Query-Based Sampling using Snippets

Almer S. Tigelaar
Database Group, University of Twente,
Enschede, The Netherlands
a.s.tigelaar@cs.utwente.nl

Djoerd Hiemstra
Database Group, University of Twente,
Enschede, The Netherlands
hiemstra@cs.utwente.nl

ABSTRACT

Query-based sampling is a commonly used approach to model the content of servers. Conventionally, queries are sent to a server and the documents in the search results returned are downloaded in full as representation of the server's content. We present an approach that uses the document snippets in the search results as samples instead of downloading the entire documents. We show this yields equal or better modeling performance for the same bandwidth consumption depending on collection characteristics, like document length distribution and homogeneity. Query-based sampling using snippets is a useful approach for real-world systems, since it requires no extra operations beyond exchanging queries and search results.

1. INTRODUCTION

Query-based sampling is a technique for obtaining a resource description of a search server. This description 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 results, but instead relies solely on information returned as part of the results: the snippets.

Knowing what server offers what content allows a central server to forward queries to the most suitable server for handling a query. This task is commonly referred to as *resource selection* [6]. Selection is based on a representation of the content of a server: a *resource description*. Most servers on the web are uncooperative and do not provide such a description, thus query-based sampling exploits only the native search functionality provided by such servers.

In conventional query-based sampling, the first step is sending a query to a server. The server returns a ranked list of results of which the top N most relevant documents are downloaded and used to build a resource description. Queries are randomly chosen, the first from an external resource and subsequent queries from the description built so far. This repeats until a stopping criterion is reached [7, 8].

LSDSIR 2010, July 23rd, 2010, Geneva, Switzerland.

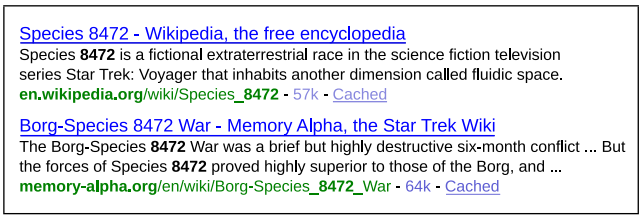

Figure 1: Example snippets. From top to bottom: each snippet consists of an underlined title, a two line summary and a link.

Disadvantages of downloading entire documents are that it consumes more bandwidth, is impossible if servers do not return full documents, and does not work when the full documents themselves are non-text: multimedia with short summary descriptions. 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 a document title, a short summary and a link as shown in Figure 1. A summary can be either dynamically generated in response to a query or is statically defined [16, p. 157]. 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 and thus reduce bandwidth usage and cope with servers that return only search results or contain multimedia content. However, since snippets are small we need to see many of them. This means that we need to send more queries compared with the full document approach. While this increases the query load on the remote servers, it is an advantage for live systems that need to sample from document collections that change over time, since it allows continously updating the language model, based on the results of live queries.

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 is preferable to know only the language model represented by these important documents, since the user is likely to look for those [3]. Recent work [5] focuses on obtaining uniform random samples from large search engines in order to estimate their size and overlap. Others [20] have evaluated this in the context of obtaining resource descriptions and found that it does not consistently work well across collections.

The foundational work for acquiring resource descriptions via query-based sampling was done by Callan et al. [7, 8]. They show 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. The test collection used in their research, TREC123, is not a web data collection. While this initially casts doubt on the applicability of the query-based sampling approach to the web, Monroe et al. [18] show that it also works very well for web data.

The approach we take has some similarities with prior research by Paltoglou et al. [19]. They show that downloading only a part of a document can also yield good modelling 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 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?"

We show that query-based sampling using snippets offers similar performance compared to using full documents. However, using snippets uses less bandwidth and enables constantly updating the resource description at no extra cost. Additionally, we introduce a new metric for comparing language models in the context of resource descriptions and a method to establish the homogeneity of a corpus.

We describe our experimental setup in section 2. This is followed by section 3 which shows the results. Finally, the paper concludes with sections 4 and 5.

