Evaluating Sampling Methods for Uncooperative Collections

Paul Thomas
Department of Computer Science
Australian National University
Canberra, Australia
paul.thomas@anu.edu.au

David Hawking
CSIRO ICT Centre
Canberra, Australia
david.hawking@acm.org

ABSTRACT

Many server selection methods suitable for distributed information retrieval applications rely, in the absence of cooperation, on the availability of unbiased samples of documents from the constituent collections. We describe a number of sampling methods which depend only on the normal query-response mechanism of the applicable search facilities. We evaluate these methods on a number of collections typical of a personal metasearch application. Results demonstrate that biases exist for all methods, particularly toward longer documents, and that in some cases these biases can be reduced but not eliminated by choice of parameters.

We also introduce a new sampling technique, "multiple queries", which produces samples of similar quality to the best current techniques but with significantly reduced cost.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*performance evaluation*

General Terms

Experimentation, Measurement

Keywords

Distributed information retrieval, random sampling

1. INTRODUCTION

Simultaneous search of documents from several independent collections presents challenges. In many cases, it is not possible to create a single central index: access restrictions, privacy concerns, or technical obstacles may mean the only interface to a collection is through a search engine. Distributed information retrieval (DIR)[1] aims to provide a unified search service spanning the combined collections of these independent search engines. Figure 1 illustrates one application: a personal DIR tool capable of searching all a user's digital collections including email, calendars, and corporate databases.

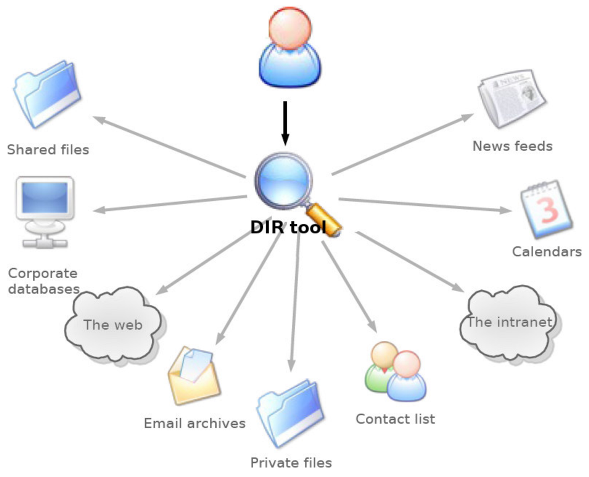

Figure 1: A personal DIR tool.

Besides providing a single entry point to multiple collections, and therefore enabling greater coverage than any single search system, such a tool would have several advantages. It can scale to large sizes with low cost, and using local search engines to process queries for their own collections allows collection-specific optimisations such as thesauri. Further, there may be significant advantages to users including a need to learn only one search interface, reduced cognitive load, and a reduced chance of certain errors.

In the most general case, we assume that search engines do not cooperate with any DIR framework and provide only the most minimal interface: they simply accept a query and produce a set of document identifiers as a result. In particular, they do not make available statistics such as the number of documents indexed, term frequencies, or weights; nor do they expose any information on their internal workings. This information is however needed for fundamental DIR tasks such as estimating overlap between collections [3], selecting a collection to answer a query [11], and merging result sets from multiple collections [6], and therefore techniques have been developed for estimating collection statistics based on samples of documents from constituent servers.

[1]DIR is also referred to as "metasearch" or "federated search".

These estimation techniques are improved by or explicitly rely upon random (unbiased) samples. Overlap estimates [3], as well as the standard capture-recapture [9] and sample-resample [14] techniques for estimating collection sizes require random samples as input; biased samples lead to systematic error in predicting overlap and systematic underestimates of collection size. The multiple capture-recapture and capture history techniques [13] also assume random samples, although estimates from these techniques are adjusted to account for sample quality, as does the random documents method [4].

Early evidence also suggests that biased samples, and hence biased term statistics and size estimates, have a negative impact on standard server selection algorithms including CORI [5], ReDDE [14], and KL-divergence [15].

1.1 Random samples

Obtaining a random sample from an uncooperative search engine is a non-trivial task. With a limited interface, it is not possible to enumerate documents in the collection or to retrieve them according to some identifier; therefore we cannot simply take a uniform sample. Further, characteristics of the search engine itself which may improve performance for typical uses are likely to make sampling less convenient. For example, certain documents may be more likely to be returned since they are considered in some sense more important, or more useful; this introduces strong bias. Similarly, result lists may be arbitarily truncated, or near-duplicates removed, to save work on the server or to present a more useful list to a client. Any random sampling technique will have to work in this environment.

