Query-Based Sampling: Can we do Better than Random?

Almer S. Tigelaar
Database Group,
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente, P.O. Box 217, 7500AE,
Enschede, The Netherlands
a.s.tigelaar@cs.utwente.nl

Djoerd Hiemstra
Database Group,
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente, P.O. Box 217, 7500AE,
Enschede, The Netherlands
hiemstra@cs.utwente.nl

ABSTRACT

Many servers on the web offer content that is only accessible via a search interface. These are part of the deep web. Using conventional crawling to index the content of these remote servers is impossible without some form of cooperation. Query-based sampling provides an alternative to crawling requiring no cooperation beyond a basic search interface. In this approach, conventionally, random queries are sent to a server to obtain a sample of documents of the underlying collection. The sample represents the entire server content. This representation is called a resource description. In this research we explore if better resource descriptions can be obtained by using alternative query construction strategies. The results indicate that randomly choosing queries from the vocabulary of sampled documents is indeed a good strategy. However, we show that, when sampling a large collection, using the least frequent terms in the sample yields a better resource description than using randomly chosen terms.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*retrieval models, search process, selection process*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed Systems*

General Terms

Design, Experimentation, Measurement, Performance

Keywords

distributed information retrieval, query-based sampling

1. INTRODUCTION

The surface web consists of static, easily indexable, pages. The deep web consists of content that is generated dynamically in response to user queries. These queries are sent through a search form or interface. The simplest incarnation of a search interface presents a single free-text search field. These interfaces commonly provide access to an underlying database, for example: a database with descriptions of movies, books or gadgets. The number of deep web pages that provide search services like this is estimated by one study to be five hundred times larger with respect to the amount of normal surface web pages [6].

The deep web is similar to the surface web: they both grow fast and offer a diverse range of information. However, there are some notable differences: the deep web is mostly structured and difficult to crawl. Some deep web sites expose direct links to the underlying content making indexing easier for traditional search engines. Despite this, the major search engines, Google, Yahoo and Bing, have major trouble indexing the deep web. A 2004 study shows that they typically index only 37% of the available deep content [10]. It has been suggested that a database-centered, discover-and-forward access model would likely be a better approach for enabling search of the deep web [16].

An existing conceptual model for this is *distributed information retrieval*. A distributed web search engine accepts user queries, forwards them to other remote search servers that index the actual content, merges the returned results and sends this back to the user. To know what remote search servers to forward a query to, the distributed web search engine requires a *resource description* of each server [7].

Query-based sampling can be used to discover the content available at a remote server. It makes only a minimal assumption: the remote server should be able to receive and process queries and send back a list of search results. No other functionality is necessary beyond what end users would use directly. Random, single-term, queries are sent to this remote search engine and the returned documents are fully downloaded locally to create a resource description of the server. This resource description is a language model: terms with occurrence counts. For example: $[(apple, 16), (pear, 23)]$. The vocabulary is defined as the terms without occurrence counts: *apple*, *pear*. The first query sent is conventionally a frequently occurring word in the (English) language obtained from an external resource, for example: a dictionary. The purpose of this first query is to retrieve some initial search results. Subsequent queries are terms drawn *randomly* from the vocabulary of the language model constructed so far. Thus, only the query sent in the first iteration is based on an external resource. The process of sending queries, downloading documents and updating the language model, iterates until a stopping criterion is reached. For example, when three hundred documents have been downloaded, or after hundred iterations [8].

This research focuses on the drawing of random terms in iterations *following* the first. We explore several alternative strategies to choose single-term queries based on the constructed language model. Our research question is:

> "Can query-based sampling be improved by using an alternative term selection strategy?"

2. RELATED WORK

The foundational work for acquiring resource descriptions via query-based sampling was done by Callan, et al. [9, 8, 13]. They showed that a small sample of several hundred documents can be used for obtaining a good quality resource description of large collections consisting of hundreds of thousands of documents. They used uniform random term selection, meaning each term in the vocabulary has an equal probability of being chosen, but suggest that query terms could be selected from the learned language model using other criteria. This is what this paper focuses on. The test collection used in their research, TREC-123, is not a web data collection. While this initially casts doubt on the applicability to the web, Monroe, et al. [23] showed that the query-based sampling approach also works very well for web data. Even when random queries are used.

Conventionally we compare the language model of the sample with that of the collection to assess the resource description quality. Sampling stops when a certain amount of documents has been obtained. Baillie, et al. [1] propose using the predictive likelihood as a stopping criterion for sampling, instead of a fixed number of documents. They use a set of reference queries that represent typical information needs. Performance is measured with respect to this set of reference user queries. This shifts the focus from similarities between language models to the expected real-world performance, given a representative query set.

