Query-Based Sampling using Only Snippets

Almer S. Tigelaar and Djoerd Hiemstra

{tigelaaras, hiemstra}@cs.utwente.nl

Abstract Query-based sampling is a popular approach to model the content of an uncooperative server. It works by sending queries to the server and downloading the returned documents in the search results in full. This sample of documents then represents the server's content. We present an approach that uses the document snippets as samples instead of downloading entire documents. This yields more stable results at the same amount of bandwidth usage as the full document approach. Additionally, we show that using snippets does not necessarily incur more latency, but can actually save time.

1 Introduction

Query-based sampling is a technique for estimating the available content of a search server. This estimate is based on the downloaded content of a small subset of documents the server returns in response to queries [8]. We present an approach that requires no additional downloading beyond the returned search results, but instead relies solely on the information returned as part of the search results: the snippet of each document.

The purpose of knowing what server offers what content is allowing a central server to forward queries to the most suitable server for handling a query. This task is one of the challenges in distributed information retrieval, commonly referred to as *resource selection* or *collection selection* [5, 6]. Selection is based on a representation of the content stored on a server: a *resource description*. Servers can *cooperate* by explicitly providing such representations. However, most servers on the World Wide Web are not cooperative in this sense. Query-based sampling does not rely on such cooperation, instead it exploits only the native search functionality provided by a server.

In conventional query-based sampling, the first step is sending a query to a server. This first query is randomly chosen from a dictionary or other external resource. In response, the server returns a ranked list of search results of which the top N most relevant documents are downloaded in their entirety. The words in this, usually small, sample of documents are then used to build a resource description in the form of a language model. Such a model consists of simple $(word, count)$ pairs, for example: $[(pear, 8), (banana, 23)]$. Each query sent thereafter is simply a random word chosen from the language model built so far. The process of sending queries and downloading documents repeats until reaching a stopping criterion, for example, after sampling 500 unique documents. The

resulting final language model represents an estimate of the available content of the server [7, 8].

The disadvantage of downloading entire documents is bandwidth consumption. One may not even know beforehand the size of a document. In contrast, some data always comes along 'for free' in the returned search results: the snippets. A snippet is a short piece of text consisting of the title and a short summary of a document. The summaries can be either dynamically generated in response to a query, also known as keyword-in-context, or they can be statically defined [13, p. 157]. It is intended to help the user to estimate the relevance of the listed document. We postulate that these snippets can also be used for query-based sampling to build a language model. This way we can avoid downloading entire documents. The most important advantage of this approach is a reduction in bandwidth usage and latency. An other advantage is the capability to constantly update the language model, based on results returned when users submit queries, at no additional overhead. This is especially important for document collections that change over time.

Whether the documents returned in response to random queries are a truly random part of the underlying collection is doubtful. Servers have a propensity to return documents that users indicate as important and the number of in-links has a substantial correlation with this importance [1]. This may not be a problem, as it may be preferable to know only the language model represented by these important documents on a server, since the user is likely to look for those [2].

The approach we take has some similarities with prior research by Paltoglou, et al. [15]. They show that downloading only a part of a document can also yield good performance. However, they download the first two to three kilobytes of each document in the result list, whereas we use small snippets and thus avoid any extra downloading beyond the returned list of search results.

Our main research question is:

> "How does query-based sampling using *only snippets* compare to downloading full documents in terms of the learned language model and imposed latency?"

We describe our experimental setup in section 2. This is followed by section 3 which shows the results. Latency issues are discussed in section 4 and finally the paper concludes with section 5.

