, we use expectation maximization (EM) [22] to calculate the λ_i weights, using the algorithm in Figure 2. (This is a simple version of the EM algorithm from [8].) Note that the λ_{m+1} weight associated with a database (as opposed to with the categories under which it is classified) is usually highest among the λ_i's and so the word-distribution statistics for the database are not "eclipsed" by the category statistics.

The "Expectation" step calculates the "likelihood" that content summary $\hat{\mathcal{R}}(D)$ corresponds to each category. The "Maximization" step weights the λ_is to maximize the total likelihood across all categories. The result of the algorithm is the shrunk content summary $\hat{\mathcal{R}}(D)$, which incorporates information from multiple content summaries and is thus hopefully closer to the complete (and unknown) content summary $S(D)$ of database D.

For illustration purposes, Table 2 reports the computed mixture weights for two databases that we used in our experiments. As we can see, in both cases the original database content summary and that of the most specific category for the database receive the highest weights (0.421 and 0.414, respectively, for the AIDS.org database, and 0.411 and 0.297, respectively, for the American Economics Association database). However, higher-level categories also receive non-negligible weights.

Finally, note that the λ_i weights are computed off-line for each database when the sampling-based database content summaries are created. This computation does not involve any overhead at query-processing time.

[4]Category content summaries were introduced in [17] to define a hierarchical database selection algorithm.

[5]An alternative is to define $\hat{p}(w|C) = \frac{\sum_{D \in db(C)} \hat{p}(w|D)}{|db(C)|}$, which "weights" each database equally, regardless of its size. We implemented this alternative and obtained results that were virtually identical to those for Equation 1.

```
**Initialization step:**
Set each $\lambda_i$ to any normalized, non-zero value such that $\sum_{i=0}^{m+1} \lambda_i = 1$. (For example, $\lambda_i = \frac{1}{m+2}$, for every $i = 0, \ldots, m+1$.)

Create the shrunk content summary $\hat{\mathcal{R}}(D)$, using the current $\lambda_i$ values (Definition 4).

**Expectation step:**
Calculate the "similarity" $\beta_i$ of each category $C_i$ with $\hat{\mathcal{R}}(D)$, for $i = 0, \ldots, m$:

    $$\beta_i = \sum_{w \in \hat{S}(D)} \frac{\lambda_i \cdot \hat{p}(w|C_i)}{\hat{p}_{\mathcal{R}}(w|D)}$$

Calculate the "similarity" $\beta_{m+1}$ of database $D$ with $\hat{\mathcal{R}}(D)$:

    $$\beta_{m+1} = \sum_{w \in \hat{S}(D)} \frac{\lambda_{m+1} \cdot \hat{p}(w|D)}{\hat{p}_{\mathcal{R}}(w|D)}$$

**Maximization step:**
Maximize the total "similarity" of $\hat{\mathcal{R}}(D)$ with the category content summaries $\hat{S}(C_i)$ and with $\hat{S}(D)$, by redefining each $\lambda_i$ as follows:

    $$\lambda_i = \frac{\beta_i}{\sum_{j=0}^{m+1} \beta_j}$$

Modify the shrunk content summary $\hat{\mathcal{R}}(D)$ using the current $\lambda_i$ values.

**Termination check:**
When calculation of the $\lambda_i$ weights converges (within a small $\epsilon$) return the "current" shrunk content summary $\hat{\mathcal{R}}(D)$. Otherwise, go to the "Expectation" step.
```

Figure 2: Using expectation maximization to determine the λ_i mixture weights for the shrunk content summary of a database D.

Database	*Category*	λ
AIDS.org	Uniform	0.075
	Root	0.026
	Health	0.061
	Diseases	0.003
	AIDS	0.414
	AIDS.org	0.421
American Economics Association	Uniform	0.041
	Root	0.041
	Science	0.055
	Social Sciences	0.155
	Economics	0.297
	American Economics Association	0.411

Table 2: The category mixture weights λ_i for two databases.

```
**Input:** Query $q = [w_1, \ldots, w_n]$; databases $D_1, \ldots, D_m$

**Content Summary Selection step:**
For each $D_i$:
    For every possible choice of values for $d_1, \ldots, d_n$ (see text):
        Compute the probability $P$ that $w_k$ appears in
        exactly $d_k$ documents in $D_i$, for $k = 1, \ldots, n$.

        Compute the score $s(q, D_i)$ assuming that $w_k$ appears
        in exactly $d_k$ documents in $D_i$, for $k = 1, \ldots, n$.

    If the standard deviation of the score distribution
        across $d_k$ values is larger than its mean
    then $A(D_i) = \hat{\mathcal{R}}(D_i)$ // use "shrunk" content summary
    else $A(D_i) = \hat{S}(D_i)$ // use "unshrunk" content summary

**Scoring step:**
For each $D_i$:
    Compute $s(q, D_i)$ using the $A(D_i)$ content summary,
    as selected in the "Content Summary Selection" step.

**Ranking step:**
Rank the databases by decreasing score $s(q, D_i)$.
```

Figure 3: Using shrinkage adaptively for database selection.

4. IMPROVING DATABASE SELECTION

As we discussed in the Introduction, database selection is a critical component of efficient metasearchers. Given a query q, a database selection algorithm assigns a score $s(q, D)$ to each database D, so that the metasearcher can evaluate (a suitably modified version of) q over just the databases with the highest scores. We now show how shrinkage can be used together with state-of-the-art database selection algorithms to substantially improve the accuracy of the selection process.

Section 3 introduced a shrinkage-based strategy to complement the incomplete content summary of a database with the summaries of topically related databases. In principle, existing database selection algorithms could proceed without modification and use the shrunk summaries to assign scores for *all* queries and databases. However, sometimes shrinkage might not be beneficial and should not be used. Intuitively, shrinkage should be used to determine the score $s(q, D)$ for a query q and a database D only if the "uncertainty" associated with this score would otherwise be large.

The uncertainty associated with an $s(q, D)$ score depends on a number of sample-, database-, and query-related factors. An important factor is the size of the document sample relative to that of database D. If an approximate summary $\hat{S}(D)$ was derived from a sample that included most of the documents in D, then this summary is already "sufficiently complete." (For example, this situation might arise if D is a small database.) In this case, shrinkage is not necessary and might actually be undesirable, since it might introduce spurious words into the content summary from topically related (but not identical) databases. Another factor is the frequency of the query words in the sample used to determine $\hat{S}(D)$. If, say, every word in a query appears in close to all the sample documents, and the sample is representative of the entire database contents, then there is little uncertainty on the distribution of the words over the database at large. This is also the case for the analogous situation in which every query word appears in close to no sample documents. In either case, shrinkage would provide limited benefit and should then be avoided. In contrast, for other query-word distribution scenarios the approximate content summary might not be sufficient to reliably establish the query-specific score for the database, in which case shrinkage is desirable.