A problem with sampling is that some documents are more likely to be sampled than others. For example: longer documents are more likely to be sampled than shorter ones. This problem is commonly referred to as sampling *bias*. Bar-Yossef and Gurevich [3] introduce two methods for obtaining an unbiased sample from a search system. The novelty of their approach is that they take into account the probability that a document is sampled. They use stochastic simulation techniques to produce nearly unbiased samples. Their goal is to produce realistic estimates of the sizes of search engine indices. The question is whether we really need unbiased samples for building resource descriptions [1]. We do not further investigate this issue in this paper. However, we do use some of their ideas for a specific querying strategy.

Ipeirotis, et al. [18] introduce a variant of query-based sampling called focused probing. They apply machine learning methods to learn the categorization of a remote server based on the returned content. For example, a server that returns documents containing 'hepatitis' and 'MRSA', would be placed in the 'Health' category. This information can be used by a central system to narrow down the server selection for a user query. For example, all queries typically associated with Health are forwarded to sites in this category, the returned results are merged and then presented to the user.

Other approaches go deeper into the functionality of the search interface itself. For example, Bergholz and Chidlovskii [5] investigate how to find out what type of complex queries a free-text search field supports, for example: if it supports Boolean queries and what operators it supports for such queries. Another approach assumes that the page on which the search interface resides contains clues about the underlying content [11]. This is suitable for search forms that contain additional interface elements to narrow a search. In our research we restrict ourselves to the basic assumption that there is a single search field devoid of semantic clues. We send only single-term queries via this interface.

Table 1: Properties of the data sets used.

Name	Raw	Index	#Docs	# Terms	# Unique
OANC	97M	117M	8,824	14,567,719	176,691
TREC123	2.6G	3.5G	1,078,166	432,134,562	969,061
WT2G	1.6G	2.1G	247,413	247,833,426	1,545,707

3. METHODOLOGY

In our experimental setup we have a single remote server whose content we wish to estimate by sampling. This server provides a minimal search interface: it can only take queries and return search results consisting of a list of documents. Each document in this list is downloaded and used to build a resource description in the form of a vocabulary with frequency information, also called a language model [8]. The act of submitting a query to the remote server, obtaining search results, downloading the documents, updating the local language model and calculating values for the evaluation metrics is called an *iteration*. An iteration consists of the following steps:

1. Pick a one-term query.
   (a) In the first iteration our local language model is empty and has no terms. Thus, for bootstrapping, we pick a random term from an external resource.
   (b) In subsequent iterations we pick a term based on a *term selection strategy*. We only pick terms that we have not yet submitted previously as query.
2. Send the query to the remote server, requesting a maximum number of results ($n = 10$).
3. Download all the returned documents ($1 \leq n \leq 10$).
4. Update the resource description using the content of the returned documents.
5. Evaluate the iteration by comparing the language model of the remote server with the local model (see metrics described in Section 3.2)
6. Terminate if a stopping criterion has been reached, otherwise go to step 1.

This set-up is similar to that of Callan and Connell [8]. However, several differences exist. During evaluation (step five) we compare all word forms in the index *unstemmed* which can lead to a slower performance increase. The database indices used by Callan and Connell contain only stemmed forms. Additionally, the underlying system uses fewer stop words. Callan and Connell use a list of 418 stop words. We use a conservative subset of this list consisting of 33 stop words which is distributed with Apache Lucene. We examine no more than 10 documents per query, as this is the number of search results commonly returned by search engines on the initial page nowadays. Callan concluded that the influence of the number of documents examined per query is small on average. Examining more documents appears to result in faster learning, although with more variation [8, 9].

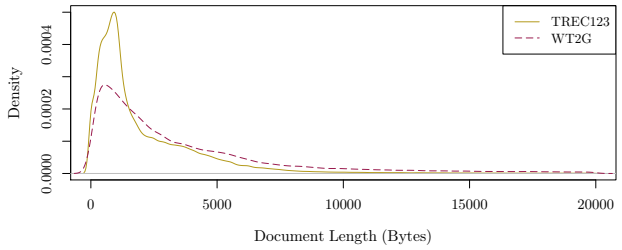

Figure 1: Kernel density plot of document length distributions up to 20 KB.

3.1 Data sets

We used the following data sets to conduct our tests:

OANC-1.1: The Open American National Corpus: A heterogeneous collection consisting of: transcribed speech of face-to-face and telephone conversations, technical documentation, fiction and non-fiction, research papers and some minor amount of web data. We use it exclusively for selecting bootstrap terms [17].

TREC-123: A heterogeneous collection consisting of TREC Volumes 1–3. Consists of: short newspaper and magazine articles, scientific abstracts, and government documents [14]. Used in previous experiments by Callan, et al.[8]

WT2G: Web Track 2G: A small subset of the Very Large Corpus web crawl conducted in 1997 [15].

Table 1 shows some properties of the data sets. It shows the raw size of document content in bytes, the size of the corresponding Apache Lucene index, the number of documents in the index, and the number of tokens (terms) and types (unique terms). Figure 1 shows the distribution of document lengths in TREC-123 and WT2G as a kernel density plot [25]. We see that WT2G has a more gradual distribution of document lengths, whereas TREC-123 shows a sharper decline near two kilobytes. Both collections consist primarily of many small documents.

