Distributed Query Sampling: A Quality-Conscious Approach*

James Caverlee
Georgia Institute of Technology
Atlanta, GA, 30332, USA
caverlee@cc.gatech.edu

Ling Liu
Georgia Institute of Technology
Atlanta, GA, 30332, USA
lingliu@cc.gatech.edu

Joonsoo Bae†
Chonbuk National University
Jeonju, Jeonbuk, 561-756, South Korea
jsbae@chonbuk.ac.kr

ABSTRACT

We present an *adaptive distributed query-sampling framework* that is quality-conscious for extracting high-quality text database samples. The framework divides the query-based sampling process into an initial seed sampling phase and a quality-aware iterative sampling phase. In the second phase the sampling process is dynamically scheduled based on estimated database size and quality parameters derived *during* the previous sampling process. The unique characteristic of our adaptive query-based sampling framework is its self-learning and self-configuring ability based on the overall quality of all text databases under consideration. We introduce three quality-conscious sampling schemes for estimating database quality, and our initial results show that the proposed framework supports higher-quality document sampling than existing approaches.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]; H.3.1 [Content Analysis and Indexing]

General Terms: Algorithms, Experimentation

Keywords: adaptive, sampling, quality, distributed IR

1. INTRODUCTION

In a variety of contexts, from digital libraries to Web databases, there are a large number of distributed text databases for which query-based access is the primary means of interaction. Since many databases are autonomous, offer limited query capability, and may be unwilling to allow complete access to their entire archives, *query-based sampling* mechanisms have become a popular approach for collecting document samples. The sampling-based approach has garnered previous research attention and shown success in several contexts, including distributed information retrieval, database selection, database categorization, and peer-to-peer information retrieval (e.g., [5, 15, 18, 25]).

There are three key challenges for using a sampling-based approach to analyze distributed text databases. First, the approach must understand the query interface of each database for effectively sampling from the space of all documents available at the database [*how to sample*]. This entails parsing the query interface, automatically filling out the query interface, and selecting appropriate queries to probe the target databases. Second, given the rate of change and evolution of many distributed text databases, we need to consider the appropriate sampling schedule to maintain sample freshness [*when to sample*]. Finally, given realistic network constraints and the size and growth rate of distributed text databases, we require an approach for knowing what to sample from each database [*what to sample*].

While there have been some previous efforts to study how to sample (e.g., [4, 24]) and when to schedule such samples (e.g., [13]), we provide the first comprehensive investigation of the problem of *what to sample* from each text database. Concretely, given a set of n distributed databases, each of which provides access primarily through a query-based mechanism, and given limited resources for sampling all n databases, can we identify an effective strategy for extracting high-quality samples from the n databases? We refer to this as the *distributed query-sampling problem*.

In this paper, we present an *adaptive distributed query-sampling framework* that is quality-conscious for extracting high-quality database samples. The framework divides the query-based sampling process into an initial seed sampling phase and a quality-aware iterative sampling phase. In the second phase the sampling process is dynamically scheduled based on estimated database size and quality parameters derived *during* the previous sampling process.

2. BASIC REFERENCE MODEL

Before introducing the distributed query-sampling framework, we first describe the basic reference model used in this paper.

We consider a *text database* to be a database that is composed primarily of text documents and that provides query-based access to these documents either through keyword search or more advanced search operators. Examples of text databases include Deep Web data sources, digital libraries, and legacy databases searchable only through query-based mechanisms.

We consider a universe of discourse $\mathcal{U}$ consisting of n text databases: $\mathcal{U} = \{D_1, D_2, \ldots, D_n\}$ where each database produces a set of

*This research is partially supported by NSF CNS, NSF ITR, IBM SUR grant, and HP Equipment Grant. Any opinions, findings, and conclusions or recommendations expressed in the project material are those of the authors and do not necessarily reflect the views of the sponsors.

†Work performed while visiting Georgia Tech.

SIGIR'06, August 6–10, 2006, Seattle, Washington, USA.

documents in response to a query. A text database D is described by the set of documents it contains: $D = \{doc_1, doc_2, \cdots\}$. We denote the number of documents in D as $|D|$. We denote the set of terms in D as the *vocabulary* V of D. The number of unique terms, denoted by $|V|$, is referred to as the *vocabulary size* of the database D. We also track the following statistics: $c(t, D)$ denotes the count of documents in D each term t occurs in; and $f(t, D)$ denotes the frequency of occurrence of each term t across all documents in D (i.e., how many times the term appears in the text database).

A document *sample* from database D, denoted by D_s, consists of a set of documents from D. We use $|D_s|$ to denote the number of documents in the sample D_s, where typically $|D_s| \ll |D|$. We refer to the set of terms in D_s as the *sample vocabulary* V_s of D_s. The number of unique terms $|V_s|$ is referred to as the *sample vocabulary size*. We also track the following statistics for each sample D_s: $c(t, D_s)$ denotes the count of documents in D_s each term t occurs in; and $f(t, D_s)$ denotes the frequency of occurrence of each term t across the sampled documents in D_s.

2.1 Query-Based Sampling

To support database sampling, Callan and his colleagues [4, 3] have previously introduced the query-based sampling approach for generating estimates of text databases by examining only a fraction of the total documents. The query-based sampling algorithm works by repeatedly sending one-term keyword queries to a text database and extracting the response documents:

Steps of Query-Based Sampling from a Database D
1: Initialize a query dictionary Q.
2: Select a one-term query q from Q.
3: Issue the query q to the database D.
4: Retrieve the top-m documents from D in response to q.
5: [Optional] Update Q with the terms in the retrieved documents.
6: Goto Step 2, until a stopping condition is met.

Critical factors impacting the performance of Query-Based Sampling algorithms include the choice of the query dictionary Q, the query selection algorithm, and the stopping condition. The optional step 5 allows the algorithm to learn a database-specific query dictionary as a source of queries after the first successful query from the initial query dictionary. Previous research [4, 3] has shown that document samples extracted by Query-Based Sampling can be of high quality based on a number of quality metrics. Concretely, these studies show that document samples consisting of a fairly small number of documents (e.g., 300) are of high quality over text databases consisting of millions of unique documents.