2. METHODOLOGY

In our experimental set-up we have one remote server which content we wish to estimate by sampling. This server can only take queries and return search results. For each document a title, snippet and download link is returned. These 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. In this case we pick a random term from an external resource as query.
   (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 of 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 *unstemmed* 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.

Since the snippet approach uses the title and summary of each document returned in the search result, the way in which the summary is generated affects the performance. Our simulation environment uses Apache Lucene which generates keyword-in-context document summaries [16, p. 158]. These summaries are constructed by using words surrounding a query term in a document, without keeping into account sentence boundaries. For all experiments the summaries consisted of two keyword-in-context segments of maximally ninety characters. This length boundary is similar to the one modern web search engines use to generate their summaries. One might be tempted to believe that snippets are biased due to the fact that they commonly also contain the query terms. However, in full-document sampling the returned documents *also* contain the query and have a similar bias, although mitigated by document length.

2.1 Data sets

We used the following data sets to conduct our tests:

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
WIKIL	163M	84M	30,006	9,507,759	108,712
WIKIM	58M	25M	6,821	3,003,418	56,330

OANC-1.1: The Open American National Corpus: A heterogeneous collection. We use it exclusively for selecting bootstrap terms [14].

TREC123: A heterogeneous collection consisting of TREC Volumes 1–3. Contains: short newspaper and magazine articles, scientific abstracts, and government documents [12]. 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 [13].

WIKIL: The large Memory Alpha Wiki. `http://memory-alpha.org`

WIKIM: The medium sized Fallout Wiki. `http://fallout.wikia.com`

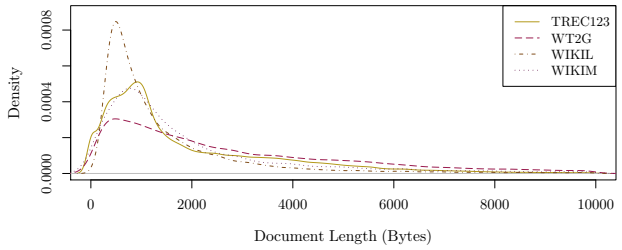

Figure 2: Kernel density plot of document lengths up to 10 Kilobytes for each collection.

The OANC is used as external resource to select a bootstrap term on the first iteration: we pick a random term out of the top 25 most-frequent terms (excluding stop words). TREC123 is for comparison with Callan's work [7]. WT2G is a representative subset of the web. It has some deficiencies, such as missing inter-server links [2]. However, since we use only the page data, this is not a major problem for this experiment.

Our experiment is part of a scenario where many sites offer searchable content. With this in mind using larger monolithic collections, like ClueWeb, offers little extra insights. After all: there are relatively few websites that provide gigabytes or terabytes of information, whereas there is a long tail that offers smaller amounts. For this purpose we have included two Wiki collections in our tests: WIKIL and WIKIM. All Wiki collection were obtained from Wikia, on October 5th 2009. Wikis contain many pages in addition to normal content pages. However, we index *only* content pages which is the reason the raw sizes of these corpora are bigger than the indices.

Table 1 shows some properties of the data sets. We have also included Figure 2 which shows a kernel density plot of the size distributions of the collections [21]. We see that WT2G has a more gradual distribution of document lengths, whereas TREC123 shows a sharper decline near two kilobytes. Both collections consist primarily of many small documents. This is also true for the Wiki collections. Especially the WIKIL collection has many very 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. For comparability with earlier work we use two metrics and introduce one new metric in this context: the Jensen-Shannon Divergence (JSD), which we believe is a better choice than the others 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]:

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

where $\mathscr{T}$ is 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 symbol α represents the sum of the CTF of all terms in the actual model $\mathscr{T}$, which is simply the number of tokens in $\mathscr{T}$. The higher the CTF ratio, the more of the important terms have been found.

The Kullback-Leibler Divergence (KLD) 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 [16, p. 231]:

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

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, 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 [3]. Secondly, the KLD is asymmetric, which is expressed using the double bar notation. Manning [17, p. 304] argues that using Jensen-Shannon Divergence (JSD) solves both problems. It is defined in terms of the KLD as [9]:

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

The Jensen-Shannon Divergence (JSD) expresses how much information is lost if we describe two distributions with their average distribution. This 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 is a form of smoothing which avoids changing the original counts 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. RESULTS

In this section we report the results of our experiments. Because the queries are chosen randomly, we repeated the experiment 30 times.