The OANC is used as external resource to select bootstrap queries as follows: on the first iteration of our experiment we pick a random term out of the top 25 most-frequent terms, excluding stop words, in the OANC.

3.2 Metrics

Evaluation is done by comparing the complete remote language model with the subset local language model each iteration. We discard stop words, and compare terms unstemmed. Various metrics have conventionally been used to conduct this comparison. Early query-based sampling papers relied on Collection Term Frequency (CTF) and the Spearman Rank Correlation Coefficient (SRCC) [8]. Later papers challenge the validity of these metrics and opt for using Kullback-Leibler Divergence (KLD) instead [2]. In this paper we report results using both CTF and KLD. Additionally, we use the Jensen-Shannon Divergence (JSD) for reasons outlined below.

We first discuss the Collection Term Frequency (CTF) ratio. This metric expresses the coverage of the terms of the locally learned language model as a ratio of the terms of the actual remote model. It is defined as follows [9]:

$$CTF_{ratio}\left(\mathscr{T}, \hat{\mathscr{T}}\right) = \sum_{\mathbf{t} \in \hat{\mathscr{T}}} \frac{CTF\left(\mathbf{t}, \mathscr{T}\right)}{\sum_{\mathbf{u} \in \mathscr{T}} CTF\left(\mathbf{u}, \mathscr{T}\right)} \tag{1}$$

where $\mathscr{T}$ the actual model and $\hat{\mathscr{T}}$ the learned model. The CTF function returns the number of times a term $\mathbf{t}$ occurs in the given model. The higher the CTF ratio, the more important terms have been found. A ratio of zero indicates that no terms were found, whereas a ratio of one indicates that the learned language model is identical to the remote model. For example, if the remote model consists of the term 'pear' forty-nine times and the term 'lion' once, and we have locally seen only the occurrences of the word 'pear', then the CTF ratio is 98 percent $(49 \div (49 + 1) = 0.98)$.

The Kullback-Leibler Divergence (KLD), sometimes called relative entropy, gives an indication of the extent to which two probability models, in this case our local and remote language models, will produce the same predictions. The output is the number of additional bits it would take to encode one model into the other. It is defined as [21, p. 231]:

$$KLD\left(\mathscr{T} \parallel \hat{\mathscr{T}}\right) = \sum_{\mathbf{t} \in \mathscr{T}} P\left(\mathbf{t} \mid \mathscr{T}\right) \cdot \log \frac{P\left(\mathbf{t} \mid \mathscr{T}\right)}{P\left(\mathbf{t} \mid \hat{\mathscr{T}}\right)} \tag{2}$$

where $\hat{\mathscr{T}}$ is the learned model and $\mathscr{T}$ the actual model. KLD has several disadvantages. Firstly, if a term occurs in one model, but not in the other it will produce zero or infinite numbers. Therefore, smoothing is commonly applied such as Laplace smoothing, which simply adds one to all counts of the learned model $\hat{\mathscr{T}}$. This ensures that each term in the remote model exists at least once in the local model, thereby avoiding divisions by zero [1]. Secondly, the KLD is asymmetric, which is expressed via the double bar notation. Manning [22, p. 304] argues that using Jensen-Shannon Divergence (JSD), also called Information Radius or total divergence to the average, solves this. It is defined in terms of the KLD as [12]:

$$JSD\left(\mathscr{T}, \hat{\mathscr{T}}\right) = KLD\left(\mathscr{T} \| \frac{\mathscr{T} + \hat{\mathscr{T}}}{2}\right) + KLD\left(\hat{\mathscr{T}} \| \frac{\mathscr{T} + \hat{\mathscr{T}}}{2}\right) \tag{3}$$

The Jensen-Shannon Divergence (JSD) expresses how much information is lost if we describe two distributions with their average distribution. This average distribution is formed by summing the counts for each term that occurs in either model and taking the average by dividing this by two. Using the average distribution is a form of smoothing which does not require assigning additional artificial probability weight in contrast with the KLD. Other differences with the KLD are that the JSD is symmetric and finite. Conveniently, when using a logarithm of base 2 in the underlying KLD, the JSD ranges from 0.0 for identical distributions to 2.0 for maximally different distributions.

As a final metric we report the Result-List lengNth (RLN). This is the average number of results returned per iteration. Since we request only 10 results, the RLN is at least 0 and at most 10.

4. STRATEGIES

In conventional query-based sampling, terms in iterations after the first one are selected at random from the language model constructed so far. This means that if we obtain 100 terms after the first iteration, each term in the language model has an equal probability of $1/100 = 0.001$ of being selected for the next iteration. The probability of an individual term being selected decreases as the vocabulary size increases. This is called uniform random selection.