2.2 Sampling From Distributed Text Databases

The problem of distributed query sampling is to determine what to sample from each of the n text databases under a given sampling resource constraint. To simplify the discussion, we assume that there are uniform sampling costs from each database, and that the distributed query sampling algorithm may sample at most a total of S documents from the n databases. Hence, the goal of the distributed query-sampling framework is to identify the optimal allocation of the S documents to the n databases.

Naive Solution – Uniform Sampling: The simplest sampling framework is to uniformly allocate the S sample documents to each database, meaning that for each database an equal number of documents will be sampled, i.e. $\lfloor S/n \rfloor$. The uniform approach is fairly standard and has been widely adopted, e.g., [12, 14, 20, 25]. Such a uniform allocation is indifferent to the relative size of each database or the relative quality of the document samples. We argue that the uniform approach misses the relative quality of each database sample with respect to the entire space of sampled databases.

Table 1: Sampling Notation

S	total number of sample documents
S_{seed}	total number of sample documents for Seed Sampling
S_{dyn}	total number of sample documents for Dynamic Sampling
$s_{seed}(D_i)$	number of sample documents allocated for Seed Sampling of database D_i
$s_{dyn}(D_i, j)$	number of sample documents allocated for Dynamic Sampling of database D_i in iteration j
$s_{tot}(D_i)$	$s_{seed}(D_i) + \sum_{j=1}^{m} s_{dyn}(D_i, j)$
$\hat{s}(D_i)$	estimated total number of sample documents that should be allocated to database D_i

3. ADAPTIVE ARCHITECTURE FOR DISTRIBUTED QUERY-SAMPLING

In this section we introduce a distributed and adaptive query-sampling framework and examine three quality-conscious sampling schemes that estimate critical database size and vocabulary parameters for guiding the sampling process. The proposed adaptive sampling framework dynamically determines the amount of sampling at each text database based on an analysis of the relative merits of each database sample *during* the sampling process.

3.1 Adaptive Sampling Steps: An Overview

The adaptive sampling approach to the distributed query sampling problem divides the sampling process into three steps:

1. Seed Sampling: In this step, we collect an initial seed sample from each database to bootstrap the iterative distributed query-sampling process. A portion of the total sample documents S is allocated for seed sampling, denoted by S_{seed}. One simple approach to selecting a seed sample is to use the uniform allocation of S to n databases under consideration, such that each database D_i ($i = 1, ..., n$) is sampled using Query-Based Sampling until $s_{seed}(D_i) = S_{seed}/n$ documents are extracted.[1]

2. Dynamic Sampling Allocation: This step utilizes the seed samples collected from each database to estimate various size and quality parameters of each participating database. The remaining number of sample documents is denoted by S_{dyn} where $S_{dyn} = S - S_{seed}$. The adaptive sampling can be conducted in m iterations and in each iteration, S_{dyn}/m sample documents are dynamically allocated to n databases based on a quality-conscious sampling scheme. We denote the number of sample documents allocated to database D_i in iteration j as $s_{dyn}(D_i, j)$.

3. Dynamic Sampling Execution: In this step, the n databases are sampled according to the documents allocated by the Dynamic Sampling Allocation step using Query-Based Sampling. In the case of $m > 1$, steps 2 and 3 will be iterated m times. At the end of the Dynamic Sampling Execution step, we will have sampled for each database D_i a total number of documents, denoted by $s_{tot}(D_i)$, such that $s_{tot}(D_i) = s_{seed}(D_i) + \sum_{j=1}^{m} s_{dyn}(D_i, j)$.

Table 1 summarizes the notation introduced in this section.

3.2 Quality-Conscious Sampling Schemes

In this section, we describe three quality-conscious sampling schemes. Each scheme recommends for each D_i a total number of documents to sample, denoted by $\hat{s}(D_i)$, which is designed to be a close approximation of what would be recommended if complete information about each database were available, denoted by $s(D_i)$. We then discuss how to find the dynamic sampling allocation $s_{dyn}(D_i, j)$ for each database based on $\hat{s}(D_i)$.

[1]We require that all sample sizes be integers since they correspond to the number of documents to be sampled from each database. In the rest of the paper, we shall omit the rounding of sample sizes for presentation purposes.

3.2.1 Scheme 1: Proportional Document Ratio [PD]

The first proposed quality-conscious sampling scheme is based on the relative size of each database. Instead of simply collecting the same number of documents, this scheme seeks to collect the same proportion of documents from each database, say 5% of the documents at D_1, 5% of the documents at D_2, and so on.

If we assume that the sampling framework has access to the total number of documents $|D_i|$ in each database D_i, $i = 1, ..., n$, then the proportion of documents to be sampled from each database is governed by the total documents available for sampling and the total size of the databases to be sampled: $ratio_{PD} = S/\sum_{i=1}^{n} |D_i|$.

Hence, the goal of the Proportional Document Ratio (PD) scheme is to sample the same fraction $ratio_{PD}$ from each database. So, the total number of documents to be extracted from database D_i is:

$$s_{PD}(D_i) = ratio_{PD} \cdot |D_i| = \frac{S}{\sum_{j=1}^{n} |D_j|} \cdot |D_i|$$

Of course, the actual number of documents in each database may not be known a priori. As a result, the PD scheme will typically rely on an estimate $|\tilde{D}_i|$ of the number of documents in database D_i, instead of the actual size $|D_i|$. Hence, we must estimate the fraction of documents to be extracted from each database with:

$$\widetilde{ratio}_{PD} = \frac{S}{\sum_{i=1}^{n} |\tilde{D}_i|}$$

As a result, we may approximate $s_{PD}(D_i)$ with the estimate:

$$\hat{s}_{PD}(D_i) = \widetilde{ratio}_{PD} \cdot |\tilde{D}_i| = \frac{S}{\sum_{j=1}^{n} |\tilde{D}_j|} \cdot |\tilde{D}_i|$$

To find $\hat{s}_{PD}(D_i)$, we estimate the database size by analyzing the document samples extracted in the Seed Sampling step and in previous iterations of the Dynamic Sampling step. There have been two previous efforts to estimate the number of documents in a text database: (1) the capture-recapture algorithm in [17]; and (2) the sample-resample algorithm in [26]. Both approaches rely on analyzing a document sample, but the sample-resample approach has been shown to be more accurate and less expensive in terms of queries and the number of documents necessary to sample.