2 Experiment Methodology

In our experimental setup we have one 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 listing of documents. For each document a title, snippet and download link is returned. These returned search results are used to locally build a resource description in the form of a vocabulary with frequency information, also called a language model [7]. The act

of submitting a query to the remote server, obtaining search results, 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 random term from our local language model that we have not yet submitted previously as query.
2. Send the query to the remote server, requesting a maximum number of results ($n = 10$). In our set-up, the maximum length for the document summaries may be no more than 2 fragments of 90 characters each ($s \leq 2 \cdot 90$).
3. Update the resource description using the returned results ($1 \leq n \leq 10$).
   (a) For the full document strategy: download all the returned documents and use all their content to update the local language model.
   (b) For the snippet strategy: use the snippet of each document in the search results to update the local language model. If a document appears multiple times in search results, use its snippet only if it differs from previously seen snippets of that document.
4. Evaluate the iteration by comparing the language model of the remote server with the local model (see metrics described in Section 2.2).
5. Terminate if a stopping criterion has been reached, otherwise go to step 1.

This set-up is similar to that of Callan and Connell [7]. However, several differences exist. During evaluation (step four) 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 contained only stemmed forms. An additional difference is that 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. 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 [7, 8].

The snippet approach uses the title and summary of each document returned in the search result. The way in which this snippet is generated affects the performance. Our simulation environment uses Apache Lucene which generates keyword-in-context document snippets [13, p. 158]. These snippets are constructed simply by using words surrounding a query term in a document, and without keeping into account sentence boundaries. For all our experiments the snippets were allowed to consist of two keyword-in-context segments each of which may be no longer than ninety characters. This length boundary is similar to the one modern web search engines use to generate their snippets.

Table 1: Properties of the data sets used.

Name	Byte Size	Index Size	#Documents	# Terms	# Unique
OANC-1.1	97MB	117MB	8,824	14,567,719	176,691
TREC-123	2.64GB	3.50GB	1,078,166	432,134,562	969,061
WT2G	1.57GB	2.10GB	247,413	247,833,426	1,545,707

2.1 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 [12].

TREC-123: A heterogeneous collection consisting of TREC Volumes 1–3. Consists of: short newspaper and magazine articles, scientific abstracts, and government documents [10]. Used in previous experiments by Callan, et al.[7]

WT2G: Web Track 2G: A small subset of the Very Large Corpus web crawl conducted in 1997 [11].

The OANC is used as external resource to select bootstrap terms 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.

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). However, this table gives little information about the distribution of the document size. Therefore we included Figure 1 which shows a kernel density plot of the size distributions [16]. The distribution is shown only for TREC-123 and WT2G. We omitted the OANC, since we use it merely for bootstrapping. 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.

2.2 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 exist to conduct this comparison.

Early query-based sampling papers relied on a combination of two metrics [7]. The first is the Collection Term Frequency (CTF) ratio, used to measure if the important terms are found. The second is Spearman's Rank Correlation

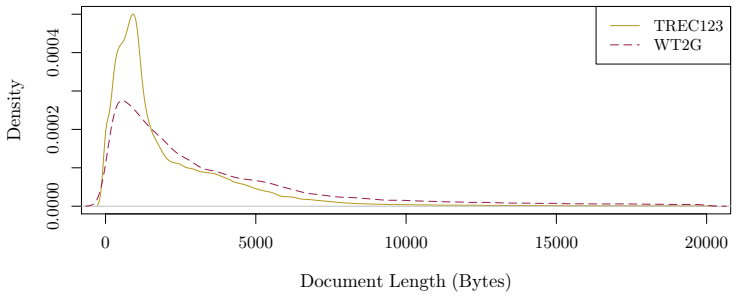

Figure 1: Kernel density plot of document length distributions up to 20 KB.

Coefficient, used to express to what extent the term rankings of the learned language model coincide with those of the actual model. Recent work questions the validity of these measures and proposed using Kullback-Leibler Divergence (KLD) instead [3].

We would like to be able to compare our results with a large body of prior research. Therefore, we will use both part of the early approach for comparability (the CTF) and the newer metric (KLD) in this research. Additionally, we introduce the usage of the Jensen-Shannon Divergence (JSD) in this context. We believe it is a better choice than the KLD 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 [8]:

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

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 of the 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 language 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 all 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 follows [13, 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. We apply 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 [2]. Secondly, the KLD is asymmetric, which is expressed via the double bar notation. Manning [14, 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 [9]:

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

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.