Ideally, for DIR we would like a sampling technique which produces samples with as little bias as possible; which requires as little run-time or pre-processing resources as possible; and which works over a wide range of collections and search engines with as little prior knowledge as possible. In particular, we do not want to rely on particulars of document format such as the availability of hyperlinks.

In this work we consider sampling techniques against these criteria. A number of candidate techniques have been developed for sampling from the Web, some of which are more generally applicable, and we consider the runtime cost and performance of these. To the best of our knowledge this is the first statistical evaluation of "random" document samples from these techniques. We also consider a new method, "multiple queries", which provides samples of similar quality to existing methods with much reduced cost. Finally, we consider the effects of the parameters available in each technique.

1.2 Notation

In the discussion which follows we use notation following that of Bar-Yossef and Gurevich [2]: $\mathcal{D}$ is the set of all documents available through a search engine, d an individual document from $\mathcal{D}$, and $N = |\mathcal{D}|$ the number of documents a search engine provides access to. $\overline{x}$ represents the mean of some value x.

For each query q sent to a search engine, $\textsc{res}(q)$ denotes the results returned. This result set may be constrained by a limit k, imposed either by the search engine itself or by our samplers; if $|\textsc{res}(q)| \geq k$ we say that q "overflows", and if $|\textsc{res}(q)| = 0$ we say q "underflows"[2].

[2]Bar-Yossef and Gurevich refer to overflow if $|\textsc{res}(q)| > k$. We prefer the formulation here as, without extra information from a search engine, it is not generally possible to tell whether a result set is bounded by k or by the number of matches. The two definitions are interchangable by incrementing or decrementing k.

2. RELATED WORK

The problem of sampling from an uncooperative collection is very similar to that of sampling from the Web, and several algorithms have been introduced for the latter. These are summarised in Table 1.

2.1 General methods

"General" methods do not rely on link structures between documents, and are applicable across a wide variety of document types.

Single queries

Bharat and Broder [3] introduced a simple technique for sampling random pages from Web search engines. The technique does not however rely on any particular characteristic of the Web or of Web pages and is applicable to a variety of collections. The algorithm is extremely simple: a single query is constructed and sent to a search engine, and a sample document is chosen at random from the set of matches returned.

The application to Web search engines, which typically return a small number of results regardless of the number of possible matches, prevents us from dealing with large result sets. As a work-around, Bharat and Broder generated queries which they expected to return between one and 100 documents, although if a query matched more than 100 documents they used only the top 100 returned. Query terms came from a lexicon built during an earlier crawl, and queries were either four-term disjuncts or two-term conjuncts with terms chosen for their frequency.

(In a later paper [7], Gulli and Signorini used this method with slight modifications to estimate the size of public search engines and hence the public Web. Query terms again came from an earlier crawl, but Gulli and Signorini used single-term queries in more than 75 languages.)

Although working with an uncontrolled, dynamic corpus meant Bharat and Broder were not able to investigate the quality of the sampler, they identified six sources of bias. Two of these are relevant to the DIR case. "Query bias" is the bias towards longer, content-rich documents which are more likely to match the queries used. "Ranking bias" is the result of search engines ranking documents and a sampler not seeing those past rank k. By choosing queries with a smaller number of results, they note that it is possible to eliminate ranking bias at the expense of increasing query bias. Query bias has been noted in other work [1, 2] and is confirmed by our results below.

In our implementation of the single queries sampler, we are able to eliminate ranking bias by ignoring any queries that underflow or overflow and only choosing from result sets where $1 \leq |\textsc{res}(q)| < k$.

Pool-based sampling

Rejection sampling is a Monte Carlo method which can be used to simulate sampling according to one distribution π (for example, the uniform distribution) when it is only feasible to draw samples with some other distribution p (such as that resulting from query or ranking biases). Bar-Yossef

	Queries needed	Docs/ run	Hyperlink graph	Doc text	Parameters
General methods					
Single queries	Stream	1	—	—	Docs/query
Pool-based	Pool	1	—	Needed	Docs/query
Random walk on MATCH$_{\mathcal{P}+}$	—	1	—	Needed	Seed, docs/query, burn-in period, query space
Multiple queries	Stream	Any	—	—	Docs/query, queries/sample, docs/sample
Hyperlink methods					
PAGERANK-SAMPLE	—	Any	Needed	—	Seed, walk length
WebWalker	—	1+	Needed	Needed	Seed, walk length
(UN)DIRECTED-SAMPLE	—	1+	Needed	—	Seed, walk length

Table 1: Characteristics of the sampling methods under investigation. "General" methods do not require a hyperlink graph and are examined in our experiments.

and Gurevich introduced an application of this technique to the problem of sampling Web pages [2]. The algorithm consists of an inner round of rejection sampling, which chooses a query according to the size of its result set; and an outer round, which chooses a document from the result set of this query.