Figure 3 shows our results on TREC123 in the conventional way for query-based sampling: a metric against the number of iterations on the horizontal axis [7]. We have omitted graphs for WT2G and the Wikia collections as they are highly similar in shape.

As the bottom right graph shows, the amount of bandwidth consumed when using full documents is much larger than when using snippets. Full documents downloads each of the ten documents in the search results, which can be potentially large. Downloading all these documents also uses many connections to the server: one for the search results plus ten for the documents, whereas the snippet approach uses only one connection for transferring the search results and performs no additional downloads.

The fact that the full documents approach downloads a

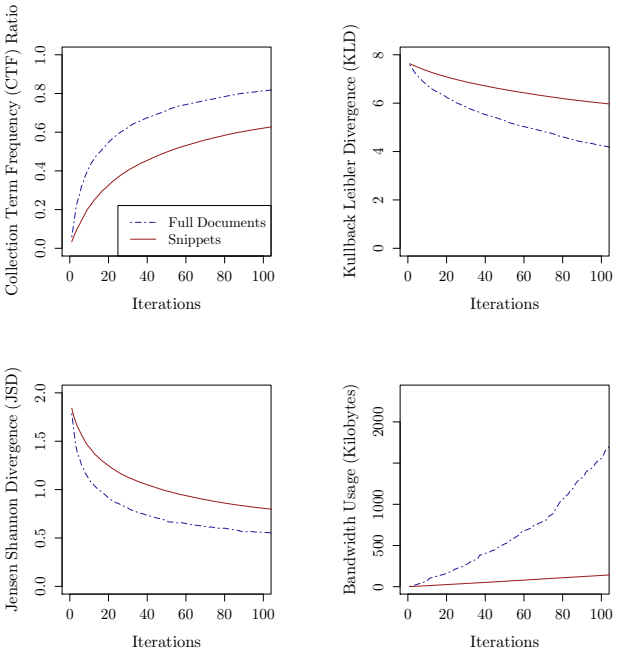

Figure 3: Results for TREC123. Shows CTF, KLD, JSD and bandwidth usage, plotted against the number of iterations. Shows both the full document and snippet-based approach. The legend is shown in the top left graph.

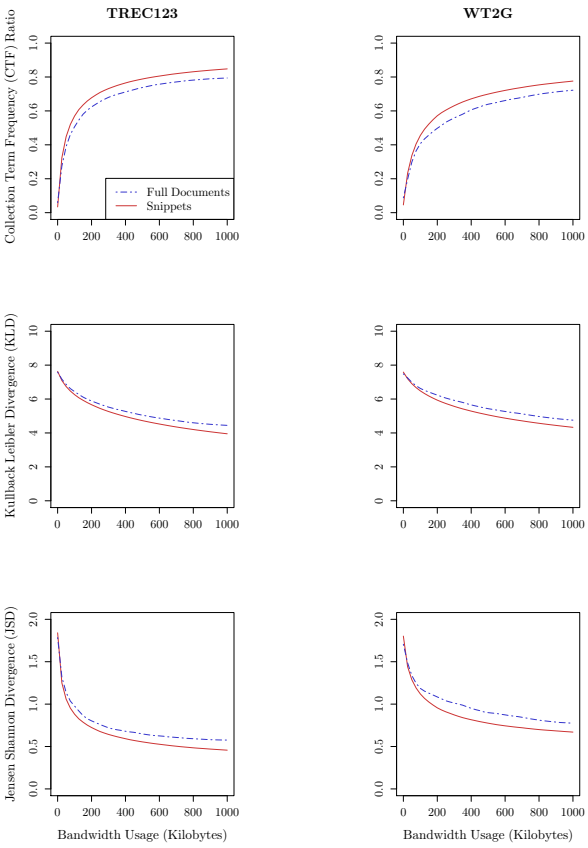

Figure 4: Interpolated plots for all metrics against bandwidth usage up to 1000 KB. The left graphs show results for TREC123, the right for WT2G. Axis titles are shown on the left and bottom graphs, the legend in the top left graph.

lot of extra information results in it outperforming the snippet approach for the defined metrics as shown in the other graphs of Figure 3. However, comparing this way is unfair. Full document sampling performs better, simply because it acquires more data in fewer iterations. A more interesting question is: how effectively do the approaches use bandwidth?