More formally, consider a query $q = [w_1, \ldots, w_n]$ with n words $w_1, \ldots, w_n$, a database D, and an approximate content summary for D, $\hat{S}(D)$, derived from a random sample S of D. Furthermore, suppose that word w_k appears in exactly s_k documents in the sample S. For every possible combination of values $d_1, \ldots, d_n$ (see below), we compute:

- The probability P that w_k appears in exactly d_k documents in D, for $k = 1, \ldots, n$:

$$P = \prod_{k=1}^{n} \frac{d_k^{\gamma} \left(\frac{d_k}{|D|}\right)^{s_k} \left(1 - \frac{d_k}{|D|}\right)^{|S|-s_k}}{\sum_{i=0}^{|D|} i^{\gamma} \cdot \left(\frac{i}{|D|}\right)^{s_k} \left(1 - \frac{i}{|D|}\right)^{|S|-s_k}} \quad (3)$$

  where γ is a database-specific constant. (For details, see Appendix B.)

- The score $s(q, D)$ that the database selection algorithm of choice would assign to D if $p(w_k|D) = \frac{d_k}{|D|}$, for $k = 1, \ldots, n$.

So for each possible combination of values $d_1, \ldots, d_n$, we compute both the probability of the value combination and the score that the database selection algorithm would assign to D for this document frequency combination. Then, we can approximate the "uncertainty" behind the $s(q, D)$ score by examining the mean and variance of the database scores over the different $d_1, \ldots, d_n$ values. This computation can be performed efficiently for a generic database selection algorithm: given the sample frequencies $s_1, \ldots, s_n$, a large number of possible $d_1, \ldots, d_n$ values have virtually zero probability of occurring, so we can ignore them. Additionally, mean and variance converge fast, even after examining only a small number of $d_1, \ldots, d_n$ combinations. Specifically, we examine random $d_1, \ldots, d_n$ combinations and periodically calculate the mean and variance of the score distribution. Usually, after examining just a few hundred random $d_1, \ldots, d_n$ combinations, mean and variance converge to a stable value. This computation can be even faster for a large class of database selection algorithms that assume independence between the query words (e.g, [13, 4, 31]). For these algorithms, we can calculate the variance for each query word separately, and then combine them into the final score variance.

Figure 3 summarizes the discussion above, and shows how we can adaptively use shrinkage with an existing database selection algorithm. Specifically, the algorithm takes as input a query q and a set of databases $D_1, \ldots, D_m$. The "Content Summary Selection" step decides whether to use shrinkage for each database D_i, as discussed above. If the distribution of possible scores has high variance, then $\hat{S}(D_i)$ is considered unreliable and the shrunk content summary $\hat{\mathcal{R}}(D_i)$ is used instead. Otherwise, shrinkage is not applied. Then, the "Scoring" step computes the score $s(q, D_i)$ for each database D_i, using the content summary chosen for D_i in the "Content Summary Selection" step. Finally, the "Ranking" step orders all databases by their final score for the query. The metasearcher then uses this rank to decide what databases to search for the query.

5. EXPERIMENTAL SETTING

We now describe the data (Section 5.1), strategies for computing content summaries (Section 5.2), and database selection algorithms (Section 5.3) that we use in our experiments.

5.1 Data

The shrinkage technique described above relies on a hierarchical categorization scheme. For our experiments, we use the subset of the *Open Directory Project (ODP)* topic hierarchy from [14]. This portion of ODP consists of 72 nodes organized in a 4-level hierarchy. To evaluate the algorithms described in this paper, we use three data sets in conjunction with the hierarchical classification scheme.

TREC4: This is a set of 100 databases created using TREC-4 documents [15] and separated into disjoint databases via clustering using the K-means algorithm, as specified in [31]. By construction, the documents in each database are on roughly the same topic.

TREC6: This is a set of 100 databases created using TREC-6 documents [29] and separated into disjoint databases using the same methodology as for *TREC4*.

Web: This set contains the top-5 *real web databases* from each of the 54 leaf categories of the hierarchy, as ranked in the Google Directory[6], plus other arbitrarily selected web sites, for a total of 315 databases. The sizes of these databases range from 100 to about 376,000 documents. Table 3 lists four example databases. We used the *GNU Foundation's wget* crawler to download the HTML contents of each site, and we kept only the text from each file by stripping the HTML tags using the "*lynx –dump*" command.

For indexing and searching the files in all three data sets we used *Jakarta Lucene*,[7] an open-source full-text search engine.

5.2 Content Summary Construction Algorithms

Sampling Algorithms: We use two different sampling algorithms from the literature for creating the approximate content summaries $\hat{S}(D)$ of each database D:

- **Query-Based Sampling (QBS)**, as presented in [2]: We send random, single-word queries to a given database until at least one document is retrieved. Then, we continue to query the database using the words in the retrieved documents. Each query retrieves at most four previously unseen documents. Sampling stops when the document sample contains 300 documents. In our experiments, sampling also stops when 500 consecutive queries retrieve no new documents. To minimize the effect of randomness, we run each experiment over five QBS document samples for each database and report average results.

- **Focused Probing (FPS)**, as presented in [17]: Instead of sending (pseudo-) randomly picked words as queries, this algorithm derives queries from a classifier learned over a topic hierarchy. Thus, queries are associated with specific topics. For example, a query *[breast cancer]* is associated with the category "Health." We retrieve the top four previously unseen documents for each query and, at the same time, keep track of the number of matches generated by each query. When the queries related to a category (e.g., "Health") generate a large number of matches, probing continues for its subcategories (e.g., "Diseases" and "Fitness"). The output of the algorithm is both an approximate content summary and the classification of the database in a hierarchical classification scheme.

Frequency and Database Size Estimation: As described above, *QBS* and *FPS* define $\hat{p}(w|D)$ for a word w and database D as the fraction of documents containing *w in the sample* extracted from D. To better approximate the fraction of documents containing *w in the database D*, we use

[6]http://directory.google.com/

[7]http://jakarta.apache.org/lucene/

Database	*URL*	*Documents*	*Classification*
Bartleby	http://www.bartleby.com/	375,734	Root→ Arts→ Literature→ Texts
Java @ Sun	http://java.sun.com/	78,870	Root→ Computers→ Programming→ Java
Math Forum @ Drexel	http://mathforum.org/	29,602	Root→ Science→ Mathematics
Union of European Football Associations	http://www.uefa.com/	28,329	Root→ Sports→ Soccer

Table 3: Some of the real web databases in the *Web* data set.

the frequency estimation technique presented in [17] and the "sample-resample" method from [27] to estimate the size of D. Both techniques exploit the number of matches that the queries generate at the database. For our experiments, we modified the technique in [17] to produce better frequency estimates. (See Appendix A for details.) Our experiments consider the *QBS* and *FPS* summaries both with and without this frequency estimation.