4.1 Vocabulary Frequency-Based Measures

A simple improvement over selecting a random query term is selecting a term based on frequency information of the terms in the documents retrieved so far. Since a language model is more than just a vocabulary, it also contains term occurrence counts. We devised several selection strategies that exploit this. To illustrate we use the following example: $[(lychee, 6), (okra, 3), (rambutan, 1)]$. This means that 60 percent of the terms seen so far are *lychee*, 30 percent are *okra* and the remaining 10 percent are *rambutan*. In a uniform random selection scenario, which disregards the frequencies, all terms are equally likely to be selected (~ 33.3 percent). Choosing between them would be like rolling a three-sided dice. Let us investigate several alternatives to uniform random selection:

Biased-Random-Collection Terms with a high frequency in the language model are more likely to be selected. This simply uses the frequency information to alter the selection probability. In this case the probability for *lychee* would be 0.6, for *okra* 0.3 and for *rambutan* 0.1.

Least-Frequent Select the term with the lowest frequency in the language model. In the example this would be *rambutan,* since: $0.1 < 0.3 < 0.6$

Most-Frequent Select the term with the highest frequency in the language model. In the example this would be *lychee*, since: $0.6 > 0.3 > 0.1$

For these last two approaches, it is possible that there are multiple terms with the same lowest or highest frequency. For example: $[(apple, 0.1), (pear, 0.1), (banana, 0.1)]$. Selection among such terms with the exact same frequency is random. Because the frequency of terms in a vocabulary follow a Zipf distribution [21, p. 82], such random selection is more likely to occur at low frequencies.

4.2 Document Frequency Based Measures

In each iteration we obtain a sample of documents. In the previous section we considered all documents together as one language model. The approaches in this section use the individual language models of each obtained document.

4.2.1 Biased-Random-Document

Each iteration we throw an n-faced die where n is the number of terms in our local language model. However, the probability of a term being selected is proportional to its document frequency. The higher the document frequency, the more likely the term will be selected. This is equivalent to biased-random-collection, but using document frequencies (in how many documents does a term appear) instead of collection frequencies (how often does a term occur in the entire sample of documents concatenated into one).

4.2.2 Document Information Radius

Each iteration we can use the statistical properties of the documents obtained so far. One way is to adapt the idea behind one of the metrics we use: Jensen-Shannon Divergence (JSD), explained in Section 3.2. We term this approach *document information radius* to prevent confusion with the metric. The intuition behind this method is: from all the documents we first select the one that has the most terms in its language model that diverge from the current local language model of all sampled documents. From this document we select the term that diverges *least* from the local language model as query.

Recall that the JSD calculates the divergence between two language models. In this approach we compare the current local language model to a pool language model of a group of documents. We define that $\mathscr{S}$ represents the entire sample of documents. The steps are as follows:

1. Determine document scores:
   (a) Each document $\mathbf{d} \in \mathscr{S}$ has an initial score of zero.
   (b) For each term $\mathbf{t}$ in the vocabulary of $\mathscr{S}$ consider the pool of documents $\mathscr{D}_\mathbf{t} \subseteq \mathscr{S}$ that contain that term at least once.
      i. Increase the score of each document $\mathbf{d} \in \mathscr{D}_\mathbf{t}$ with the JSD between the language model defined over $\mathscr{D}_\mathbf{t}$ for each term $\mathbf{t}$ and the model of the entire sample $\mathscr{S}$. The document score is thus a sum of JSD values.
2. Select the document with the highest score.
3. From this document select the term $\mathbf{t}$ whose $JSD(\mathscr{D}_\mathbf{t}, \mathscr{S})$ contributed least to the document score.

For example: if we have document A with score 0.5 and B with 1.0, we would select document B since it has a higher score. Thereafter we would select from the individual terms in document B the one with the lowest JSD.

4.2.3 Document Potential

Assume that it is preferable to use terms from many different documents as queries to obtain a good sample. Given this we need some way to avoid selecting query terms from one document too often. We need to determine the *document potential*. To do this we use appearance counts. Each iteration a document appears in the search results, its count is incremented by one. Thereafter, a query term is selected randomly from the document with the *lowest* appearance count. So, if we have document A with count 1 and a document B with count 2, a term will be drawn randomly from the vocabulary of document A. If there are multiple documents with the same, lowest, appearance count, a document is first selected randomly and then a term from that document's vocabulary. So, if A and B both have count 1, we first select either document A or B with a fifty percent probability and then randomly select a term from the selected document as query.

This approach continually attempts to 'harvest' terms from documents that either appeared in more recent iterations or were neglected before. It indirectly also penalizes long documents that have a higher probability of appearing in search results often.