Pool-based sampling requires as input a "query pool" $\mathcal{P}$, a set of queries drawn from all possible queries a search engine accepts. Ideally queries in $\mathcal{P}$ have high recall (meaning that taken together, they cover a large proportion of the documents in $\mathcal{D}$), and simultaneously a low probability of underflow or overflow. We use $\mathcal{P}+$ to denote the subset of $\mathcal{P}$ which neither underflows nor overflows.

We also use $\text{MATCH}_{\mathcal{P}}(d)$ to denote the queries which a document d matches, from amongst a pool of queries $\mathcal{P}$. $\text{MATCH}_{\mathcal{P}}(d)$ can be calculated from the text of a document — for example, if $\mathcal{P}$ is the set of all 3-gram queries, it suffices to extract text from d and enumerate all 3-term phrases in the document.

The inner loop uses rejection sampling to choose a query q from $\mathcal{P}+$ with a distribution based on $|\text{RES}(q)|$. A query is chosen uniformly from the pool and forwarded to the search engine; if it neither underflows nor overflows we accept it with probability $|\text{RES}(q)|/k$ and return to the outer loop[3].

In the outer loop, a candidate document is chosen uniformly from $\text{RES}(q)$. At this point the probability of choosing a document d as a candidate is proportional to the probability that d matches the query; in other words to $|\text{MATCH}_{\mathcal{P}+}(d)|$. A second round of rejection sampling therefore returns d as a sample with probability $1/|\text{MATCH}_{\mathcal{P}}(d)|$, and otherwise iterates selecting another query and document. (Note that although documents are presented as candidates with a distribution based on $|\text{MATCH}_{\mathcal{P}+}(d)|$, they are selected as samples based on $1/|\text{MATCH}_{\mathcal{P}}(d)|$. This introduces an error the size of which depends on the difference between $\mathcal{P}$ and $\mathcal{P}+$, which is hard to determine without detailed knowledge of how queries and documents are processed at the search engine.)

Experiments to characterise possible pools $\mathcal{P}$ used text from a crawl of web pages in the Open Directory Project (ODP) [10]. Bar-Yossef and Gurevich used a crawl of a subset of the ODP as a source of terms and considered single terms, 3-, 5-, and 7-term phrases for recall, underflow, overflow, and other properties; 5-term phrases were considered the best tradeoff in this instance, and we use 5-term phrases in our initial experiments below.

[3]The original description of the algorithm leaves the parameters C and $\hat{\phi}(q)$, the envelope constant and the unnormalised sample distribution, unspecified. In our implementation we let $\phi(q)$ be uniform, use an unnormalised version $\hat{\phi}(q) = 1$ for all q, and set $C = k/\hat{\phi}(q) = k$.

Random walk on $\text{MATCH}_{\mathcal{P}+}$

A further variant on sampling through random walks was introduced at the same time as pool-based sampling [2]. This variant uses the Metropolis-Hastings algorithm, which carries out a random walk with each step chosen according to a "proposal function", and before each step employs an acceptance/rejection procedure to determine whether or not the step will be taken. If the proposal function satisfies certain simple criteria, a walk with the Metropolis-Hastings algorithm will eventually converge on a desired distribution. The parameter B, the burn-in period, determines the number of (possible) steps in the walk: as with other random walk methods, this can only be set empirically without prior knowledge of the collection.

Bar-Yossef and Gurevich adapt this algorithm to collections without a hyperlink structure by defining a graph such that two documents are joined by an edge iff they both match at least one query: i.e. an edge exists between two documents x and y iff $\text{MATCH}_{\mathcal{P}+}(x) \cap \text{MATCH}_{\mathcal{P}+}(y) \neq \emptyset$. Given a document d, the sampler then proceeds by choosing a query uniformly from $\text{MATCH}_{\mathcal{P}+}(d)$ (which determines a subset of edges from the current document); choosing a document d' uniformly from the results of this query (which determines an individual edge); and then choosing to follow the edge and set $d \leftarrow d'$ with probability $\frac{|\text{MATCH}_{\mathcal{P}+}(d)|}{|\text{MATCH}_{\mathcal{P}+}(d')|}$ (which normalises for the number of queries d' matches). After B iterations, the current document is returned as the sample.