The sample-resample technique assumes that the database responds to queries by indicating the total number of documents in which the query occurs (although only a fraction of this total will be available for download by the client). The "sample" phase collects a document sample, then the "resample" phase issues a handful of additional queries and collects the $c(t, D)$ statistics for these queries. The driving assumption of this technique is that the fraction of sampled documents that contain a term is the same as the fraction of all documents in the database that contain the same term, i.e., $c(t, D_s)/|D_s| = c(t, D)/|\tilde{D}|$, where the database provides $c(t, D)$ or a reasonable approximation. Hence, $|\tilde{D}| = |D_s| \cdot c(t, D)/c(t, D_s)$. This database size estimate may be further refined by considering a set of probe terms and taking the average. In practice, we use a weighted random selection of resample probes based on the term frequency in the sampled documents and collect $c(t, D)$ statistics for all queries.

3.2.2 Scheme 2: Proportional Vocabulary Ratio [PV]

Instead of collecting the same proportion of documents from each database, the Proportional Vocabulary Ratio (PV) scheme seeks to sample the same *vocabulary* proportion from each database, say 10% of the vocabulary terms at D_1, 10% at D_2, and so on. Unlike the first scheme, the PV scheme is intuitively more closely linked to the quality of the database samples, since it emphasizes the presence of *unique* terms, and not just document quantity.

According to Heaps Law [2], a text of n words will have a vocabulary size of Kn^{β}, where K and β are free parameters and typically $0.4 \leq \beta \leq 0.6$. In the context of distributed text databases, documents sampled earlier in the sampling process will tend to contribute more new vocabulary terms than will documents that are sampled later in the process.

Suppose we randomly examine documents in a database and then plot the number of documents examined versus the vocabulary size, yielding a Heaps Law curve. To identify the number of sample documents $s_{PV}(D_i)$ necessary to extract $ratio_{PV} \cdot |V_i|$ vocabulary terms, we need only consult the Heaps Law curve to identify $s_{PV}(D_i)$. Due to the inherently random nature of the sampling process, this technique provides an average-case estimate of the number of documents necessary to achieve a particular fraction of all vocabulary terms.

In practice, the sampling framework must rely only on the previously sampled documents to estimate the vocabulary fraction and the corresponding number of documents necessary to achieve the fraction $ratio_{PV} \cdot |V_i|$. To estimate the number of documents necessary requires that we carefully inspect the previously sampled documents. We may begin by approximating the total size of the vocabulary $|V|$ of D based on a sample of documents D_s by relying on a version of Heaps Law adapted to the distributed text database domain: $|V| = K\,(f(D))^{\beta}$ where $f(D) = \sum_{t \in V} f(t, D)$ refers to the total frequency of all terms in D. The key for the vocabulary estimation is to identify the appropriate values for K, β, and $f(D)$ based on the sample documents only. Hence, we will find K_s, β_s, and $f(\tilde{D})$ respectively by analyzing the sample documents only.

Estimating $f(D)$: If we let $|d|_{avg}$ denote the average number of terms per document for the entire database, then we may write $f(D)$ as a product of $|d|_{avg}$ and the total number of documents in the database: $f(D) = |d|_{avg} \cdot |D|$. Since we showed how to approximate $|D|$ with $|\tilde{D}|$ in the previous section, we need only approximate $|d|_{avg}$ in order to find $f(\tilde{D})$. We can estimate the average number of terms per document $|d|_{avg}$ for the entire database by examining the seed sample: $|\tilde{d}|_{avg} = f(D_s)/|D_s|$, where $f(D_s)$ refers to the total frequency of all terms in the document sample D_s. Hence, we have $f(\tilde{D}) = |\tilde{d}|_{avg} \cdot |\tilde{D}|$.

Estimating K and β: To approximate K and β with K_s and β_s, respectively, we consider increasingly larger sub-samples of the total set of sampled documents. In particular, we randomly select a document (without replacement) from the set of sampled documents D_s and add it to the sub-sample D_{ss}. For each sub-sample D_{ss}, we plot the total term frequency (i.e. text size) $f(D_{ss})$ versus the actual vocabulary size $|V_{ss}|$ of the sub-sample. This process repeats until all documents in the sample have been selected. As a result, we will have $|D_s|$ points consisting of the text size and the corresponding vocabulary size. We may then estimate K_s and β_s by using a curve fitting tool to fit the best function that conforms to Heaps Law to these $|D_s|$ points.

With these estimated parameters, the total vocabulary size $|V|$ of the database may be approximated with the estimate $|\tilde{V}|$ by calculating: $|\tilde{V}| = K_s(|f(\tilde{D})|)^{\beta_s}$. Hence, by analyzing only the documents sampled in earlier steps, we may estimate the total vocabulary size for each of the n text databases. To find the sample size $\hat{s}_{PV}(D)$ that should yield $ratio_{PV} \cdot |\tilde{V}|$ vocabulary terms at a database, we can rely on the estimates $|\tilde{V}|$, K_s, β_s, and $|\tilde{d}|_{avg}$ for

each database to yield n vocabulary ratio equations, where each is of the form:

$$ratio_{PV} \cdot |\tilde{V}| = K_s(|\tilde{d}|_{avg} \cdot \hat{s}_{PV}(D))^{\beta_s}$$

Solving for $\hat{s}_{PV}(D)$ yields:

$$\hat{s}_{PV}(D) = e^{\frac{\ln ratio_{PV}|\tilde{V}| - \ln K_s}{\beta_s} - \ln |\tilde{d}|_{avg}}$$

Given the n equations for $\hat{s}_{PV}(D_i)$ (one for each database), we need to determine the appropriate choice of $ratio_{PV}$ such that the total documents to be sampled is $S = \sum_1^n \hat{s}_{PV}(D_i)$, where $0 \leq ratio_{PV} \leq 1$. Since each equation is monotonically increasing with respect to the choice of $ratio_{PV}$, we may solve the non-linear optimization problem using a simple search of the space $[0, 1]$ to find $ratio_{PV}$ such that $S = \sum_1^n \hat{s}_{PV}(D_i)$.