4.3 Controlled Query Generation

Frequency information can also be used in more complex ways. Controlled query generation was proposed as a means of evaluating blind relevance feedback algorithms [20]. In this approach, queries with high discriminative power are generated from the documents seen so far. Kullback-Leibler Divergence (KLD), also called relative entropy, forms the basis for this calculation. The KLD for each term is calculated between all the documents it appears in and the entire collection.

$$score\left(\mathbf{t}, \mathscr{D}_{\mathbf{t}}, \mathscr{S}\right) = P\left(\mathbf{t} \mid \mathscr{D}\right) \cdot \frac{P\left(\mathbf{t} \mid \mathscr{D}_{\mathbf{t}}\right)}{P\left(\mathbf{t} \mid \mathscr{S}\right)}$$

where $\mathbf{t}$ represents a single term, $\mathscr{D}_{\mathbf{t}}$ the subset of documents of the sample in which $\mathbf{t}$ occurs and $\mathscr{S}$ the sample of the entire collection seen so far. Hence, $\mathscr{D}_{\mathbf{t}} \subseteq \mathscr{S}$. The resulting score represents the power of a term to discriminate a subset of documents with respect to all other terms.

The highest scoring terms are used for querying. This might appear counter-intuitive, since terms with high discriminative power imply that these terms also return fewer and more specific documents. We will see later that returning fewer documents each iteration does not necessarily yield poor modeling performance.

4.4 Query Cardinality

Bar-Yossef, et al. [3, 4] present several approaches to obtain a uniform random sample from the index of large search engines to compare their coverage of the web. An important point they make is that of query overflow and underflow. A query that underflows is one that returns less than the number of desired results, whereas one that overflows returns more. For example: assume that we want each query to yield 10 documents, if only 5 are returned by a query it is a query that underflows. We call the number of results that a query yields the *cardinality* of the query.

The problem of underflow is relevant to query-based sampling. Each iteration we can send one query and get back results. Ideally we would always want each query to return as many document as we request, since processing a query is a costly operation. If one query yields less than the amount of requested documents, we are partially wasting an iteration. This problem is ignored in the foundational query-based sampling papers [9, 8].

To avoid underflow we adopt the rejection sampling method which is illustrated in the pool-based sampler of Bar-Yossef, et al. [3]. To determine which query to send we adopt the following procedure:

1. Select a random term $\mathbf{t}$ from the set of terms $\mathscr{T}$ seen so far (the vocabulary of the local language model).
2. Count the number of documents $\#\mathscr{D}$ in our sample $\mathscr{S}$ that contain $\mathbf{t}$. Use this count as an estimate of the number of results that will be returned.
3. If $\#\mathscr{D}$ is exactly the number of desired documents n, then accept and use this term. Otherwise: with probability $1 - \#\mathscr{D}/n$ reject the term and return to step 1 and with probability $\#\mathscr{D}/n$ accept and use this term.

Terms that refer to few documents in the sample obtained thus far have a lower probability of being selected for obtaining more documents.

5. RESULTS

In this section we report the results of our experiments. Because the queries are chosen randomly, we repeated the experiment 30 times. Thus, all plots shown are based on 30 experiment repetitions. We derived the regression plots from scatter plots. Based on the shape of the data we fitted regression lines using $y = \log(x) + c$. The graphs show results for 100 iterations. We verified that the trends shown continue beyond the graph limit up to 125 iterations. In each iteration a variable number of documents is returned. We show the average number of returned results per iteration in separate graphs.

Figure 2 shows results on TREC-123 using the basic frequency strategies: random, biased-random-collection, biased-random-doc, least-frequent and most-frequent. We see that the baseline, random term selection, performs quite well. While the biased approaches both perform worse than random, although using document frequencies instead of collection frequencies appears to be more optimal. Realize that document frequencies are in fact coarse collection frequencies. The only strategy that actually performs better than random is least-frequent. The opposite strategy, most-frequent, performs worst. Apparently, if we try to sample from a large underlying collection, using the least frequent terms as queries is the most optimal approach.

If we regard the region between the least-frequent and most-frequent strategies in each graph as an upper and lower limit, we can explain why the random strategy already works well. Since there are just a few high frequency terms and many low frequency terms, the probability of randomly selecting a less frequent term is quite large. This is the reason that random is so close to least-frequent. However, there is still a probability of selecting a frequently occurring term, which explains why random is slightly less optimal than always selecting the least frequent term. The number of returned results per iteration decreases more rapidly for least-frequent, but this apparently does not affect the strategy's capability of optimally sampling the underlying collection. The most-frequent strategy always returns ten results. However, these results are likely to be largely the same after each iteration. This redundancy in the results explains most-frequent's relatively poor performance.

Figure 3 shows the results for the advanced strategies. The random baseline is repeated in this graph for comparison. The document potential strategy performs quite poorly. Even though it manages to consistently select terms as query that retrieve a relatively stable number of results, these queries retrieve documents that poorly model the collection. The original scatter plots suggest that this strategy gets stuck on using query terms from one particular document in early iterations. The document information radius strategy always retrieves ten results, but similarly does not perform so well. Query cardinality performs just a little bit worse than random. In contrast, controlled query generation performs a little bit better than random, comparable to least-frequent select.