Database Classification: To apply our shrinkage-based technique we need to classify each database into the 72-node hierarchy. Unfortunately, such classification is not available for TREC data, so for the *TREC4* and *TREC6* data sets we resort to the probe-based automatic database classification technique in [14][8]. A manual inspection of the classification results confirmed that they are generally accurate. For example, the *TREC4* database *all-83*, with articles about AIDS, was correctly classified under the "Root→ Health→ Diseases→ AIDS" category. Interestingly, in the (rare) case in which databases were not classified correctly, "similar" databases were still classified into the same (incorrect) category. For example, *all-14*, *all-21*, and *all-44* are about middle-eastern politics and were classified under the "Root→ Science→ Social Sciences→ History" category.

Unlike for *TREC4* and *TREC6*, for which no "external" classification of the databases is available, for the *Web* databases we do not have to rely on query probing for classification. Instead we can use the categories assigned to the databases in the Google Directory. For *QBS*, the classification of each database in our data set was indeed derived from the Google Directory. For *FPS*, we can either use the (correct) Google Directory database classification, as for *QBS*, or rely on the automatically computed database classification that this technique derives during document sampling. We tried both choices and found only small differences in the experimental results. Therefore, for conciseness, we only report the *FPS* results for the automatically derived database classification [14].

5.3 Database Selection Algorithms

In addition to measuring content summary "quality," our evaluation studies the effect of the content summary completeness on database selection accuracy. Our technique of Figure 3 is built on top of an underlying "base" database selection algorithm to assign scores to databases. We consider three well-known such algorithms from the literature[9].

- bGlOSS, as described in [13]. Databases are ranked for a query q by decreasing score $s(q,D)=|D|\cdot\prod_{w\in q}\hat{p}(w|D)$.

- CORI, as described in [10]. Databases are ranked for a query q by decreasing score $s(q,D)=\sum_{w\in q}\frac{0.4+0.6\cdot T\cdot I}{|q|}$, where $T=(\hat{p}(w|D)\cdot|D|)/(\hat{p}(w|D)\cdot|D|+50+150\cdot\frac{cw(D)}{mcw})$, $I=\log\left(\frac{m+0.5}{cf(w)}\right)/\log(m+1.0)$, $cf(w)$ is the number of databases containing w, m is the number of databases being ranked, $cw(D)$ is the number of words in D, and mcw is the mean cw among the databases being ranked. One potential problem with the use of CORI in conjunction with shrinkage is that virtually every word has $cf(w)$ equal to the number of databases in the data set: every word appears with non-zero probability in every shrunk content summary. Therefore, when we calculate $cf(w)$ for a word w in our CORI experiments, we consider w as "present" in a database D only when $round(|D|\cdot\hat{p}_{\mathcal{R}}(w|D))\geq 1$.

- Language Modelling (LM), as described in [28]. Databases are ranked for a query q by decreasing score $s(q,D)=\prod_{w\in q}(\lambda\cdot\hat{p}(w|D)+(1-\lambda)\cdot\hat{p}(w|G))$. The LM algorithm is equivalent to the KL-based database selection method described in [31]. For LM, $p(w|D)$ is defined differently than in Definition 1. Specifically, $p(w|D)=\frac{tf(w,D)}{\sum_i tf(w_i,D)}$, where $tf(w,D)$ is the total number of occurrences of w in D. The algorithms described in Section 3 can be easily adapted to reflect this difference, by substituting this definition of $p(w|D)$ for that in Definition 1. LM smoothes the $\hat{p}(w|D)$ probability with the probability $\hat{p}(w|G)$ for a "global" category G. (Our shrinkage technique extends this principle and does multilevel smoothing of $\hat{p}(w|D)$, using the hierarchical classification of D.) In our experiments, we derive the probabilities $\hat{p}(w|G)$ from the "Root" category summary and we use $\lambda=0.5$ as suggested in [28].

We experimentally evaluate the three database selection algorithms above with three variations: (1) using "unshrunk" (incomplete) database content summaries extracted via *QBS* or *FPS*; (2) using shrinkage when appropriate (as discussed in Figure 3), again over database content summaries extracted via *QBS* or *FPS*; and (3) using "unshrunk" database content summaries (extracted via *QBS* or *FPS*) that are "aggregated" for each category as described in [17]. This last strategy attempts to compensate for content summary incompleteness by exploiting a categorization of databases, and hence is close in spirit to our shrinkage-based technique. However, the technique in [17] does not modify the database content summaries per se, but rather aggregates them together into category content summaries. This allows database selection to proceed hierarchically by exploring the categories that best match a query according to a "base" algorithm such as CORI, bGlOSS, or LM [17].

6. EXPERIMENTAL RESULTS

We now report experimental results on the quality of the content summaries generated by the different techniques (Section 6.1), as well as on the associated database selection accuracy (Section 6.2).

[8]We adapted the technique in [14] slightly so that each database is classified under exactly one category.

[9]Experiments using shrinkage together with ReDDE [27], a promising, recently proposed database selection algorithm, remain as interesting future work.

6.1 Content Summary Quality

Consider a database D and a content summary $A(D)$ computed using any of the techniques in Section 5.2. We now evaluate the quality of $A(D)$ in terms of how well it approximates the "perfect" content summary $S(D)$ –determined by examining every document in D. In the following definitions, W_A is the set of words that appear in $A(D)$, while W_S is the (complete) set of words that appear in $S(D)$.

Recall: An important property of content summaries is their coverage of the actual database vocabulary. The *weighted recall (wr)* of $A(D)$ with respect to $S(D)$ is defined as $wr = \frac{\sum_{w \in W_A \cap W_S} p(w|D)}{\sum_{w \in W_S} p(w|D)}$, which corresponds to the *ctf ratio* in [2]. This metric gives higher weight to more frequent words. The shrunk content summaries include (with a non-zero probability) every word in any content summary. Most words in any given content summary, however, tend to exhibit a very low probability. Therefore, to not inflate artificially the recall results (and conversely, to not hurt artificially the precision results), we drop from the shrunk content summaries every word w with $round(|D| \cdot \hat{p}_{\mathcal{R}}(w|D)) < 1$ during evaluation. Intuitively, we drop from the content summary all the words that are estimated to appear in less than one document.