Evaluation experiments compared this random walk with the pool based and single query samplers on a testbed of 2.4 million documents from the ODP. With both the pool based and random walk samplers, "little or no" bias was seen due to document size, and no "significant bias" due to the static document rank used by their search sytem. As with other methods, to the best of our knowledge no quantitative tests for bias have been performed.

Unlike other variants on random walks, a random walk on $\text{MATCH}_{\mathcal{P}+}$ does not require an explicit hyperlink structure and is appropriate to a wide range of collections. We therefore include this sampler in our experiments below.

2.2 Hyperlink methods

A further set of random-walk sampling methods assume documents are linked in a graph, such as a web graph. Notable among these are the PAGERANK-SAMPLE method of Henzinger et al. [8], the WebWalker algorithm of Bar-Yossef et al. [1] and the DIRECTED-SAMPLE and UNDIRECTED-SAMPLE methods of Rusmevichientong et al. [12].

These methods are not generally applicable in the DIR case, since not all constituent corpora will have such a structure. Many component collections typical of personal DIR, such as email, databases, calendars and catalogues are not hyperlinked. Accordingly, we have excluded these methods from our experiments.

3. MULTIPLE QUERIES

The multiple queries sampler is a straightforward extension of the single queries sampler. To reduce query bias, we run several queries with a large cutoff k; we then choose any number of documents from the union of all result sets. (Note that if a document is returned in reponse to more than one query, we record it only once. This has a similar effect to adjusting for visit frequency in PAGERANK-SAMPLE.) We choose queries from a pool defined independently of the collection, with as high a recall as possible, and as with other samplers ignore any which under- or overflow.

The high values of k (10,000 in our initial experiments) suggest a large amount of network traffic; however since we do not need to download the text of documents, and can sample many documents in a single run, this traffic does not seem excessive. Further, although we rely on search engines providing large result sets we do not rely on them making document text available. Experiments varying k and queries used are described in Section 6.2 below.

4. SAMPLING COST

A desideratum for a working DIR system is that samplers should require as few resources as possible. The most significant resource is communication with the search engines we are sampling, and in this section we consider the cost for each document sampled in the number of queries issued and in the number of documents downloaded and parsed.

The PAGERANK-SAMPLE, WebWalker, and (UN)DIRECTED-SAMPLE algorithms rely on explicit hyperlinks between documents and are not applicable to most collections likely to be used in DIR. In the remainder of this paper we therefore consider only the single queries, pool based, random walk, and multiple queries samplers.

For each of the following analyses a first step is to determine how many queries successfully complete (i.e. neither underflow nor overflow). Following Bar-Yossef and Gurevich we refer to the "validity density" of a document, $\text{VDENSITY}(d)$: this is just $|\text{MATCH}_{\mathcal{P}+}(d)|/|\text{MATCH}_{\mathcal{P}}(d)|$, or the proportion of those queries d matches which return $1 \dots k-1$ results. Similarly, we use $\text{VDENSITY}(\mathcal{P})$ to represent the proportion of queries in a pool $\mathcal{P}$ which neither underflow nor overflow: $\text{VDENSITY}(\mathcal{P}) = |\mathcal{P}+|/|\mathcal{P}|$.

4.1 Single queries

In its original form, the cost of the single queries sampler is a single query per document sampled for any collection and any source of queries. In the implementation used here, we first must find a query which neither underflows nor overflows; the number of queries we will issue is geometrically distributed with $p_1 = \text{VDENSITY}(\mathcal{P})$, so we expect to issue $1/\text{VDENSITY}(\mathcal{P})$ queries per document sampled.

In both implementations, there are no documents downloaded or parsed.

4.2 Pool-based sampling

We consider the inner round of rejection sampling (to find a query) and the outer round (to find a document) seperately.

In the inner round, the number of iterations needed to first find a valid query and then select it is geometrically distributed with $p_1 = \left(\overline{|\text{RES}(q)|}/k\right) \text{VDENSITY}(\mathcal{P})$; so the expected number of queries issued before one is selected is $E_c = \left(k/\overline{|\text{RES}(q)|}\right)\left(1/\text{VDENSITY}(\mathcal{P})\right)$.

In the outer round, having selected a query a document from the result set is downloaded and considered for the final sample. The chance of sampling a document after each iteration is $1/|\text{MATCH}_{\mathcal{P}}(d)|$, so the number of queries selected before a document is selected is geometrically distributed with $E_s = \overline{|\text{MATCH}_{\mathcal{P}}(d)|}$.