Figure 4 and Figure 5 show the results for the strategies for WT2G. This collection is more representative for the Web. The pattern for the basic strategies in Figure 4 is mostly the same as for TREC-123. The only difference is the results per iteration which varies much more for WT2G. The more heterogeneous nature of the corpus likely causes this higher variation. Indeed, the performance of

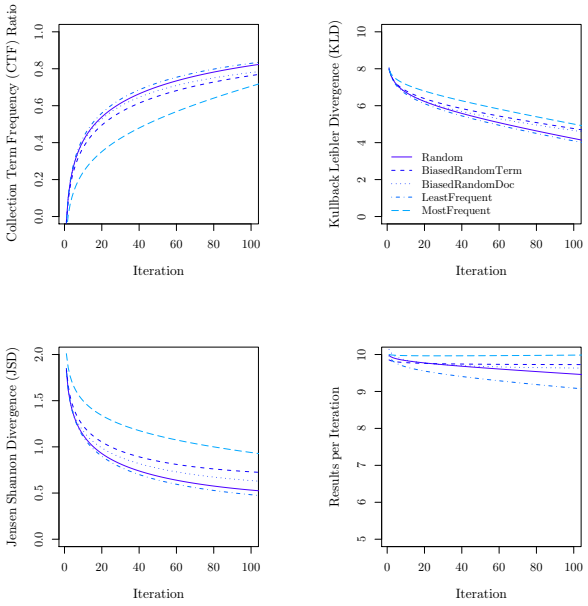

Figure 2: Results for TREC-123 for the basic frequency strategies Legend in the top right graph.

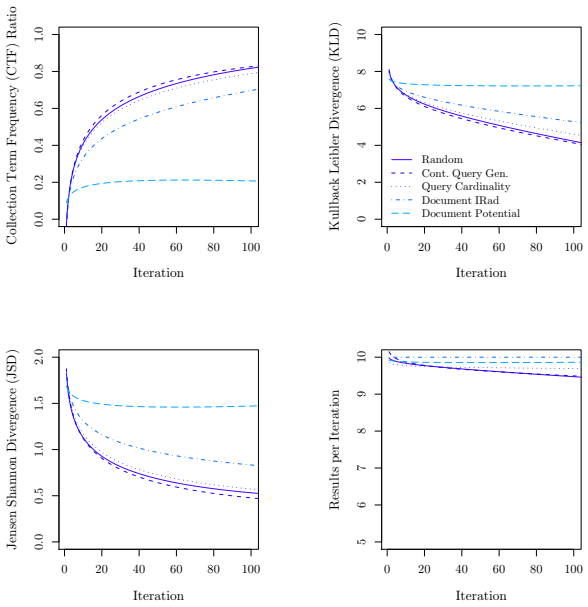

Figure 3: Results for TREC-123 for the advanced strategies. Legend in the top right graph.

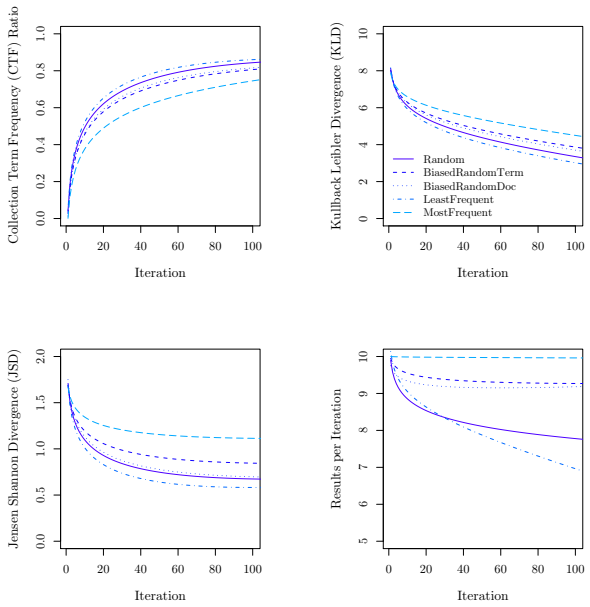

Figure 4: Results for WT2G for the basic frequency strategies. Legend in the top right graph.

least-frequent is quite good considering how few documents it retrieves in later iterations. The advanced strategies in Figure 5 show more difference with TREC-123. Controlled query generation, a strategy that performed better than random for TREC-123, performs quite poorly here. It appears to get stuck on terms that often retrieve the same documents. Other than this the result is similar to TREC-123 with query cardinality performing very close to random.

To give further insight into why least-frequent select performs better, Figure 6 shows scatter plots of the Jensen-Shannon Divergence (JSD) against the number of iterations and against bandwidth. Recall that the regression lines in all the previous graphs are based on 30 repetitions. These scatter plots show about 1000 samples of the source data per graph. Specifically, the two top graphs show the basis for the regression lines in the bottom-left graph of Figure 4. We can see from these two graphs that the random approach has more outliers than the least-frequent strategy. We believe that the outliers for random are frequently occurring terms. Least-frequent never selects these terms. As a result its performance has less variance which explains the better regression line.