Table 4 shows the weighted recall wr for different content summary construction algorithms. Most methods exhibit high weighted recall, which shows that document sampling techniques identify the most frequent words in the database. Not surprisingly, shrinkage increases the (already high) wr values and all shrinkage-based methods have close-to-perfect wr; this improvement is statistically significant in all cases: a paired t-test [21] showed significance at the 0.01% level. The improvement for the *Web* set is higher compared to that for the *TREC4* and *TREC6* data sets: the *Web* set contains larger databases, and the approximate content summaries are less complete than the respective approximate content summaries of *TREC4* and *TREC6*. Our shrinkage technique becomes increasingly more useful for larger databases.

To understand whether low-frequency words are present in the approximate and shrunk content summaries, we resort to the *unweighted recall (ur)* metric, defined as $ur = \frac{|W_A \cap W_S|}{|W_S|}$. The ur metric is the fraction of words in a database that are present in a content summary. Table 5 shows that the shrunk content summaries have higher unweighted recall as well.

Finally, recall is higher when shrinkage is used in conjunction with the frequency estimation technique, reviewed in Section 5.2. This behavior is to be expected: when frequency estimation is enabled, the words introduced by shrinkage are close to their real frequencies, and are used in precision and recall calculations. When frequency estimation is not used, the estimated frequencies of the same words are often below 0.5, and are therefore not used in precision and recall calculations.

Precision: A database content summary constructed using a document sample contains only words that appear in the database. In contrast, the shrunk content summaries may include words not in the corresponding databases. To measure the extent to which "spurious" words are added –with high weight– by shrinkage in the content summary, we use the *weighted precision (wp)* of $A(D)$ with respect to $S(D)$, $wp = \frac{\sum_{w \in W_A \cap W_S} \hat{p}(w|D)}{\sum_{w \in W_A} \hat{p}(w|D)}$. Table 6 shows that shrinkage decreases weighted precision by just 0.8% to 5.7%.

We also report the *unweighted precision (up)* metric, defined as $up = \frac{|W_A \cap W_S|}{|W_A|}$. This metric reveals how many words introduced in a content summary do not appear in the complete content summary (or, equivalently, in the underlying database). Table 7 reports the results for the *up* metric, which show that the shrinkage-based techniques have unweighted precision that is usually above 90% and always above 84%.

Data Set	*Sampling Method*	*Freq. Est.*	*Shrinkage Yes*	*Shrinkage No*
Web	QBS	No	**0.962**	0.875
	QBS	Yes	**0.976**	0.875
	FPS	No	**0.989**	0.887
	FPS	Yes	**0.993**	0.887
TREC4	QBS	No	**0.937**	0.918
	QBS	Yes	**0.959**	0.918
	FPS	No	**0.980**	0.972
	FPS	Yes	**0.983**	0.972
TREC6	QBS	No	**0.959**	0.937
	QBS	Yes	**0.985**	0.937
	FPS	No	**0.979**	0.975
	FPS	Yes	**0.982**	0.975

Table 4: Weighted recall wr

Data Set	*Sampling Method*	*Freq. Est.*	*Shrinkage Yes*	*Shrinkage No*
Web	QBS	No	**0.438**	0.424
	QBS	Yes	**0.489**	0.424
	FPS	No	**0.681**	0.520
	FPS	Yes	**0.711**	0.520
TREC4	QBS	No	**0.402**	0.347
	QBS	Yes	**0.542**	0.347
	FPS	No	**0.678**	0.599
	FPS	Yes	**0.714**	0.599
TREC6	QBS	No	**0.549**	0.475
	QBS	Yes	**0.708**	0.475
	FPS	No	**0.731**	0.662
	FPS	Yes	**0.784**	0.662

Table 5: Unweighted recall ur

Word-Ranking Correlation: Precision and recall measure the accuracy and completeness of $A(D)$ based *only* on the presence (or absence) of words in the content summaries. However, these metrics do not capture the relative "rank" of words in the content summary according to their associated probabilities. For this, we rely on the *Spearman Rank Correlation Coefficient*, which is also used in [2] to evaluate content summary quality. When two rankings are identical, *SRCC*=1; when they are uncorrelated, *SRCC*=0; and when they are in reverse order, *SRCC*=-1.

Table 8 shows that *SRCC* is higher for the shrunk content summaries. In general, *SRCC* is better for the shrunk than for the unshrunk content summaries: not only do the shrunk content summaries have better vocabulary coverage, as the recall figures show, but also the newly added words tend to be ranked properly.

Word-Frequency Accuracy: The *SRCC* metric examines only the word ranking, otherwise ignoring the values of the associated estimates. The *KL-divergence* compares the "similarity" of the $A(D)$ estimates against the real values in $S(D)$: $KL = \sum_{w \in W_A \cap W_S} p(w|D) \cdot \log \frac{p(w|D)}{\hat{p}(w|D)}$, where $p(w|D)$ is defined as for the LM algorithm (see Section 5.3). The KL metric takes values from 0 to infinity, with 0 indicating that the two content summaries being compared are equal.

Table 9 shows that shrinkage helps decrease large KL values. (Recall that lower KL values indicate higher quality summaries.) This is a characteristic of shrinkage [16]: all summaries are shrunk towards some "common" content sum-

Data Set	*Sampling Method*	*Freq. Est.*	*Shrinkage* Yes	No
Web	QBS	No	0.981	**1.000**
	QBS	Yes	0.973	**1.000**
	FPS	No	0.987	**1.000**
	FPS	Yes	0.947	**1.000**
TREC4	QBS	No	0.992	**1.000**
	QBS	Yes	0.978	**1.000**
	FPS	No	0.987	**1.000**
	FPS	Yes	0.984	**1.000**
TREC6	QBS	No	0.978	**1.000**
	QBS	Yes	0.943	**1.000**
	FPS	No	0.976	**1.000**
	FPS	Yes	0.958	**1.000**

Table 6: Weighted precision *wp*

Data Set	*Sampling Method*	*Freq. Est.*	*Shrinkage* Yes	No
Web	QBS	No	0.954	**1.000**
	QBS	Yes	0.942	**1.000**
	FPS	No	0.923	**1.000**
	FPS	Yes	0.909	**1.000**
TREC4	QBS	No	0.965	**1.000**
	QBS	Yes	0.955	**1.000**
	FPS	No	0.901	**1.000**
	FPS	Yes	0.856	**1.000**
TREC6	QBS	No	0.936	**1.000**
	QBS	Yes	0.847	**1.000**
	FPS	No	0.894	**1.000**
	FPS	Yes	0.850	**1.000**