The expected number of queries executed per document sampled is therefore

$$\begin{aligned} E_q &= E_c \times E_s \\ &= \left(k/\overline{|\text{RES}(q)|}\right)\left(1/\text{VDENSITY}(\mathcal{P})\right)\overline{|\text{MATCH}_{\mathcal{P}}(d)|} \end{aligned}$$

We also expect to download and parse $\overline{|\text{MATCH}_{\mathcal{P}}(d)|}$ documents per sample.

4.3 Random walk on $\text{MATCH}_{\mathcal{P}+}$

As with the single queries sampler above, the expected number of queries we must issue before finding one which neither underflows nor overflows is $1/\text{VDENSITY}(\mathcal{P})$. After each such query is found we download and parse a document, and we repeat the process B times (recall that B is the burn-in time of the random walk). We can therefore expect to need $B/\text{VDENSITY}(\mathcal{P})$ queries and B downloads per document sampled, although with more knowledge of the query pool and collection it is possible to collect second and subsequent documents more cheaply by continuing the random walk.

Since the cost depends on B, a parameter of our choosing, we can choose to improve runtime at the expense of the randomness of our sample.

4.4 Multiple queries

The cost of the multiple queries sampler is determined by the number of queries per sample s_q, the number of documents per sample s_d, and the pool validity density $\text{VDENSITY}(\mathcal{P})$. Since we need s_q queries which neither underflow nor overflow, from which we select s_d documents, we expect $(s_q/s_d)\,/\,(1/\text{VDENSITY}(\mathcal{P}))$ queries per document sampled and no downloads.

5. EXPERIMENTS

We have carried out a number of experiments to explore two questions: first, do the sampling techniques described above provide unbiased samples across a range of collections? Secondly, what effect do the available parameters have on performance?

		Size (terms)			
Collection	Docs	Range	Mean	Std dev	Topics
calendar	1k	1–20	4	2	Mixed
zsh-list	9k	2–59k	176	179	Narrow
procmail	24k	2–14k	207	215	Narrow
email	25k	1–26k	199	295	Mixed
WSJ	99k	9–10k	462	450	Broad
.GOV	1.2M	0–43k	6803	5720	Broad

Table 2: Summary statistics of collections used in our experiments.

5.1 Collections

Our six collections, summarised in Table 2, represent a range of sizes (over three orders of magnitude), data types, and topic skew which are likely to be characteristic of personal DIR and personal information management (PIM) applications.

The "calendar" collection contains 1049 documents (appointments) from a calendar application, spanning about two years. Documents are typically sentence fragments, only a few terms long, and the terms used in each document have little overlap with others.

The "zsh-list" and "procmail" collections represent the archives of two public mailing lists, both on narrow technical topics. "Email" is a third email collection, this time of documents from a personal email archive with much broader topics.

"WSJ" (from TREC CD 1) collects several years of contents of the Wall Street Journal, including articles and letters. It covers a broad range of topics, and document length varies widely.

The largest collection, ".GOV" (from the TREC Web Track), is a 1.25M page partial crawl of Web hosts in the `.gov` top-level domain (US government agencies). As with the WSJ collection, document size, style, and topics vary considerably.

5.2 Parameter settings

In the first instance, we have run each sampler with parameters set as originally described. For the single queries sampler, samples were taken with $k = 100$ and with queries from a 1% subset of terms in each collection. (If all queries were exhausted in the course of a long run, two- and finally three-term disjuncts were used.)

The pool-based sampler was run with a limit $k = 5$, and a query pool of a 1% subset of 5-grams from each collection (for .GOV, a 0.1% subset was taken as the collection is large). The random walk sampler used the same parameters, although we were able to use a pool of all possible 5-grams since in this instance queries do not need to be enumerated in advance.

The multiple queries sampler used single-term queries, with terms chosen independently of the collection from a list of common English words. We used 100 queries per sample and sampled twenty documents at a time with a result limit $k = 10{,}000$.

The estimated cost of each sampler with these settings is summarised in Table 3. The random walk sampler is significantly more expensive than others; however this cost scales with B, the burn-in time. We experiment with lower burn-in times in Section 6.2 below. Costs for the pool-based and multiple queries sampler are similar, although the multiple queries sampler generally requires fewer interactions and never requires document text.

	Single queries		Pool based		Random walk		Multiple queries	
Collection	q	d	q	d	q	d	q	d
calendar	1	0	1	1	1000	1000	34	0
zsh-list	1	0	7	2	1123	1000	5	0
procmail	1	0	8	2	1137	1000	6	0
email	1	0	8	2	1173	1000	5	0
WSJ	1	0	25	4	1161	1000	8	0
.GOV	2	0	11	1	2140	1000	23	0

Table 3: Estimated cost in queries ("q") and document downloads ("d"), per document sampled, for each sampler.