3.2.3 Scheme 3: Vocabulary Growth [VG]

The final sampling scheme is based on the vocabulary growth rate at each database. The goal of the Vocabulary Growth (VG) scheme is to extract the *most* vocabulary terms from across the space of distributed text databases.

This scheme relies on the Heaps Law parameter estimates κ_s and β_s for each database, as we presented in the previous section. By analyzing how fast each database "produces" new vocabulary terms as more documents are sampled, our goal is to extract the "cheapest" vocabulary terms first. Some databases may have a very slow growing vocabulary, meaning that many more documents must be sampled to equal the same number of vocabulary terms as a database with a fast growing vocabulary. At some point the additional vocabulary terms for each sampled document may slow to a point that additional sampling is not worthwhile in the context of many distributed text databases.

To guide the VG sampling scheme, we first estimate the growth rate of the vocabulary size of each database by considering the incremental vocabulary terms that each additional sampled document may be expected to yield. If we let x denote the number of documents sampled from a database, then we can write the estimated vocabulary size for the x documents as $|V(x)|$:

$$|V(x)| = K_s(|\tilde{d}|_{avg} \cdot x)^{\beta_s}$$

To determine the incremental vocabulary terms – denoted by $\Delta(|V(x)|)$ – available by sampling x documents versus sampling $x-1$ documents, we take the difference between the expected vocabulary size for a sample of size x documents and for a sample of size $x-1$:

$$\Delta(|V(x)|) = K_s(|\tilde{d}|_{avg} \cdot x)^{\beta_s} - K_s(|\tilde{d}|_{avg} \cdot (x-1))^{\beta_s}$$

Hence, we may determine the expected incremental vocabulary terms of each successive sampled document. For each of the databases, we may calculate the incremental vocabulary terms $\Delta(|V(x)|)$ for all documents available at the database. If we consider the "price" of each additional vocabulary term extracted from a database as the number of documents per new vocabulary term, then we may choose to sample from each database based on the "cheapest" database for extracting new vocabulary terms. Equivalently, we choose from which database to sample based on how many additional vocabulary terms it is expected to yield per document sampled. With a total number of documents to sample of S, we select the top-S documents from across all databases as scored by $\Delta(|V(x)|)$. We then allocate to each database $\hat{s}_{VG}(D_i)$ documents based on the total number of D_i's documents in the top-S.

3.3 Dynamic Sampling Allocation

We have described three quality-conscious sampling schemes, and have seen how each recommends a total number of documents to allocate to each database. Now we need to determine the number of documents to be sampled from each of the n databases for dynamic sampling in iteration k – $s_{dyn}(D_i, k)$ – such that the total documents sampled from each database closely matches the number of documents prescribed by the sampling scheme, that is: $\hat{s}(D_i) \approx s_{tot}(D_i)$. There are two cases to consider:

Case 1. If for all databases, the Seed Sampling step and the previous $k-1$ iterations of the Dynamic Sampling have extracted fewer documents than the scheme recommends in total (i.e., $\hat{s}(D_i) > s_{seed}(D_i) + \sum_{j=1}^{k-1} s_{dyn}(D_i, j)$), then we let the Dynamic Execution step sample for the k-th iteration:

$$s_{dyn}(D_i, k) = \frac{\hat{s}(D_i) - s_{seed}(D_i) - \sum_{j=1}^{k-1} s_{dyn}(D_i, j)}{m - k + 1}$$

In the case of a single round of dynamic sampling, then $s_{dyn}(D_i, 1)$ is simply the leftover documents to be sampled: $s_{dyn}(D_i, 1) = \hat{s}(D_i) - s_{seed}(D_i)$.

Case 2. However, it may be the case that a database has already been oversampled in the Seed Sampling step and previous $k-1$ iterations of the Dynamic Sampling with respect to the sampling recommendation by the quality-conscious sampling scheme, i.e., $\hat{s}(D_i) < s_{seed}(D_i) + \sum_{j=1}^{k-1} s_{dyn}(D_i, j)$. This oversampling requires two corrections. First, any database that has been sampled sufficiently is dropped from this Dynamic Sampling Execution step. Second, the documents allocated to the remaining databases must be re-scaled to reflect the additional documents available as a result of dropping the oversampled databases from this round.

4. EXPERIMENTS

In this section, we present three sets of experiments designed to test the distributed query-sampling framework. The experiments rely on data drawn from two standard TREC information retrieval datasets summarized in Table 2.

Table 2: Overall TREC Dataset Summary Information

Name	Size (GB)	Documents	Total Terms
TREC123	3.2	1,078,166	258,212,077
TREC4	2.0	567,529	155,575,164

TREC123: This set consists of 100 databases created from TREC CDs 1, 2, and 3. The databases are organized by source and publication date as described in [23].

TREC4: This set consists of 100 databases drawn from TREC 4 data. The databases correspond to documents that have been clustered by a k-means clustering algorithm using a KL-divergence based distance measure as described in [28].

In addition, we created six large databases to further test the distributed query-sampling framework. The large databases were created from TREC123 data and are listed in Table 3. The large database AP is composed of all 24 databases of Associated Press articles in the TREC123 dataset. Similarly, WSJ is composed of the 16 Wall Street Journal databases; FR the 13 Federal Register databases; and DOE the six Department of Energy databases. The two other databases – Rand1 and Rand2 – are each combinations of 20 non-overlapping randomly selected databases from TREC123. Based on these large databases, we created three additional datasets designed to test the sampling framework in the presence of more skewed datasets.