Table 7: Unweighted precision *up*

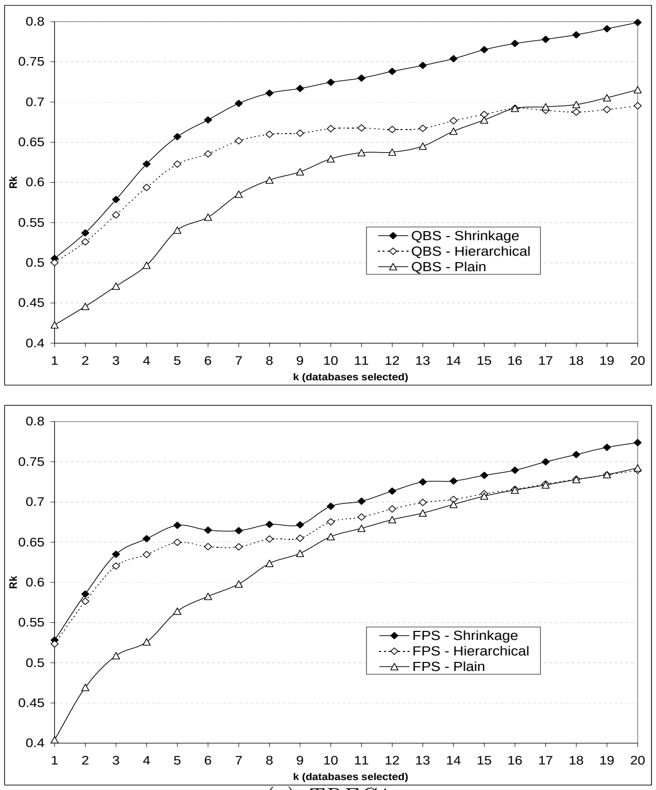
mary, which has an "average" distance from all the summaries. This effectively reduces the variance of the estimations and leads to reduced estimation "risk." However, shrinkage (moderately) hurts content-summary accuracy in terms of the KL metric in cases where KL is already low for the unshrunk summaries. This supports our rationale behind our adaptive database selection algorithm of Section 4. Our algorithm attempts to identify the cases where shrinkage is likely to help general database selection accuracy and avoids applying shrinkage in other cases.

Evaluation Conclusions: The general conclusion from our experiments on content summary quality is that shrinkage drastically improves content summary recall, at the expense of precision. The high *weighted* precision of the shrinkage-based summaries suggests that the spurious words introduced by shrinkage appear with low weight in the summaries, which should reduce any potential negative impact on database selection. Next, we present experimental evidence that the loss in precision ultimately does not hurt, since shrinkage improves overall database selection accuracy.

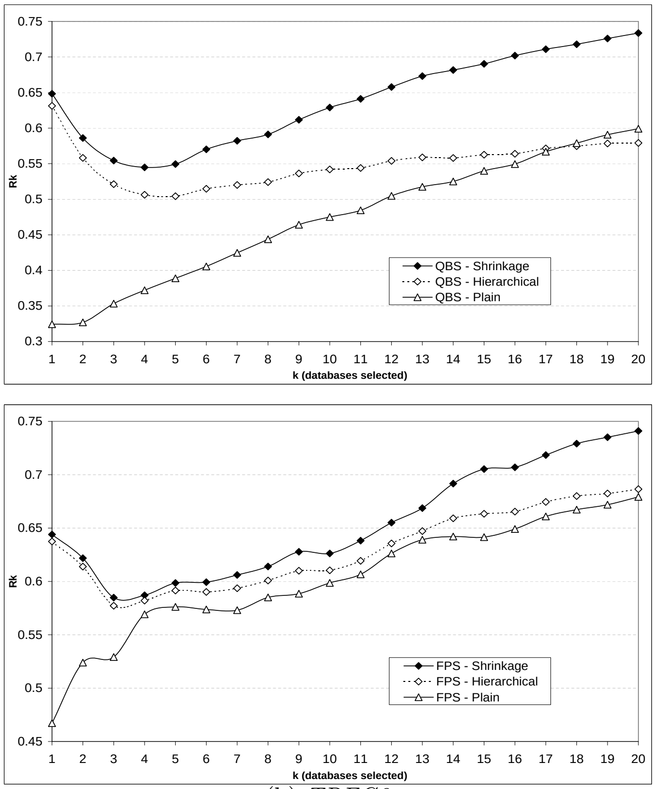
6.2 Database Selection Accuracy

The evaluation presented so far focused on the content summaries. We now turn to the central question of how the quality of the (approximate) content summaries affects the output of the database selection algorithms.

Consider a ranking of the databases $\vec{D} = D_1, \ldots, D_m$ according to the scores produced by a database selection algorithm for some query q. To measure the "goodness" or general "quality" of such a rank, we follow an evaluation methodology that is prevalent in the information retrieval community, and consider the number of documents in each database that are *relevant* to q, as determined by a human judge [25]. Intuitively, a good rank for a query includes –at the top– those databases with the largest number of relevant



(a) *TREC4*



(b) *TREC6*

Figure 4: The R_k ratio for CORI over the *TREC4* and *TREC6* data sets.

Data Set	*Sampling Method*	*Freq. Est.*	*Shrinkage* Yes	*Shrinkage* No
Web	QBS	No	**0.904**	0.812
	QBS	Yes	**0.904**	0.812
	FPS	No	**0.917**	0.813
	FPS	Yes	**0.917**	0.813
TREC4	QBS	No	**0.981**	0.833
	QBS	Yes	**0.981**	0.833
	FPS	No	**0.943**	0.884
	FPS	Yes	**0.943**	0.884
TREC6	QBS	No	**0.961**	0.865
	QBS	Yes	**0.961**	0.865
	FPS	No	**0.937**	0.905
	FPS	Yes	**0.937**	0.905

Table 8: Spearman Correlation Coefficient *SRCC*

Data Set	*Sampling Method*	*Freq. Est.*	*Shrinkage* Yes	*Shrinkage* No
Web	QBS	No	**0.361**	0.531
	QBS	Yes	**0.382**	0.472
	FPS	No	0.298	**0.254**
	FPS	Yes	0.281	**0.224**
TREC4	QBS	No	**0.296**	0.300
	QBS	Yes	**0.175**	0.180
	FPS	No	0.253	**0.203**
	FPS	Yes	0.193	**0.118**
TREC6	QBS	No	**0.305**	0.352
	QBS	Yes	**0.287**	0.354
	FPS	No	0.223	**0.193**
	FPS	Yes	0.301	**0.126**

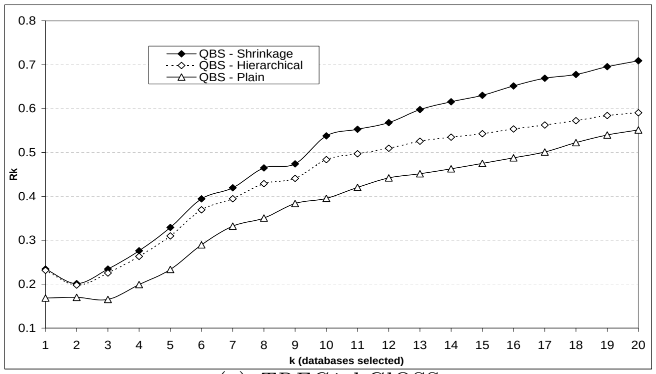
Table 9: KL-divergence

documents for the query.