5.3 Tests for bias

Two simple tests were used to indicate bias in samples.

"T", number of times seen If from a collection of size N we draw i independent samples, each of size n, and each sample is drawn randomly, then the probability of seeing a document t times in the i samples follows a binomial distribution: $\Pr(\text{seen } t \text{ times}) = \binom{i}{t}(\frac{n}{N})^t(\frac{N-n}{N})^{i-t}$. We used a χ^2 test to compare this expected distribution with observed frequencies from large numbers of samples.

Failure of this test would most likely indicate that some individual documents are being sampled too frequently and others too infrequently — in other words that there is a bias towards some subset of the documents.

"S", size distribution Test "S" considers one likely source of bias. Earlier work has suggested that the single query sampler, in particular, strongly favours longer documents. This could be due to two factors: a ranking bias, if search engines promote longer documents, or a query bias, since longer documents are more likely to contain our chosen search terms. The first factor is controlled in our samplers, but we may still expect to see effects of the second.

To investigate, we divided each collection into deciles according to document length, and recorded the proportion of documents sampled from each decile. A random sample should have documents from all deciles equally represented. As for test "T", we use a χ^2 test ($df = 9$) for comparisons. Failure of this test indicates a bias towards documents of particular lengths; in practice this tends to be towards longer documents.

5.4 Exploring parameter settings

Informed by the results of our initial tests, a series of experiments investigated the effects of the available parameters for each sampler. These are described in more detail in Section 6.2 below.

6. RESULTS

Results showed samplers performed well with only some collections, and that all were subject to some degree of query bias.

Altering query pools or other parameters did not provide improvements.

t	Expected	Single queries	Pool based	Random walk	Multiple queries
0	24381	24540	24389	24381	24378
1	586	333	570	586	592
2	7	63	15	7	4
3		27			
4		4			
5		3			
6		1			
7		1			
8		2			
p		<u>0.00</u>	<u>0.01</u>	1.00	0.54

Table 4: Email samples for test "T". $i = 30$ samples of $n = 20$ documents each where possible; $i = 600$ and $n = 1$ otherwise. $N = 24,974$ documents.

	Single queries	Pool based	Random walk	Multiple queries
calendar	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>
zsh-list	<u>0.00</u>	<u>0.02</u>	0.51	1.00
procmail	<u>0.00</u>	0.34	0.90	1.00
email	<u>0.00</u>	<u>0.01</u>	1.00	0.54
WSJ	<u>0.00</u>	0.90	0.42	0.92
.GOV	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>

Table 5: χ^2 results (p values) for test "T". Significant deviations from randomness (i.e. probable non-uniform samples) are underlined.

6.1 Tests for bias

Test "T", number of times seen Table 4 summarises the number of times each document in the email collection was returned by each sampler, and the theoretical distribution described in Section 5.3. p is the chance of seeing a distribution this far from the expected if we had a truly uniform sample (χ^2 test, $df = 2$ with $t \geq 2$ counted as one category). We use the email collection for illustration; similar trends were seen with the other collections tested.

In this example, the single queries sampler has returned fewer documents than we would expect: 434 unique documents are returned whereas we would expect 593 from a truly random sample. Further, those documents which are sampled are returned too frequently: 63 documents are seen twice (we expect only seven), and two documents are seen eight times each. Together these observations suggest a bias towards a subset of the collection. (We examine one form of this bias below.)

Table 5 summarises test "T" across each collection and sampler. In every case of test "T" failing, the sampler has sampled too few documents too frequently, suggesting a bias towards some part of the collection. This bias is most extreme with the calendar collection, on which every sampler failed: the random walk sampler returned one document 24 times and the pool based sampler returned seven distinct documents more than 15 times each.

We surmise the failures on the calendar collection are due to the unusual combination of document length and document language. Since documents are very short, the graph on MATCH$_{\mathcal{P}+}$ used by the random walk sampler is much