Table 3: Large Databases

Name	Documents	Vocab Size	Total Terms
AP	242,918	347,762	61,381,800
WSJ	173,252	314,791	43,542,976
FR	45,820	503,774	34,588,476
DOE	226,087	206,653	16,874,516
Rand1	232,031	544,558	50,129,684
Rand2	228,701	553,007	50,152,660

TREC123-A: This dataset consists of the large AP and WSJ databases, plus the 60 other TREC123 databases (excluding the AP and WSJ databases), for a total of 62 databases. The AP and WSJ databases are much larger than the other databases and contain a disproportionate share of relevant documents for the tested query mix for the database selection application scenario.

TREC123-B: This dataset consists of the large FR and DOE databases, plus the 81 other TREC123 databases (excluding the FR and DOE databases), for a total of 83 databases. The FR and DOE databases are much larger than the other databases, but contain very few relevant documents for the tested query mix.

TREC123-C: This dataset consists of the large Rand1 and Rand2 datasets, plus the 60 other TREC123 databases, for a total of 62 databases. The Rand1 and Rand2 datasets contain approximately the same proportion of relevant docs as all the other databases.

All database sampling and selection code was written in Java. The curve fitting necessary for parameter estimation was performed with Mathematica via the Java interface J/Link. Each dataset was indexed and searched using the open source Lucene search engine. The search engine indexes and database samples do not include a list of standard stopwords; the terms have been stemmed using the standard Porter's Stemmer [22]. In all cases the Query-Based Sampling component relied on a weighted random prober (by term frequency) that drew probe terms initially from the standard UNIX dictionary and subsequently from the sampled documents; a maximum of four documents were retrieved for each query.

4.1 Estimation Error

In the first set of experiments, we study how effectively the sampling framework may estimate the database size parameters necessary to drive the three sampling schemes. Depending on the sampling scheme, the seed sample is used to estimate either the number of documents at each database [for the PD scheme], or the total vocabulary size of each database and vocabulary-related parameters κ and β [for the PV and VG schemes]. We measure the relative error for the database size as $E_D = (|\tilde{D}| - |D|)/|D|$, and the relative vocabulary size error as $E_V = (|\tilde{V}| - |V|)/|V|$. We measure the error rates across the 100 TREC4 databases, the 100 TREC123 database, and the six large databases of Table 3. For each database, we extracted from 50 to 500 documents and calculated the error rate. We repeated this process five times and report the average.

We begin by reporting the database size error E_D. Across the TREC4 and TREC123 databases, as the sample size increases from 50 documents to 500, the estimation error quickly becomes reasonable. For TREC4, the relative error ranges from 13% to 18%. Similarly, for TREC123, the relative error ranges from 14% to 18%. For the large databases, the database size error ranges from 20% to 30%, on average. These results validate the results from the original sample-resample paper [26] and provide strong evidence that the database size may be estimated for large database by examining only a small fraction of the documents.

The vocabulary size estimation is significantly more difficult since it relies on estimating multiple parameters, including the κ and β parameters for the Heaps Law curve, as well as the average size of each document in the database. In Figure 1 we report the average vocabulary estimation error for the 100 TREC4 databases, the 100 TREC123 databases, and the six large databases. Since the vocabulary size estimation relies on knowing the number of documents in a database, we report the vocabulary error in the realistic case when the database size must be estimated. The results are encouraging. In all cases, the error falls quickly, requiring sample sizes of fewer than 200 documents for reasonably accurate quality estimates. Both the TREC4 and TREC123 estimates are within 50% of the actual after examining only 150 documents. For the large databases, the error is within a factor of 2 after only 100 documents.

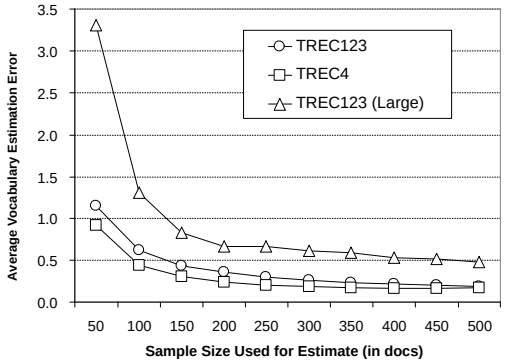


Figure 1: Vocabulary Estimation Error

4.2 Database Sample Quality

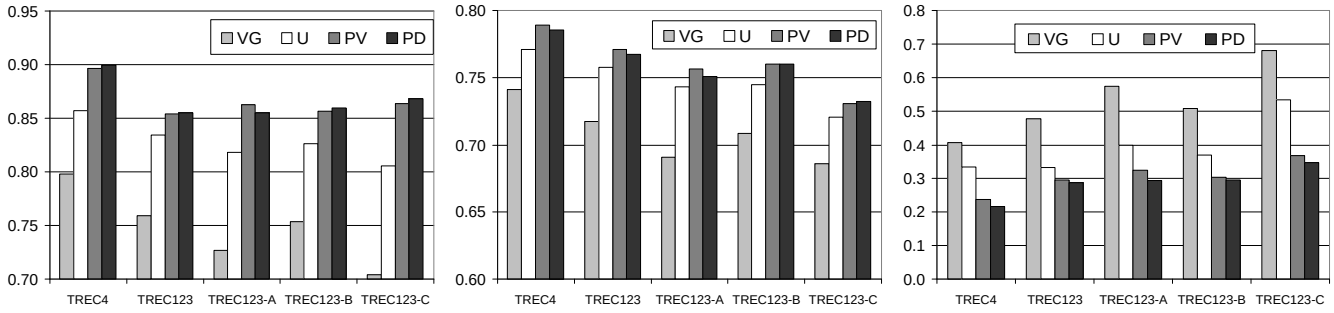
We next study the impact on the overall sample quality of the three quality-conscious sampling schemes – Proportional Document Ratio (PD), Proportional Vocabulary Ratio (PV), and Vocabulary Growth (VG) – versus the uniform approach.