3 Experimental 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.

Figure 2 shows our experimental results in the conventional way for query-based sampling: a metric on the vertical axis and the number of iterations on the horizontal axis [3, 7]. To approximate the number of documents seen, the iteration number should be multiplied by 9.5, the average number of documents returned in each iteration ± 0.5. The graphs thus show results for both approaches up to about 950 documents. The lines shown are based on averaging the result of each iteration over the 30 repetitions. We have omitted graphs for WT2G as they are similar in shape.

The full document approach outperforms the snippet approach for all the defined metrics. However, the comparison is unfair. As the bottom right graph shows, the amount of bandwidth consumed when using full documents is much larger than when using snippets. Full documents performs better, simply because

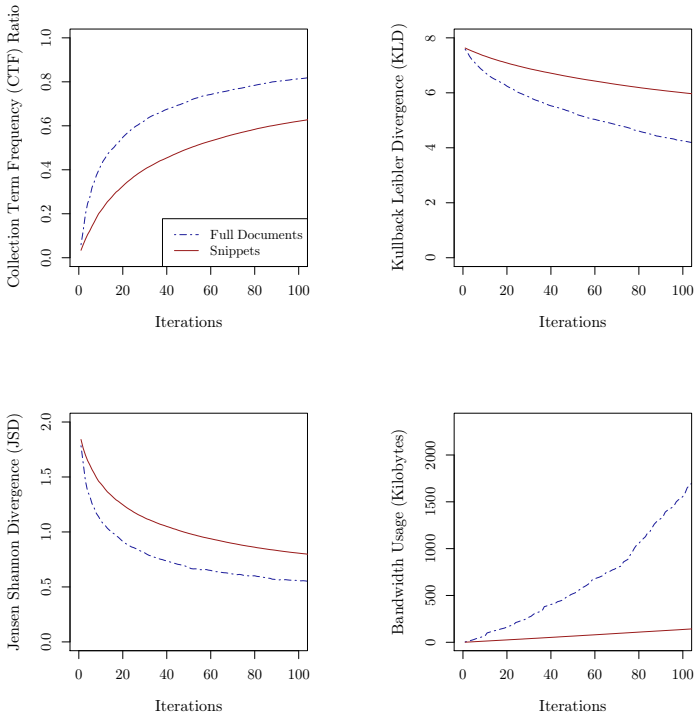

Figure 2: Results for TREC-123. Shows CTF, KLD, JSD and bandwidth usage, plotted against the number of iterations. Shows both the full document and snippet-based approach. The legend is shown in the top left graph.

it acquires more data in less iterations. The reason for this is that the snippet approach has a relatively stable document size: the size of the snippet, which can vary only between 1–180 bytes. The full document approach downloads whole documents which can wildly differ in size. So, a reasonable question at this point is: how do the approaches perform with respect to bandwidth usage?

Figure 3 shows scatter plots of the Jensen-Shannon Divergence (JSD) against bandwidth usage. Scatter plots are frequently used to examine the relationship between variables [17, p. 154]. The top row shows results for full documents, the bottom row for snippets. The left graphs show results using TREC-123, the right graphs using WT2G. We plotted approximately 1000 samples for both methods in the 0–1000KB range obtained from the 30 repetitions of the experiment. Samples with close values overlap visually in the graph and form dense regions. For the full document approach the number of required iterations to get to 1000 KB greatly varies (75–125 for TREC-123). For snippets this is stable at around 682 regardless of the collection used.