3.1 Bandwidth

Figures 4 and 5 show the metrics plotted against bandwidth usage. The graphs are 41-point interpolated plots based on experiment data. These plots are generated in a similar same way as recall-precision graphs, but they contain more points: 41 instead of 11, one every 25 kilobytes. Additionally, the recall-precision graphs, as frequently used in TREC, use the maximum value at each point [11]. We use linear interpolation instead which uses averages.

Figure 4 shows that snippets outperform the full document approach for all metrics. This seems to be more pronounced for WT2G. The underlying data reveals that snippets yield much more stable performance increments per unit of bandwidth. Partially, this is due to a larger quantity of queries. The poorer performance of 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.

TREC123 and WT2G are very large heterogeneous test collections as we will show later. The WIKI collections are more homogeneous and have different document length distribution characteristics. In Figure 5 we see that the performance of snippets on the WIKIL corpus is worse for the JSD, but undecided for the other metrics. For WIKIM performance measured with CTF is slightly better and undecided for the other metrics. Why this difference? We conducted tests on several other large size Wiki collections to verify our results. The results suggest that there is some relation between the distribution of document lengths and the performance of query-based sampling using snippets. In Figure 2 we see a peak at the low end of documents lengths for WIKIL. Collections that exhibit this type of peak all showed similar performance as WIKIL: snippets performing slightly worse especially for the JSD. In contrast, collections that have a distribution like WIKIM, also show similar performance: slightly better for CTF. Collections that have a less pronounced peak at higher document lengths, or a more gradual distribution appear to perform at least as good or better using snippets compared to full documents.

The reason for this is that as the document size decreases and approaches the snippet summary size, the full document strategy is less heavily penalised by mistakes. It can no longer download very large unrepresentative documents, only small ones. However, this advantage is offset if the document sizes equal the summary size. In that case the

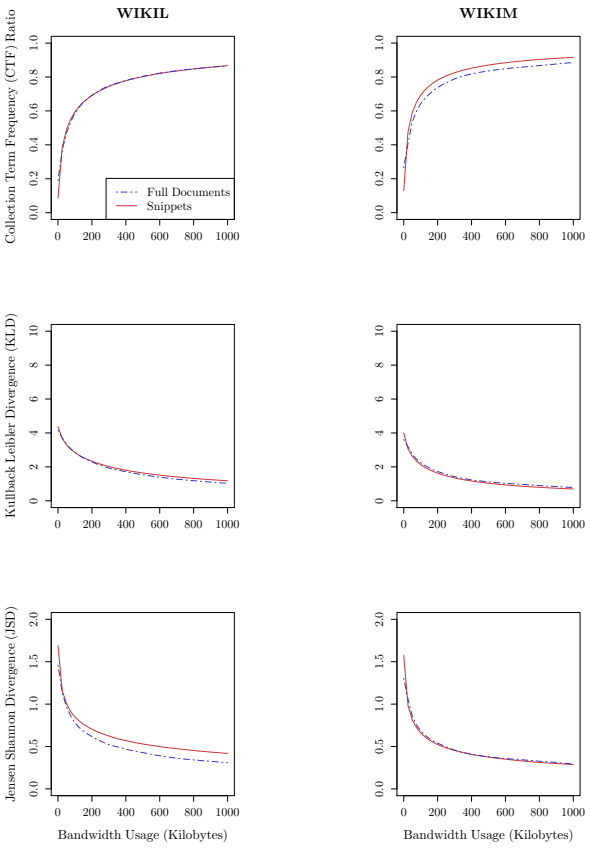

Figure 5: Interpolated plots for all metrics against bandwidth usage up to 1000 KB. The left graphs show results for WIKIL, the right for WIKIM. Axis titles are shown on the left and bottom graphs, the legend in the top left graph.

full document approach would actually use double the bandwidth with no advantage: once to obtain the search results, with summaries, and once again to download the entire documents which are the same as the summaries in the search results.