Since previous research efforts have claimed that 300 documents are reasonable for extracting high-quality database samples from databases ranging in size from thousands of documents to millions (e.g., [4, 3]), for each of the datasets we assumed a total document budget of $S = 300 \cdot n$, where n is the number of databases in the dataset. So, the total budget for TREC123 (100 databases) is 30,000 documents, for TREC4 (100 databases) 30,000, and so on for TREC123-A, TREC123-B, and TREC123-C.

For each of the datasets, we collected baseline document samples using the Uniform sampling approach (U), meaning that 300 documents were sampled from each database. For the three quality-conscious sampling schemes we allocated half of the total budget for the Seed Sampling step (i.e., 150 documents per database). We then allocated the remaining half of the total budget based on the specific sampling scheme. In this first set of sample quality experiments we consider a single round of dynamic sampling. To minimize the randomness inherent in any sampling-based approach, we report the average results based on repeating the sampling five times for all tested schemes.

4.2.1 Sample Quality Metrics

To assess the quality of the database samples produced by each sampling scheme, we consider a suite of three distinct quality metrics. Each metric compares a database sample D_s to the database D from which it is drawn. For each of the three quality metrics, we measure the overall quality of the collected database samples for a dataset by calculating the average quality metric weighted by the actual size of each database: $\sum_{i=1}^{n} Q(D_{is}, D_i)/|D_i|$.



(a) Weighted Common Terms (b) Term Rankings (Spearman) (c) JS-Divergence [Lower is Better]

Figure 2: Sample Quality: Comparing the Three Quality-Conscious Sampling Schemes to the Uniform Approach

Weighted Common Terms: This first metric measures the weighted degree of term overlap between the sample and the database:

$$wct(D_s, D) = \frac{\sum_{t \in V \wedge V_s} f(t, D)}{\sum_{t \in V_s} f(t, D)}$$

Term Rankings: To assess the quality of the relative frequency of terms in the database sample, we rely on the Spearman rank correlation coefficient as defined in [3]. The Spearman coefficient measures the level of agreement between two rankings. In our case, we compare the rankings induced by ordering the terms in the actual database by $c(t, D)$ with the rankings induced by ordering the terms in the database sample by $c(t, D_s)$. The Spearman coefficient measures only the quality of the relative ranking assignment, not the values assigned to each term. If both the database and the sample rank every term in the same position, then the Spearman coefficient is 1. Uncorrelated rankings result in a Spearman coefficient of 0; reverse rankings (i.e., the top-ranked term in the database is the lowest-ranked term in the database sample) result in a Spearman coefficient of -1.

Distributional Similarity: To measure the distributional similarity of the database sample and the actual database, we rely on the Jensen-Shannon divergence (or JS-divergence) [16]. It is based on the relative entropy measure (or KL-divergence), which measures the difference between two probability distributions p and q over an event space X: $KL(q, p) = \sum_{x \in X} p(x) \cdot \log(p(x)/q(x))$. Intuitively, the KL-divergence indicates the inefficiency (in terms of wasted bits) of using the q distribution to encode the p distribution. Adopting a probabilistic interpretation, we can consider a text database D as a source randomly emitting a term t according to the overall prevalence of t in D: $Pr(t|D) = f(t, D)/f(D)$. Hence, $kl(D_s, D) = \sum_{t \in V} Pr(t|D) \cdot \log(Pr(t|D)/Pr(t|D_s))$. Unfortunately, when evaluating a database sample that lacks a single term from the actual database (which is almost always the case), the KL-divergence will be unbounded and, hence, will provide little power for evaluating database samples. In contrast, the Jensen-Shannon divergence avoids these problems. The JS-divergence is defined as:

$$js(D_s, D) = \alpha_1 \cdot kl(\alpha_1 D + \alpha_2 D_s, D) + \alpha_2 \cdot kl(\alpha_1 D + \alpha_2 D_s, D_s)$$

where $\alpha_1, \alpha_2 > 0$ and $\alpha_1 + \alpha_2 = 1$. We consider $\alpha_1 = \alpha_2 = 0.5$. The lower the JS-divergence, the more similar the two distributions.

4.2.2 Sample Quality Results

In Figure 2, we compare the Uniform (U) sampling approach to the three quality-conscious sampling schemes of the sampling framework – PD, PV, and VG. We note several interesting results. First, even under the strong constraint that the sampling schemes must rely solely on the seed samples for guiding the rest of the sampling process, we see that the PV and PD schemes outperform the uniform sampling approach U over *all five datasets* and *all three quality metrics*, validating the intuitive strengths of the distributed query-sampling framework.

Second, the VG scheme significantly underperforms the U approach in all cases. On inspection, we discovered that the VG scheme resulted in an overall collection vocabulary of from 1.5 to 3 times as many vocabulary terms versus the other approaches across all settings. As we would expect, the VG scheme was very effective at extracting the most vocabulary terms of all the schemes tested, since it focuses solely on sampling from the most efficient databases in terms of vocabulary production. The VG scheme tended to allocate all of the sampling documents to a few small databases each with a fairly large vocabulary. These databases had significantly steep vocabulary growth curves, and as a result, the overall collection vocabulary for the VG approach was higher than for the other approaches. But, since the sampling documents were assigned to only a handful of small databases, the larger databases (which tend to have slower growing vocabulary growth rates) were undersampled. We are interested in further exploring the effectiveness of the VG scheme in application scenarios that rely on rich coverage of vocabulary terms.

Given the good results for the PD and PV schemes, we next tweak several of the factors. First, we consider the impact of the total number of sample documents S on the quality of the extracted database samples. As the framework is able to sample more documents, we would expect to extract higher quality samples. We consider three scenarios. In Scenario 1, we have total sample documents $S = 100 \cdot n$, where n is the number of databases in the dataset; in Scenario 2, we have $S = 300 \cdot n$; and in Scenario 3, we have $S = 500 \cdot n$. So, for example, the total sampling allocation for TREC123 and its 100 databases is 10,000 documents in Scenario 1 up to 50,000 documents in Scenario 3.