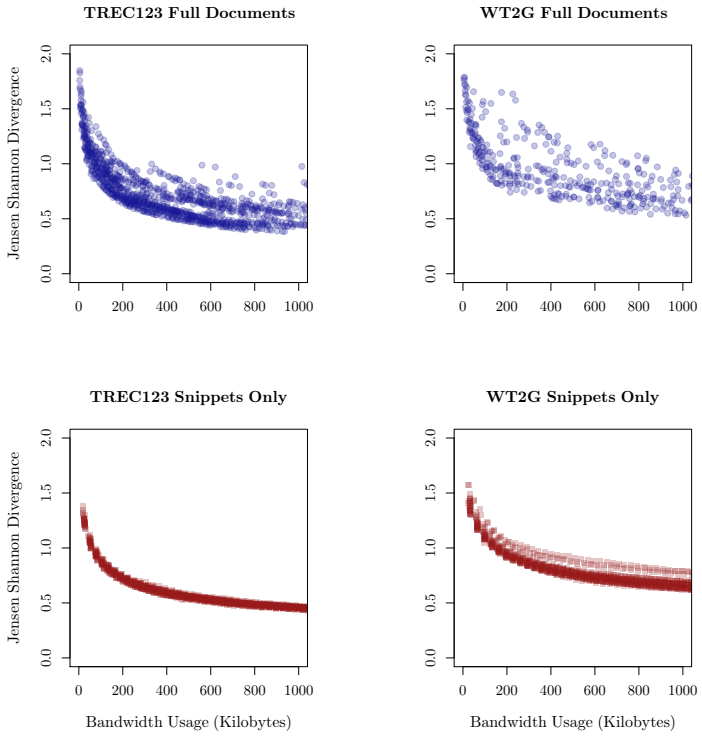

Figure 3: Jensen-Shannon Divergence (JSD) against bandwidth of up to 1000 KB for TREC-123 and WT2G. Axis titles are shown on the left and bottom graphs.

The figures show that the performance of the full document approach greatly varies. This problem seems to be more prevalent in the WT2G web document corpus. In contrast, the snippet approach delivers stable performance for a given amount of bandwidth with few outliers. The instability for full documents is caused by variations in document length and quality. Downloading a long document that poorly represents the underlying collection is heavily penalised. The snippet approach never makes very large 'mistakes' like this, because its document length is bound to the maximum summary size (180 bytes).

We observed that variance in the performance of the full document approach exists for all metrics at least to some extent. The snippet approach is more stable. We generated regression plots based on the scatter plots. Figure 4 shows the result. Based on the shape of the data we fitted regression lines using $y = \log(x) + c$. Keep in mind that the performance for the snippet approach deviates much less from these lines than those of full documents. The bottom two graphs are based on the scatter plots in Figure 3.