We also plotted the JSD against the bandwidth consumption, shown as the bottom graphs in Figure 6. The horizontal axis in these graphs is not the number of iterations, but the combined size in kilobytes of the sample. It appears that the least-frequent strategy also shows more stable performance when plotted against bandwidth. This suggests that it retrieves higher quality documents. Where we define quality in terms of representativeness as measured by the Jensen-Shannon Divergence (JSD). For random, the quality of retrieved documents seems to vary more compared with least-frequent. This implies that least-frequent provides, on average, better results per unit of bandwidth used.

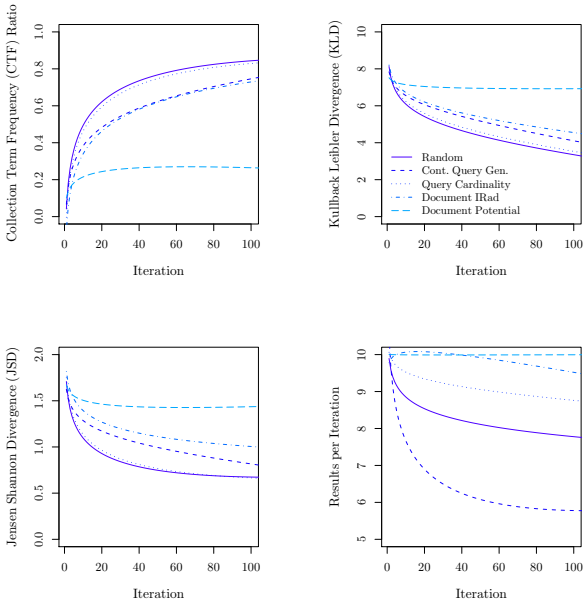

Figure 5: Results for WT2G for the advanced strategies. Legend in the top right graph.

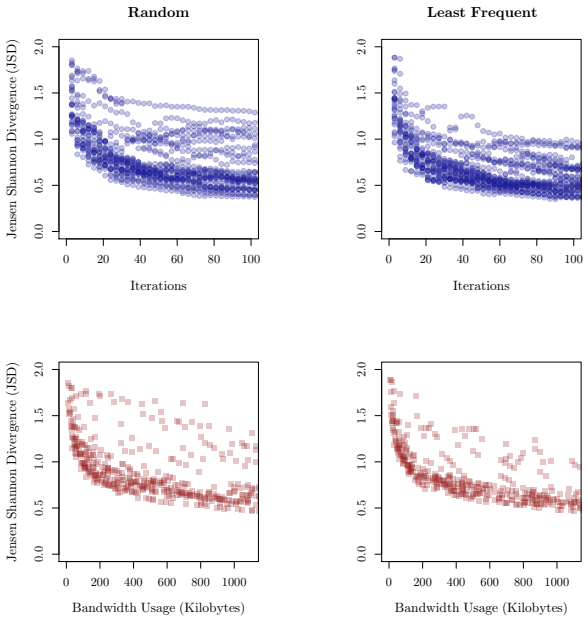

Figure 6: Scatter plots of the Jensen Shannon Divergence (JSD) against the number of iterations (top) and the bandwidth consumption (bottom) for WT2G. The left graphs for the Random strategy, right graphs for Least-Frequent. Each graph is based on approximately 1000 samples.

6. CONCLUSION

In conventional query-based sampling, remote servers are sampled by sending random terms as queries, retrieving the results and using these to build a resource description. We presented several alternative approaches to using random terms. Selecting the least-frequent term in the language model as query yields better performance than selecting a random term. Most other presented strategies did not show consistent improvement for the two test collections used.

The results confirm that using random selection is quite optimal. Yet, using the least-frequent term outperforms random term selection. Even more interesting is that the least-frequent strategy outperforms others, while downloading less documents per iteration on average. It saves more bandwidth as the number of iterations increase. Indeed, this suggests that we need to look beyond the quantity of data, the number of documents, used to build resource descriptions and pay more attention to quality and representativeness of those documents.

Despite the good results for least-frequent, we believe that the size of the underlying database has a huge influence on the performance of any querying strategy. As such the result of this research should be seen as strictly applying to sites that index a large underlying collection in the same, or higher, order as the collections used in this research. The rationale behind this: using a least-frequent strategy on small collections may result in many iterations with a low number of search results. Since rarely occurring terms might appear only in one specific document. For a large collection this problem is less pronounced. Low-frequency terms likely occur in at least some other documents. This also increases the chances that new documents are retrieved, which leads to a more accurate model in fewer iterations than the random approach.