In Figure 3, we show the impact of increasing S over the Uniform sampling approach ($U[100]$, $U[300]$, and $U[500]$) as compared to the Proportional Document sampling scheme ($PD[100]$, $PD[300]$, and $PD[500]$). For the PD cases, we allocate half the documents available for seed sampling (meaning that in Scenario 1, we collect a seed sample of 50 documents from each database; in Scenario 2, we collect a seed sample of 150 documents; in Scenario 3, we collect a seed sample of 250 documents). We restrict Figure 3 to results for two datasets and two quality metrics; note that the general results hold for all datasets and quality metrics.

Of course, as we increase the total sample document allocation,

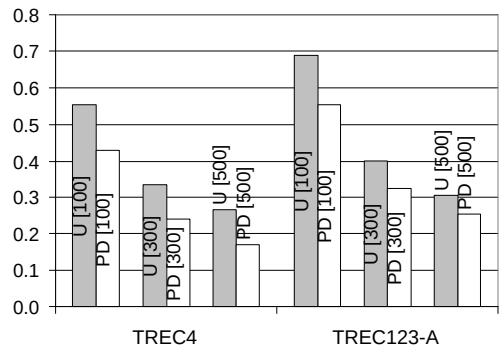


Figure 3: Impact of Increasing Sample Documents S [JS-Div]

both the uniform and quality-conscious sampling schemes result in higher quality samples, since more of each database may be sampled. We note that in all cases, the PD and PV schemes outperform the uniform approach, even when the total sample document allocation is significantly limited and the sampling framework must rely on even smaller seed samples for estimating the database size.

Second, we study the impact of the total allocation to the Seed Sampling step (S_{seed}) versus the dynamic sampling step (S_{dyn}), where recall that $S = S_{seed} + S_{dyn}$. Devoting too many sampling documents for seed sampling may result in more precise estimates for use by each quality-conscious sampling scheme, but leave too few documents available for dynamic sampling. Conversely, devoting too few sampling documents for seed sampling may result in less precise parameter estimates, and hence may lead to the sampling scheme misallocating the remaining documents.

We consider three scenarios – in Scenario 1, we collect a seed sample of 50 documents from each database ($PD[50, 250]$), leaving $250 \cdot n$ documents available for dynamic sampling; in Scenario 2, we collect a seed sample of 150 documents from each database, leaving $150 \cdot n$ documents available for dynamic sampling ($PD[150, 150]$); in Scenario 3, we collect a seed sample of 250 documents, leaving only $50 \cdot n$ documents available for dynamic sampling ($PD[250, 50]$). For comparison, we also consider the uniform sampling approach U. Interestingly, the PD and PV schemes result in higher quality samples in all cases. As S_{seed} increases, the advantage of the quality-conscious schemes is diminished only slightly relative to the uniform approach. Even when almost all of the total resources are allocated for seed sampling ($PD[250, 50]$) and the dynamic sampling has only $1/6$ of the total resources, the adaptive sampling approach still significantly outperforms the uniform approach.

Finally, we have also studied the impact of the number of rounds on the multiple iteration distributed query-sampling framework. Interestingly, we find that increasing the number of rounds from 1 to 10 results in slight improvements to the extracted database samples primarily due to the refined parameter estimates made possible by re-calculating the appropriate allocation after each round. Due to the space constraint, we omit these results here.

4.3 Application Scenario: Database Selection

In this section, we evaluate the impact of our adaptive sampling framework on the real-world application of database selection. Current approaches for text database selection map queries to databases based on previously acquired metadata for each database.

Typical database selection algorithms work in the context of a query q and a set of candidate databases $\mathcal{U}$. For each database $D \in \mathcal{U}$, a goodness (or quality) score is assigned in terms of query relevance. The databases are ranked according to the relevance score and the query is then routed to the top-k ranked databases. We consider the popular CORI algorithm as introduced in [6] and described in [9]. The quality of such an algorithm will be impacted by the quality of the frequency and count estimates generated by the document samples from the sampling process.

For the TREC123, A, B, and C datasets, we use queries drawn from the TREC topics 51-100 title field. These queries are, on average fairly short (ranging from 1 to 11 words, with an average of 3.8), and resemble web-style keyword queries. For the TREC4 dataset, we use queries from the TREC topics 201-250 description field (ranging from 8 to 33 words, with an average of 16 words).

To isolate the quality of database selection from the rest of the distributed information retrieval problem (which also includes components for results merging and ranking), we adopt a commonly accepted criterion for measuring the database selection recall. The database selection recall metric, denoted by R_n, evaluates the quality of the database selection algorithm's ranked list of databases versus a baseline ranking [11]. If we let $rel(q, D)$ denote the number of relevant documents in database D to the query q, then for a baseline ranking of n databases: $B = (B_1, ..., B_n)$ and a ranking induced by the database selection algorithm $E = (E_1, ..., E_n)$, we may define the recall for a particular query q as:

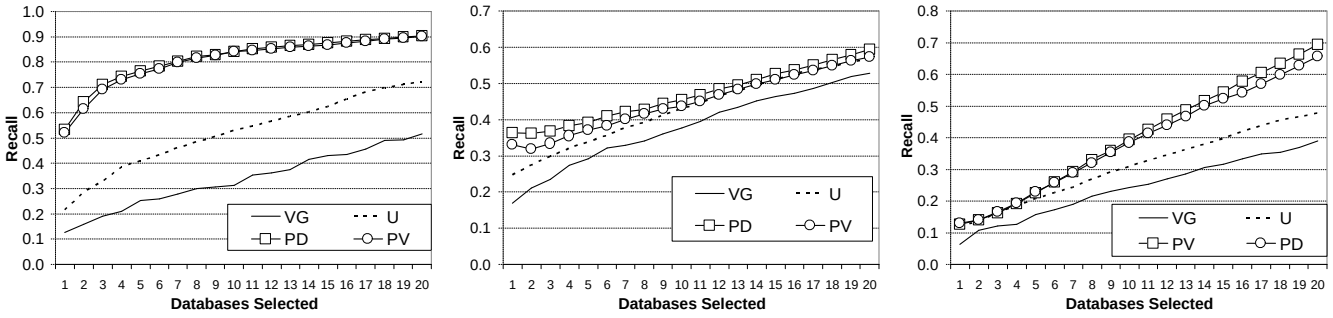
$$R_k(q) = \frac{\sum_{i=1}^{k} rel(q, E_i)}{\sum_{i=1}^{k} rel(q, B_i)}$$