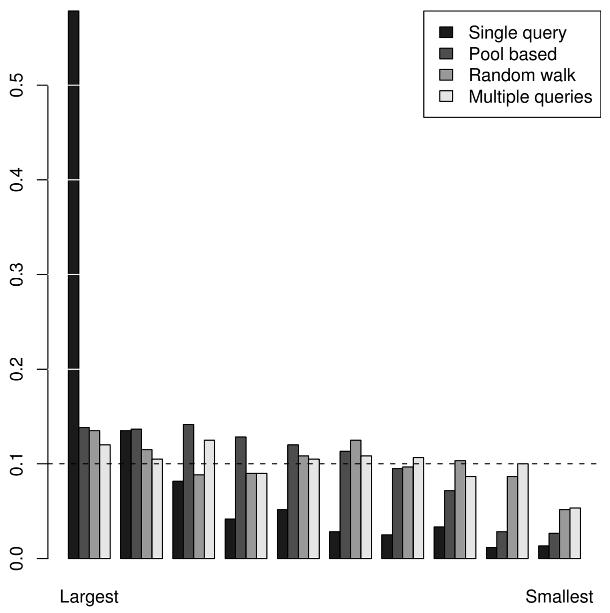

Figure 2: Email samples by size decile for test "S". A uniform sample would have 10% of sampled documents in each decile.

	Single queries	Pool based	Random walk	Multiple queries
calendar	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>
zsh-list	<u>0.00</u>	<u>0.00</u>	0.46	0.94
procmail	<u>0.00</u>	<u>0.00</u>	0.41	0.16
email	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>	<u>0.01</u>
WSJ	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>
.GOV	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>

Table 6: χ^2 results (p values) for test "S". Significant deviations from randomness (i.e. probable non-uniform samples) are underlined.

sparser and there is less opportunity to move from the initial document. For other samplers, we would expect to see strong bias towards the small fraction of documents matching the query pool since the terms used in each document vary considerably.

Test "S", size distribution Figure 2 illustrates the distribution of documents sampled by each algorithm from the email collection. A truly random sample would include around 10% from each decile; instead a bias is apparent in all samplers towards larger documents. Table 6 has χ^2 results ($df = 9$) for all collections and samplers.

In the great majority of cases, a failed test is due to a bias towards longer documents. This is likely due to query bias; a longer document is more likely to match any given query, and by ignoring overflowing queries the samplers may be effectively counteracting any length normalisation at the search engine.

As may be expected, the single queries sampler performs very poorly on this test, failing with every collection with a very large bias towards longer documents. The pool based sampler also fails in all six cases, which we expect is due to the size of the query pool: a uniform subsample of 1% of 5-grams would cover a small proportion of documents and these documents would tend to be longer. If this conjecture is correct, a larger pool (although more difficult to construct) would result in less bias. We explored this possibility by varying query pools in experiments described below.

The random walk sampler fails in several cases again with a bias towards longer documents. It is not clear why this should be the case, since the sampler explicitly controls for the number of queries a document matches; however differences in computing $\text{MATCH}_{\mathcal{P}}(d)$ at the search engine and at our sampler may give rise to a bias of this nature. Since we must agree with an unknown search engine with each step of query processing, including tokenisation, stemming and stopping, and parsing queries, errors of this kind are very likely and may explain the observed bias.

The larger number of documents considered by the mutiple queries sampler provides some improvement on the single queries sampler, and for the zsh-list and procmail collections suffices to overcome query bias. In other cases a significant bias to longer documents remains, although we note this sampler performs no worse than the random walk sampler and with significantly lower runtime.

6.2 Exploring parameter settings

The biases described above may be due in part to the particular parameters chosen for each sampler. For each sampler, we varied the available parameters hoping to generate improvements.

Single queries The single queries sampler proved particularly susceptible to query bias. Since we ignore all queries which overflow, varying k will change the subset of queries used and may affect the quality of the resulting sample; so may varying the source of queries.

We ran tests "T" and "S" as above for the single queries sampler with $k = 100$, 500, and 10,000; and with queries chosen from a subsample of terms indexed, from a subsample of 5-grams, and from a collection-independent set of common English words. In each case the sampler performed poorly, again returning too few unique documents and with a strong bias towards longer documents. We conclude that the single queries sampler is unlikely to avoid bias for any combination of parameters.

Pool based In our initial experiments we observed a consistent bias in the pool based sampler towards longer documents, and surmised that this was due to our small query pool (1% of 5-grams for most collections). With a larger pool of 10% of 5-grams, results improved for both tests and most collections, which is consistent with this conjecture.

While a larger pool improves results, we generated pools by pre-processing the collections used. It is not clear how a working system might generate pools except from document text; since any bias in the pool will result in biased samples, documents for the pool should be returned by an unbiased sampler we are faced with a difficult bootstrapping problem.

Further experiments varied k, using the original pool of 1% of all indexed 5-grams. We observed improvements over $k = 5$ with $k = 50$, and further improvements with $k = 500$ across most collections. These samplers however have significantly increased runtime, as k increases much faster than $\overline{|\text{RES}(q)|}$.