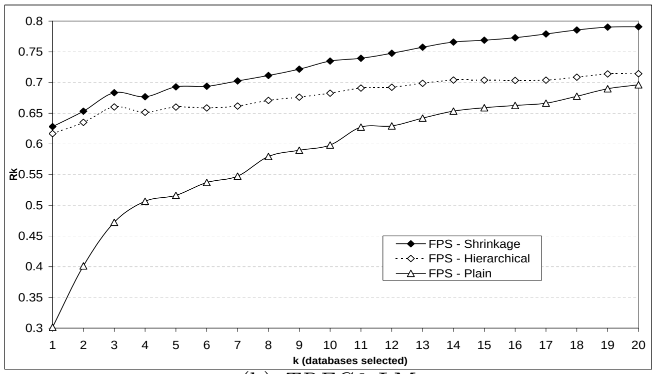
If $r(q, D_i)$ denotes the number of D_i documents that are relevant to query q, then $A(q, \vec{D}, k) = \sum_{i=1}^{k} r(q, D_i)$ measures the total number of relevant documents among the top-k databases in $\vec{D}$. To normalize this measure, we consider a hypothetical, "perfect" database rank $\vec{D}_H = D_{h_1}, \ldots, D_{h_m}$ in which databases are sorted by their $r(q, D_{h_i})$ value. (This is of course unknown to the database selection algorithm.) Then we define the R_k metric for a query and database rank $\vec{D}$ as $R_k = \frac{A(q,\vec{D},k)}{A(q,\vec{D}_H,k)}$ [13]. A "perfect" ordering of k databases for a query yields $R_k = 1$, while a (poor) choice of k databases with no relevant content results in $R_k = 0$. We note that when a database receives the "default" score from a database selection algorithm (i.e., when the score assigned to a database for a query is equal to the score assigned to an empty query) we consider that the database is *not* selected for searching. This sometimes results in a database selection algorithm selecting fewer than k databases for a query.

The R_k metric relies on human-generated relevance judgments for the queries and documents. For our experiments on database selection accuracy, we focus on the *TREC4* and *TREC6* data sets, which include queries and associated relevance judgments. (We do not consider the *Web* data set for these experiments because of the lack of relevance judgments for it.) We use queries 201-250 from TREC-4 with the *TREC4* data set and queries 301-350 from TREC-6 with the *TREC6* data set. The TREC-4 queries are long, with 8 to 34 words and an average of 16.75 words per query. The TREC-6 queries are shorter, with 2 to 5 words and an average of 2.75 words per query.

We considered eliminating stopwords (e.g., "the") from the queries, as well as applying stemming to the query and document words (e.g., so that a query *[computers]* matches documents with word "computing"). While the results improve with stopword elimination, a paired t-test showed that the difference in performance is not statistically significant. Stemming can help alleviate the data spareseness problem. We generated content summaries with and without application of stemming. While stemming tends to improve performance for small values of k, the results are mixed when $k > 10$. Due to space constraints we only report results with stopword elimination and stemming.



(a) *TREC4*, bGlOSS



(b) *TREC6*, LM

Figure 5: The R_k ratio for bGlOSS and LM over the *TREC4* and *TREC6* data sets.

Figure 4 shows results for the CORI database selection algorithm (Section 5.3). We consider both the *TREC4* and *TREC6* data sets and queries, as well as the *QBS* and *FPS* content summary construction strategies (Section 5.2). We consider applying CORI over "unshrunk" content summaries (QBS-Plain and FPS-Plain), using our adaptive shrinkage-based strategy (QBS-Shrinkage and FPS-Shrinkage), and using the hierarchical algorithm presented in [17] (QBS-Hierarchical and FPS-Hierarchical) (see Section 5.3). Due to space constraints, Figure 5 shows only a more limited set of experiments for the bGlOSS and LM database selection algorithms (Section 5.3). The results that we omit confirm the general trends in the reported results. Overall, a paired t-test shows that QBS-Shrinkage improves the database selection performance over QBS-Plain, and this improvement is statistically significant ($p < 0.05$). FPS-Shrinkage also improves the database selection performance relative to FPS-Plain, but this improvement is statistically significant only when $k < 10$. We now describe the details of our findings.

Shrinkage vs. Plain: The first conclusion from our experiments is that QBS-Shrinkage and FPS-Shrinkage improve performance compared to QBS-Plain and FPS-Plain, respectively. Shrinkage helps because new words are added in the content summaries in a *database-* and *category-specific* manner. In Table 10 we list the number of times shrinkage was applied for each database selection algorithm. Since the queries for *TREC6* are shorter, shrinkage was applied comparatively fewer times for *TREC6* than for *TREC4*.

Shrinkage vs. Hierarchical: QBS-Hierarchical and FPS-Hierarchical [17] generally outperform their "plain" counterparts. This confirms the conclusion in [17] that categorization information helps compensate for incomplete summaries. Exploiting this categorization via shrinkage results in even higher accuracy: QBS-Shrinkage and FPS-Shrinkage significantly outperform QBS-Hierarchical and FPS-Hierarchical. This improvement is due to the "flat" nature of our shrinkage method: QBS-Shrinkage and FPS-Shrinkage can rank the databases "globally," while QBS-Hierarchical and FPS-Hierarchical make irreversible choices at each *category* level of the hierarchy. Even when a chosen category contains only a small number of databases with relevant documents, the hierarchical algorithm continues to select (irrelevant) databases from the (relevant) category. When a query "cuts across" multiple categories, the hierarchical algorithm might fail to select the appropriate databases. In contrast, our shrinkage-based approach can potentially select databases from multiple categories and hence manages to identify the appropriate databases for a query, no matter if they are similarly classified or not.