where $0 \leq R_k(q) \leq 1$. By evaluating $R_k(q)$ for different values of k, we may assess the recall at different levels (e.g., recall for the top-5 databases, the top-10, and so on). A database selection algorithm that induces a ranking that exactly matches the baseline (optimal) ranking, will result in R_k values of 1 for all choices of k. We adopt a commonly-accepted baseline ranking that ranks databases by the number of query-relevant documents each has (note that the TREC data includes relevance decisions).

Finally, we run experiments on the database samples using the setup described in Section 4.2.2 to compare our adaptive sampling framework with the uniform sample allocation scheme. For each dataset, we evaluated the CORI algorithm over the extracted database samples and the query mix discussed above. In Figure 4, we report the database recall metric for the TREC123-A, B, and C datasets. These results confirm that the higher quality PV and PD samples reported in the earlier set of experiments positively impact the performance of database selection relative to the uniform approach, and again, the VG scheme significantly lags. We see similar, though less pronounced, results over the TREC4 and TREC123 datasets; due to the space constraint, we omit these results here.

5. RELATED WORK

In addition to the related work cited elsewhere in this paper, there have been a number of other studies that have relied on *sampling* a database, including [7, 15, 19, 27]. The sampling approaches typically rely on interacting with the database through a query interface and extracting sample data through a series of query probes. Querying methods suggested include the use of random queries, queries learned from a classifier, and queries based on a feedback cycle between the query and the response. To assess the quality of query-based database sampling techniques, others have developed a formal reachability graph model [1]. In contrast to the sampling-based approach, other researchers have studied the problem of downloading the entire contents of a Web-based text database using only a query-based mechanism for extracting documents [21], which we



(a) TREC123-A (b) TREC123-B (c) TREC123-C

Figure 4: Database Selection Recall: Comparing the Three Quality-Conscious Sampling Schemes to the Uniform Approach

have noted may be infeasible or very expensive in many realistic settings.

In the database and IR communities, considerable research has been dedicated to the *database (or resource) selection* problem (e.g., [8, 9, 10, 11, 23, 26]). We are interested in testing the database samples extracted through the adaptive framework over some of the database selection algorithms previously introduced.

6. CONCLUSION

To the best of our knowledge, the proposed *adaptive distributed query-sampling framework* is the first one for sampling that takes into account both the overall quality of all text databases under consideration *and* the presence of realistic resource constraints. We have introduced three sample allocation schemes for estimating database quality, and have shown how the adaptive framework supports higher-quality document sampling than existing solutions, and how database selection may be improved.

Our research on distributed query sampling continues along a number of directions. We are extending the framework to consider sampling costs that may vary across databases, as well as incorporating utility-theoretic models for determining the total number of possible sample documents S. We are also interested in alternative schemes that consider a richer set of database-specific quality metrics like data freshness and topic-sensitive database coverage.

7. REFERENCES

[1] E. Agichtein, P. Ipeirotis, and L. Gravano. Modeling query-based access to text databases. In *WebDB*, 2003.

[2] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[3] J. Callan and M. Connell. Query-based sampling of text databases. *Information Systems*, 19(2):97–130, 2001.

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

[5] J. Callan et al. The effects of query-based sampling on automatic database selection algorithms. Technical Report CMU-LTI-00-162, CMU, 2000.

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

[7] W. W. Cohen and Y. Singer. Learning to query the Web. In *AAAI Workshop on Internet-Based Info. Systems*. 1996.

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

[9] J. C. French et al. Comparing the performance of database selection algorithms. In *SIGIR*, 1999.

[10] N. Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM TOIS*, 17(3):229–229, 1999.

[11] L. Gravano and H. García-Molina. Generalizing GlOSS to vector-space databases and broker hierarchies. In *VLDB*, 1995.

[12] D. Hawking and P. Thomas. Server selection methods in hybrid portal search. In *SIGIR*, 2005.

[13] P. Ipeirotis et al. Modeling and managing content changes in text databases. In *ICDE*, 2005.

[14] P. Ipeirotis and L. Gravano. Improving text database selection using shrinkage. In *SIGMOD*, 2004.

[15] P. Ipeirotis, L. Gravano, and M. Sahami. Probe, count, and classify: Categorizing hidden-web databases. In *SIGMOD*, 2001.

[16] J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. on Inf. Theory*, 37(1):145–151, 1991.

[17] K.-L. Liu, C. Yu, and W. Meng. Discovering the representative of a search engine. In *CIKM*, 2001.

[18] J. Lu and J. Callan. Federated search of text-based digital libraries hierarchical peer-to-peer networks. In *SIGIR Workshop on P2P Information Retrieval*, 2004.

[19] W. Meng, C. T. Yu, and K.-L. Liu. Detection of heterogeneities in a multiple text database environment. In *CoopIS*, 1999.

[20] H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *SIGIR*, 2003.

[21] A. Ntoulas, P. Zerfos, and J. Cho. Downloading textual hidden Web content through keyword queries. In *JCDL*, 2005.

[22] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[23] A. L. Powell et al. The impact of database selection on distributed searching. In *SIGIR*, 2000.

[24] S. Raghavan and H. Garcia-Molina. Crawling the hidden Web. In *VLDB*, 2001.

[25] L. Si and J. Callan. Using sampled data and regression to merge search engine results. In *SIGIR*, 2002.

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

[27] W. Wang, W. Meng, and C. Yu. Concept hierarchy based text database categorization in a metasearch engine. In *WISE '00*.

[28] J. Xu and J. Callan. Effective retrieval with distributed collections. In *SIGIR*, 1998.