Random walk The quality of the samples generated by the random walk sampler seem to depend strongly on B, the burn-in period of the walk. Our initial experiments used $B = 1000$, as originally specified, and followup experiments used values of $B = 100$ or 10. With smaller values of B we saw a marked increase in bias; this bias was more apparent with larger corpora. The choice of seed for the walk is strongly effected by query bias, so this result can be seen as a straightforward result of the sampler taking fewer steps from this initial biased choice. With some knowledge of the size of the corpus and the connectivity of the $\text{MATCH}_{\mathcal{P}}$ graph it would be possible to choose a minimal useful B, but it is not clear how this could be derived without prior knowledge.

Multiple queries As for the single queries sampler, the multiple queries sampler is sensitive to the value of k. Samples were taken with k set to 10,000, 1000, and 100; in general, results were somewhat worse with $k = 1000$ and deteriorated markedly with $k = 100$. This is expected: as k decreases, only queries which are more specific will not overflow. This leads to a smaller number of documents from which to take a final sample.

This observation may also explain variation seen with different query sources. With collection-specific query pools of 1% of terms, the multiple queries sampler showed more bias. We conclude that for our data — documents of varying size and format but all in English and most with full sentences — a list of common words is both more convenient and produces better samples.

7. CONCLUSIONS

Several key methods for DIR rely, explicitly or implicitly, on an unbiased sample of documents from constituent collections. Techniques for generating these samples, given only a query interface and an otherwise uncooperative search engine, must contend with biases due to search engine optimisations ("ranking bias") and document content ("query bias"). It appears possible to eliminate ranking bias but query bias is persistent across a variety of samplers and collections.

We have described seven samplers. Of these, four are applicable to document types without hyperlink structures and were tested across six collections representing a range of sizes and document types. No sampler performed well across all collections.

The single queries sampler is very badly effected by query bias, and consistently prefers longer documents across all collections tested. The pool based sampler as initially described performs poorly, but depends greatly on the choice of pool — with a larger sample of document text the samples generated are of higher quality. It is not clear, however, how a real-world system can generate such a query pool without prior knowledge of collection contents.

The random walk sampler performs better than most alternatives, but has a much higher cost as measured in interactions with the search engine. It also requires some knowledge of how queries are processed at the search engine; any errors in assumptions here are reflected in biased samples. Our multiple queries sampler can produce samples of comparable quality, with fewer interactions and no prior knowledge of the collection being sampled, but requires search engines to support large result sets.

Future work is needed in two areas. It remains to develop a sampling technique which is applicable across a range of collections and which requires little prior knowledge of collection contents. Further work also remains to quantify the impact improved samples have on standard methods for DIR.

8. REFERENCES

[1] Z. Bar-Yossef, A. Berg, S. Chien, J. Fackcharoenphol, and D. Weitz. Approximating aggregate queries about web pages via random walks. In *Proc. VLDB*, 2000.

[2] Z. Bar-Yossef and M. Gurevich. Random sampling from a search engine's index. In *Proc.WWW*, 2006.

[3] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public web search engines. In *Proc. 7th WWW*, 1998.

[4] A. Broder, M. Fontura, V. Josifivski, R. Kumar, R. Motwani, S. Nabar, R. Panigrahy, A. Tomkins, and Y. Xu. Estimating corpus size via queries. In *Proc.CIKM*, 2006.

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

[6] N. Craswell, D. Hawking, and P. Thistlewaite. Merging results from isolated search engines. In *Proc. Australasian Database Conference*, 1999.

[7] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Proc. WWW*, 2005. Poster.

[8] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. On near-uniform URL sampling. In *Proc. 9th WWW*, 2000.

[9] K.-L. Liu, A. Santoso, C. Yu, W. Meng, and C. Zhang. Discovering the representative of a search engine. In *Proc.CIKM*, 2001. Poster.

[10] Open directory project. `http://dmoz.org/`.

[11] A. L. Powell, J. C. French, J. Callan, M. Connell, and C. L. Viles. The impact of database selection on distributed searching. In *Proc. SIGIR*, 2000.

[12] P. Rusmevichientong, D. M. Pennock, S. Lawrence, and C. L. Giles. Methods for sampling pages uniformly from the world wide web. In *Proc. AAAI Fall Symposium on Using Uncertainty Within Computation*, 2001.

[13] M. Shokouhi, J. Zobel, F. Scholer, and S. M. M. Tahaghoghi. Capturing collection size for distributed non-cooperative retrieval. In *Proc. SIGIR*, 2006.

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

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