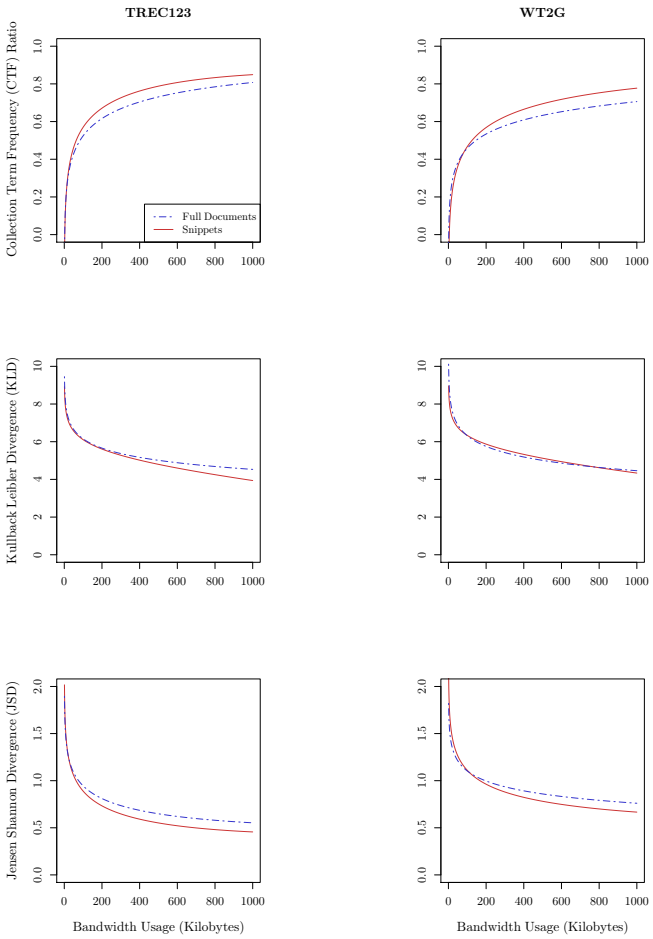

Figure 4: Regression plots for all metrics against bandwidth usage up to 1000 KB. The left graphs show results for TREC-123, the right for WT2G. Axis titles are shown on the left and bottom graphs, the legend in the top left graph.

Initially little difference can be seen between the approaches. However, as the amount of bandwidth used increases, the snippet approach wins out in terms of performance per unit of bandwidth used for most metrics. It makes better usage of bandwidth as the amount available increases. These trends continue beyond the graph limit. However, the differences appear to be less pronounced for the WT2G collection than for TREC-123. Indeed, in the KLD graph of WT2G the lines are nearly identical. This might be an effect of Laplace smoothing, since the JSD is unaffected.

4 Latency

One argument against snippet query-based sampling is that of latency. Posing a query and returning results takes time. Since the snippet approach needs many iterations to consume the same amount of bandwidth as full documents, it might impose additional delay. We wonder: at the same bandwidth usage, does the snippet approach incur more latency compared with full documents?

An argument in favour of snippet query-based sampling is that nothing extra needs to be downloaded: it relies solely on the snippets. In a real-world system these snippets come along 'for free' with the result returned in response to each query posed to the system. We assume a set-up were obtaining search results costs 100 milliseconds (ms), downloading a document costs 100 ms for setting up the connection and 1 ms for each kilobyte downloaded (assuming an 8 megabit connection). The same server that provides the search service also stores all documents. Each iteration the full document strategy costs at least 1050 ms: 100 ms for the search results, 950 ms for downloading the on average 9.5 returned documents. Whereas for snippets the costs are only 100 ms: for downloading the search results only.

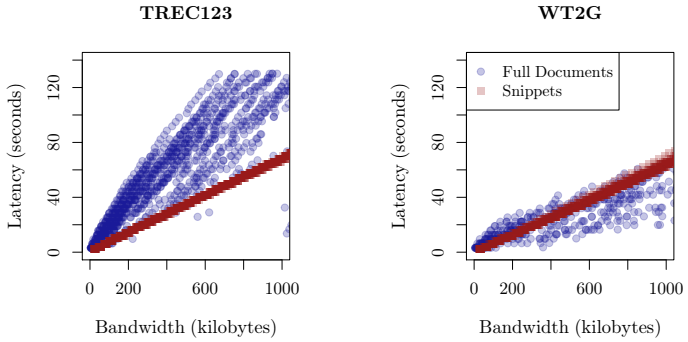

Figure 5: Latency of full documents and snippets.

Figure 5 shows the comparison in scatter plots. The performance variation of the full document approach is large. TREC-123 contains many small documents which is the reason for its consistent poor performance with full documents: it needs to initiate more downloads. The snippet approach performs significantly better. However, for WT2G the full document approach seems more optimal. WT2G has a more even document length distribution with fewer small documents as shown in Figure 1. The result thus depends on the characteristics of the documents that make up the collection. Selecting different costs also leads to different performance. For example, with a rate one kilobyte per 50 ms the snippet approach always has the lowest latency for both data sets. Since the full document approach requires initiating more connections and also requires more bandwidth, it is much more sensitive to the characteristics of the connection.

5 Conclusions and Future Work

We have shown an approach for obtaining resource description based solely on the document snippets returned as part of search results. Compared with the conventional query-based sampling approach, that uses full documents, our snippet approach shows more stable performance per unit of bandwidth consumed. However, it also requires many more iterations of sending queries and examining search results. Nevertheless, we have also shown that, depending on the underlying collection, this does not necessarily impose additional latency. Often it can be even faster than downloading full documents, which also has latency issues.