7. FUTURE WORK

We regard investigating the exact influence of the size of the underlying collection as important future work. Combining this with collection size estimation enables faster and less costly construction of resource descriptions tailored to collections of specific sizes. We would need to test on a higher number of recent collections representative of the present day web with varying sizes to produce robust experimental results. Other strategies for term selection could also be explored. However, we believe that improving further over usage of the sampled language model will be difficult. We wish to emphasize that a resource description needs to represent the underlying resource well. It need not necessarily be unbiased. In fact, we believe that some bias makes resource selection easier. The costs of term selection and query construction is an other aspect that should be more deeply evaluated. Directions that explore beyond the usage of the sampled language model have so far focused on intelligently selecting queries from an external resource [18, 19] or using reference queries [1]. Future research could explore the construction of multi-term queries, which possibly return a more diverse set of documents each iteration [20]. Additionally, querying in web forms that have more search field, possibly not all of them free text, could be further explored [24].

8. ACKNOWLEDGMENTS

We thank Stephen Robertson for his insights and Jan Flokstra for his help in setting up the experiments. This paper, and the experiments, were created using only Free and Open Source Software. Finally, we gratefully acknowledge the support of the Netherlands Organization for Scientific Research (NWO) under project DIRKA (NWO-Vidi), Number 639.022.809.

9. REFERENCES

[1] Baillie, M., Azzopardi, L., and Crestani, F. *Adaptive Query-Based Sampling of Distributed Collections*, vol. 4209 of *Lecture Notes in Computer Science*. Springer, 2006, pp. 316–328.

[2] Baillie, M., Azzopardi, L., and Crestani, F. Towards better measures: Evaluation of estimated resource description quality for distributed ir. In *Proceedings of InfoScale* (New York, NY, US, 2006), ACM, p. 41.

[3] Bar-Yossef, Z., and Gurevich, M. Random sampling from a search engine's index. In *Proceedings of WWW* (New York, NY, US, May 2006), ACM, pp. 367–376.

[4] Bar-Yossef, Z., and Gurevich, M. Random sampling from a search engine's index. *Journal of the ACM 55*, 5 (2008), 1–74.

[5] Bergholz, A., and Chidlovskii, B. Using query probing to identify query language features on the web. In *Proceedings of SIGIR Workshop on Distributed Information Retrieval* (July 2003), vol. 2924 of *Lecture Notes in Computer Science*, pp. 21–30.

[6] Bergman, M. K. The deep web: Surfacing hidden value. *Journal of Electronic Publishing 7*, 1 (August 2001).

[7] Callan, J. *Distributed Information Retrieval*. Advances in Information Retrieval. Kluwer Academic Publishers, 2000, ch. 5.

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

[9] Callan, J., Connell, M., and Du, A. Automatic discovery of language models for text databases. In *Proceedings of SIGMOD* (June 1999), ACM Press, pp. 479–490.

[10] Chang, K. C.-C., He, B., Li, C., Patel, M., and Zhang, Z. Structured databases on the web: Observations and implications. *SIGMOD Record 33*, 3 (2004), 61–70.

[11] Chang, K. C.-C., He, B., and Zhang, Z. Toward large scale integration: Building a metaquerier over databases on the web, 2004.

[12] Dagan, I., Lee, L., and Pereira, F. Similarity-based methods for word sense disambiguation. In *Proceedings of ACL* (Morristown, NJ, US, Aug. 1997), Association for Computational Linguistics, pp. 56–63.

[13] Du, A., and Callan, J. Probing a collection to discover its language model. Tech. Rep. UM-CS-1998-029, University of Massachusetts, Amherst, MA, US, 1998.

[14] Harman, D. K. *Overview of the Third Text Retrieval Conference (TREC-3)*. National Institute of Standards and Technology, 1995.

[15] Hawking, D., Voorhees, E., Craswell, N., and Bailey, P. Overview of the trec-8 web track. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, US, 2000.

[16] He, B., Patel, M., Zhang, Z., and Chang, K. C.-C. Accessing the deep web. *Communications of the ACM 50*, 5 (2007), 94–101.

[17] Ide, N., and Suderman, K. The open american national corpus, 2007.

[18] Ipeirotis, P. G., and Gravano, L. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proceedings of VLDB* (2002), VLDB Endowment, pp. 394–405.

[19] Ipeirotis, P. G., and Gravano, L. Classification-aware hidden-web text database selection. *ACM Transactions on Information Systems 26*, 2 (2008), 1–66.

[20] Jordan, C., Watters, C., and Gao, Q. Using controlled query generation to evaluate blind relevance feedback algorithms. In *Proceedings of JCDL* (New York, NY, US, 2006), ACM, pp. 286–295.

[21] Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, US, 2008.

[22] Manning, C. D., and Schütze, H. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, US, June 1999.

[23] Monroe, G., French, J. C., and Powell, A. L. Obtaining language models of web collections using query-based sampling techniques. In *Proceedings of HICSS* (Washington, DC, US, Jan. 2002), vol. 3, IEEE Computer Society, p. 67.

[24] Senellart, P., Mittal, A., Muschick, D., Gilleron, R., and Tommasi, M. Automatic wrapper induction from hidden-web sources with domain knowledge. In *Proceeding of WIDM* (New York, NY, US, Nov. 2008), ACM, pp. 9–16.

[25] Venables, W. N., and Smith, D. M. *An Introduction to R*, Aug. 2009.