Adaptive vs. Universal Application of Shrinkage: The strategy in Section 4 decides dynamically when to apply shrinkage for database selection. To understand whether this decision step is necessary, we evaluated the performance of the algorithms when we *always* decide to use shrinkage (i.e., when the $\hat{\mathcal{R}}(D_i)$ content summary is always chosen in Figure 3). The only database selection algorithm that performs better in this case is bGlOSS: unlike CORI and LM, bGlOSS does not have any form of "smoothing" already built in, so if a query word is not present in a content summary, the corresponding database gets a zero score from bGlOSS. In contrast, CORI and LM handle such cases more gracefully via smoothing. When we applied shrinkage universally, both CORI and LM performed worse. For lack of space we do not report the actual experimental results beyond this discussion.

Frequency Estimation: Frequency estimation (Section 5.2) considerably improved the performance of CORI, by 20% to 30%, with respect to the case where the raw word frequencies for the document sample are used. In contrast, frequency estimation has little effect on the performance of bGlOSS and LM: bGlOSS and LM rely on *probabilities* that remain virtually unaffected after the frequency estimation step, while CORI relies on document frequencies.

Evaluation Conclusions: A general conclusion from the database selection experiments is that shrinkage significantly improves database selection. This improvement is achieved by just exploiting the topical classification of the databases, without any additional sampling cost.

7. RELATED WORK

Most database selection algorithms [13, 23, 30, 33, 4, 11, 6, 28, 5, 32] rely on statistical content summaries of the databases. Recent work by Liu et al. [19] estimates the potential inaccuracy of the database rank produced for a query by a database selection algorithm. If this inaccuracy is unacceptably large, then the query is dynamically evaluated on a few carefully chosen databases to reduce the uncertainty associated with the database rank. This work does not take content-summary accuracy into consideration. In contrast, we address the scenario where summaries are derived from document samples –and are hence incomplete– and decide dynamically whether shrinkage should be applied, without actually querying databases during database selection.

Data Set	*Sampling Method*	*Database Selection*	*Shrinkage Application*
TREC4	FPS	bGlOSS	35.42%
		CORI	17.32%
		LM	15.40%
	QBS	bGlOSS	78.12%
		CORI	15.68%
		LM	17.32%
TREC6	FPS	bGlOSS	33.43%
		CORI	13.12%
		LM	12.78%
	QBS	bGlOSS	58.94%
		CORI	14.32%
		LM	11.73%

Table 10: Percentage of query-database pairs for which shrinkage was applied.

To extract content summaries from "uncooperative" databases, Callan et al. [2, 3] presented the *QBS* technique (reviewed in Section 2.2), while in our previous work [17] we introduced the *FPS* technique (reviewed in Section 5.2). We used both *QBS* and *FPS* in our experimental evaluation in Section 6. Craswell et al. [7] compared database selection algorithms in the presence of incomplete content summaries, extracted using document sampling, and observed that the performance of the algorithms deteriorated with respect to their behavior over complete summaries.

Xu and Croft [31] and Larkey et al. [18] show that organizing documents by topic helps improve database selection accuracy. Other database selection algorithms rely on hierarchical classification schemes –mostly *for efficiency*– to direct queries to appropriate categories of the hierarchy [9, 26, 13, 5, 32].

Our previous work in [17] is closest to this paper and shows that a hierarchical classification scheme helps compensate for the incompleteness of the extracted content summaries. In this paper, we build on this observation and generate "shrunk" content summaries that can be used subsequently by any *flat* (and hence more flexible) database selection algorithm. In Section 6 we experimentally compared our shrinkage-based technique against the hierarchical database selection algorithm from [17].

Our content summary construction technique is based on the work by McCallum et al. [22], who introduced a shrinkage-based approach for hierarchical document classification. We adapted their approach to our problem; see Section 3.

8. CONCLUSION

Database selection is critical to building efficient metasearchers that interact with potentially large numbers of databases. Exhaustively searching all available databases to answer each query is impractical (or even not possible) in increasingly common scenarios. In this paper, we showed how to improve the performance of database selection algorithms in the realistic case where complete database content summaries are not available. In such scenarios, content summaries need to be derived from relatively small document samples, which results in incomplete summaries that could hurt the performance of database selection algorithms. To alleviate this degradation in performance, our method exploits content summaries of similarly classified databases and combines them in a principled manner using "shrinkage." The shrinkage-based content summaries are more complete than their "unshrunk" counterparts and can substantially improve the accuracy of database selection. Our shrinkage-based technique achieves this performance gain efficiently, without requiring any increase in the size of the document samples.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IIS-98-17434. We thank Regina Barzilay and Victor Liu for their comments, and Jamie Callan and Luo Si for their help with the experimental setup.

9. REFERENCES

[1] L. A. Adamic. Zipf, power-laws, and Pareto – A ranking tutorial. Available at `http://ginger.hpl.hp.com/shl/-papers/ranking/ranking.html`.

[2] J. P. Callan and M. Connell. Query-based sampling of text databases. *ACM TOIS*, 19(2), 2001.

[3] J. P. Callan, M. Connell, and A. Du. Automatic discovery of language models for text databases. In *SIGMOD'99*, 1999.

[4] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR'95*, 1995.

[5] Y. S. Choi and S. I. Yoo. Text database discovery on the Web: Neural net based approach. *JIIS*, 16(1), Jan. 2001.

[6] J. G. Conrad, X. S. Guo, P. Jackson, and M. Meziou. Database selection using actual physical and acquired logical collection resources in a massive domain-specific operational environment. In *VLDB 2002*, 2002.

[7] N. Craswell, P. Bailey, and D. Hawking. Server selection on the World Wide Web. In *ICDL 2000*, 2000.

[8] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B(39), 1977.

[9] R. Dolin, D. Agrawal, and A. El Abbadi. Scalable collection summarization and selection. In *ICDL'99*, 1999.

[10] J. C. French, A. L. Powell, J. P. Callan, C. L. Viles, T. Emmitt, K. J. Prey, and Y. Mou. Comparing the performance of database selection algorithms. In *SIGIR'99*, 1999.

[11] N. Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM TOIS*, 17(3), May 1999.

[12] L. Gravano, C.-C. K. Chang, H. García-Molina, and A. Paepcke. *STARTS*: Stanford proposal for Internet meta-searching. In *SIGMOD'97*, 1997.

[13] L. Gravano, H. García-Molina, and A. Tomasic. *GlOSS*: Text-source discovery over the Internet. *ACM TODS*, 24(2), June 1999.