We believe that snippet query-based sampling is highly applicable for low connection speeds, and when a guaranteed level of performance per bandwidth unit is required. Additionally, we think that this approach is also very practical for real-world distributed information retrieval systems that use a central server. Assume a scenario where the central server forwards queries it receives from users to other machines and aggregates the returned search results. The server can use this same search result data to keep its resource descriptions up to date without imposing additional overhead. This approach also copes with changes in the document collection at the other machines.

This research opens up many possibilities for future work. For example, what is the effect of collection size? Also, Kullback-Leibler Divergence (KLD) and especially Jensen-Shannon Divergence (JSD) are good approaches for comparing probabilistic models. However, to what extent they reflect the utility of search systems for users in a practical, real-world, setting has not been well studied. A measure for how representative the resource descriptions obtained by sampling in general are for real-world usage would be very useful. This remains an open problem, also for full document query-based sampling, even though some attempts have been made to solve it [4]. Also, the influence of the ratio of snippet to document size could be further investigated.

6 Acknowledgements

We thank the USI Lugano Information Retrieval group for their comments, notably Mark Carman and Cyrus Hall. We also thank Kien Tjin-Kam-Jet and Jan Flokstra. This paper, and the experiments, were created using only Free and Open Source Software. Finally, we gratefully acknowledge the support of the Netherlands Organisation for Scientific Research (NWO) under project DIRKA (NWO-Vidi), Number 639.022.809.

Bibliography

[1] Leif Azzopardi, Maarten de Rijke, and Krisztian Balog. Building simulated queries for known-item topics: An analysis using six european languages. In *Proceedings of SIGIR*, pages 455–462, New York, NY, US, July 2007. ACM.

[2] Mark Baillie, Leif Azzopardi, and Fabio Crestani. *Adaptive Query-Based Sampling of Distributed Collections*, volume 4209 of *Lecture Notes in Computer Science*, pages 316–328. Springer, 2006.

[3] Mark Baillie, Leif Azzopardi, and Fabio Crestani. Towards better measures: Evaluation of estimated resource description quality for distributed ir. In *Proceedings of InfoScale*, page 41, New York, NY, US, 2006. ACM.

[4] Mark Baillie, Mark James Carman, and Fabio Crestani. A topic-based measure of resource description quality for distributed information retrieval. In *Proceedings of ECIR*, volume 5478 of *Lecture Notes in Computer Science*, pages 485–497. Springer, April 2009.

[5] Sander Bockting. Collection selection for distributed web search. Master's thesis, University of Twente, February 2009.

[6] Jamie Callan. *Distributed Information Retrieval*, chapter 5. Advances in Information Retrieval. Kluwer Academic Publishers, 2000.

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

[8] Jamie Callan, Margaret Connell, and Aiqun Du. Automatic discovery of language models for text databases. In *Proceedings of SIGMOD*, pages 479–490. ACM Press, June 1999.

[9] Ido Dagan, Lillian Lee, and Fernando Pereira. Similarity-based methods for word sense disambiguation. In *Proceedings of ACL*, pages 56–63, Morristown, NJ, USA, August 1997. Association for Computational Linguistics.

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

[11] David Hawking, Ellen Voorhees, Nick Craswell, and Peter Bailey. Overview of the trec-8 web track. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, US, 2000.

[12] Nancy Ide and Keith Suderman. The open american national corpus, 2007.

[13] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[14] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, US, June 1999.

[15] Georgios Paltoglou, Michail Salampasis, and Maria Satratzemi. Hybrid results merging. In *Proceedings of CIKM*, pages 321–330, New York, NY, US, November 2007. ACM.

[16] William N. Venables, David M. Smith, and R Development Team. *An Introduction to R*, August 2009.

[17] Neil A. Weiss. *Elementary Statistics*. Addison-Wesley, 5th edition, 2002.