[14] L. Gravano, P. G. Ipeirotis, and M. Sahami. QProber: A system for automatic classification of hidden-web databases. *ACM TOIS*, 21(1), Jan. 2003.

[15] D. Harman. Overview of the Fourth Text REtrieval Conference (TREC-4). In *NIST Special Publication 500-236: The Fourth Text REtrieval Conference (TREC-4)*, 1996.

[16] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer Verlag, Aug. 2001.

[17] P. G. Ipeirotis and L. Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. In *VLDB 2002*, 2002.

[18] L. S. Larkey, M. E. Connell, and J. P. Callan. Collection selection and results merging with topically organized U.S. patents and TREC data. In *CIKM 2000*, 2000.

[19] Z. Liu, C. Luo, J. Cho, and W. Chu. A probabilistic approach to metasearching with adaptive probing. In *ICDE 2004*, 2004.

[20] B. B. Mandelbrot. *Fractal Geometry of Nature*. W. H. Freeman & Co., 1988.

[21] J. P. Marques De Sá. *Applied Statistics*. Springer Verlag, 2003.

[22] A. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *ICML'98*, 1998.

[23] W. Meng, K.-L. Liu, C. T. Yu, X. Wang, Y. Chang, and N. Rishe. Determining text databases to search in the Internet. In *VLDB'98*, 1998.

[24] G. A. Monroe, J. C. French, and A. L. Powell. Obtaining language models of web collections using query-based sampling techniques. In *HICSS'02*, 2002.

[25] G. Salton and M. J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, 1983.

[26] M. A. Sheldon. *Content Routing: A Scalable Architecture for Network-Based Information Discovery*. PhD thesis, M.I.T., 1995.

[27] L. Si and J. P. Callan. Relevant document distribution estimation method for resource selection. In *SIGIR 2003*, 2003.

[28] L. Si, R. Jin, J. P. Callan, and P. Ogilvie. A language modeling framework for resource selection and results merging. In *CIKM 2002*, 2002.

[29] E. Voorhees and D. Harman. Overview of the Sixth Text REtrieval Conference (TREC-6). In *NIST Special Publication 500-240: The Sixth Text REtrieval Conference (TREC-6)*, 1998.

[30] J. Xu and J. P. Callan. Effective retrieval with distributed collections. In *SIGIR'98*, 1998.

[31] J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. In *SIGIR'99*, 1999.

[32] C. T. Yu, W. Meng, K.-L. Liu, W. Wu, and N. Rishe. Efficient and effective metasearch for a large number of text databases. In *CIKM'99*, 1999.

[33] B. Yuwono and D. L. Lee. Server ranking for distributed text retrieval systems on the Internet. In *DASFAA'97*, 1997.

A. ESTIMATING WORD FREQUENCIES

In our previous work [17] we described a technique to estimate the *real* frequencies of the words in a database. This technique exploits Mandelbrot's law [20], which states a relationship $f = \beta(r + c)^{\alpha}$ between the rank r and the frequency f of a word in a text database. The constants α, β, and c are parameters of the specific document collection. We used the rank-frequency statistics for the words that are sent to a database as one-word queries during document sampling. (The number of matches for each of these queries corresponds to the frequency of the associated word in the database.) Based on these statistics, we used curve fitting to compute the α, β, and c values for the database. The resulting formula is used to compute the frequencies of the words in the sample that were not single-word queries.

One problem with this approach is that it assumes that the rank of a word is the same in both the document sample and in the database. Unfortunately, this is many times not the case, especially for low-ranked words. When low-ranked words are sent as query probes and used for fitting the distribution, the resulting "curve" is a bad fit for high-ranked words, which tend to have largely overestimated frequencies.

In this paper, we avoid using the incorrect rank estimates *for distribution fitting*. Instead, we use rank and frequencies as they appear *in the sample* to fit Mandelbrot's law. However, the parameters that describe the frequency distribution in the sample are not assumed to be the same as those for the actual database. We examine how the parameters change for different sample sizes and use a slightly simplified version of Mandelbrot's law [20], where $f = \beta r^{\alpha}$. We measured α and β for database samples of different size and observed that α and $\log(\beta)$ generally tend to increase logarithmically with the sample size $|S|$. Specifically:

$$\alpha = A_1 \log(|S|) + A_2 \tag{4a}$$

$$\log(\beta) = B_1 \log(|S|) + B_2 \tag{4b}$$

where A_1, A_2, B_1, and B_2 are database-specific constants, independent of the sample size.

Based on the above empirical observations, we proceed as follows for a database D. At different points during the document sampling process, we calculate α and β. After finishing sampling, we use regression to estimate the values of A_1, A_2, B_1, and B_2. We also estimate the size of database D using the "sample-resample" method presented in [27]. Finally, we compute the values of α and β for the database by substituting the estimated $|D|$ for $|S|$ in Equations 4a and 4b. The final equation that we use to estimate the frequency of a word that appears in the document sample is then:

$$\log(f) = \underbrace{(A_1 \log(|D|) + A_2)}_{\alpha} \log(r) + \underbrace{B_1 \log(|D|) + B_2}_{\log(\beta)} \tag{5}$$

where r is the rank of the word in the sample.

B. ESTIMATING SCORE DISTRIBUTIONS

Section 4 discussed how to estimate the "uncertainty" associated with a database score for a query. Specifically, this estimate relied on the probability P of the different possible query keyword frequencies. To compute P, we assume independence of the words in the sample: P is a product of the $p(d_k|s_k)$ values, defined in turn as the probability that w_k occurs in d_k documents in D given that it occurs in s_k documents in S. Using the Bayes rule, we have $p(d_k|s_k) = \frac{p(s_k|d_k)p(d_k)}{\sum_{i=0}^{|D|} p(d_i)p(s_k|d_i)}$. To compute $p(s_k|d_k)$, we assume that the presence of each word w_k follows a binomial distribution over the S documents, with $|S|$ trials and probability of success $\frac{d_k}{|D|}$ for every trial. Then, $p(s_k|d_k) = \binom{|S|}{s_k}(\frac{d_k}{|D|})^{s_k}(1 - \frac{d_k}{|D|})^{|S|-s_k}$. Finally, to compute $p(d_k)$ we use the well-known fact that the distribution of words in text databases tends to follow a power law [20]: approximately cf^{γ} words in a database have frequency f, where c and γ are database-specific constants. Then, $p(d_k) = \frac{cd_k^{\gamma}}{\sum_{i=1}^{|D|} ci^{\gamma}}$. Interestingly, $\gamma = \frac{1}{\alpha} - 1$, where α is a parameter of the frequency-rank distribution of the database [1] and can be computed as described in Appendix A.