3.2 Homogeneity

While WIKIM has a fairly smooth document length distribution, the performance increase of snippets over full documents with regard to the JSD and KLD metrics is not the same as that obtained with TREC123 and WT2G. This is likely caused by the homogeneous nature of the collection. Consider that if a collection is highly homogeneous, only a few samples are needed to obtain a good representation. Every additional sample can only slightly improve such a model. In contrast, for a heterogeneous collection, each new sample can improve the model significantly.

So, how homogeneous are the collections that we used? We adopt the approach of Kilgariff and Rose [15] of splitting the corpus into parts and comparing those, with some slight adjustments. As metric we use the Jensen-Shannon Divergence (JSD) explained in Section 2.2 and also used by Eiron and McCurley [10] for the same task. The exact procedure we used is as follows:

Table 2: Collection homogeneity expressed as Jensen-Shannon Divergence (JSD): Lower scores indicate more homogeneity ($n = 100, \sigma = 0.01$).

Collection name	μ JSD
TREC123	1.11
WT2G	1.04
WIKIL	0.97
WIKIM	0.85

1. Select a random sample $\mathscr{S}$ of 5000 documents from a collection.
2. Randomly divide the documents in the sample $\mathscr{S}$ into ten bins: $s_1 \ldots s_{10}$. Each bin contains approximately 500 documents.
3. For each bin s_i calculate the Jensen-Shannon Divergence (JSD) between the *bigram* language model defined by the documents in bin s_i and the language model defined by the documents in the remaining nine bins. Meaning: the language model of documents in s_1 would be compared to that of those in $s_2 \ldots s_{10}$, et cetera. This is known as a leave-one-out test.
4. Average the ten JSD scores obtained in step 3. The outcome represents the homogeneity. The lower the number, the more self similarity within the corpus, thus the more homogeneous the corpus is.

Because we select documents from the collection randomly in step 1, we repeated the experiment ten times for each collection. Results are shown in Table 2.

Table 2 shows that the large collections we used, TREC123 and WT2G, are more heterogeneous compared to the smaller collections WIKIL and WIKIM. It appears that WIKIL is more heterogeneous than WIKIM, yet snippet-based sampling performs better on WIKIM. We conjecture that this is caused by the difference in document length distributions discussed earlier: see Figure 2. Overall, it appears that query-based sampling using snippets is better suited towards heterogeneous collections with a smooth distribution of document lengths.

4. CONCLUSION

We have shown that query-based sampling using snippets is a viable alternative for conventional query-based sampling using entire documents. This opens the way for distributed search systems that do not need to download documents at all, but instead solely operate by exchanging queries and search results. Few adjustments are needed to existing operational distributed information retrieval systems, that use a central server, as the remote search engines and the central server already exchange snippets. Our research implies that the significant overhead incurred by downloading documents in today's prototype distributed information retrieval systems can be completely eliminated. This also enables modeling of servers from which full documents can not be obtained and those which index multimedia content. Furthermore, the central server can continuously use the search result data, the snippets, to keep its resource descriptions up to date without imposing additional overhead, naturally coping with changes in document collections that occur over

time. This also provides the extra iterations that snippet query-based sampling requires without extra latency.

Compared to the conventional query-based sampling approach our snippet approach shows equal or better performance per unit of bandwidth consumed for most of the test collections. The performance also appears to be more stable per unit of bandwidth consumed. Factors influencing the performance are document length distribution and the homogeneity of the data. Snippet query-based sampling performs best when document lengths are smoothly distributed, without a large peak at the low-end of document sizes, and when the data is heterogeneous.

Even though the performance of snippet query-based sampling depends on the underlying collection, the information that is used always comes along 'for free' with search results. No extra bandwidth, connections or operations are required beyond simply sending a query and obtaining a list of search results. Herein lies the strength of the approach.

5. FUTURE WORK

We believe that the performance gains seen in the various metrics leads to improved selection and merging performance. However, this is something that could be further explored. A measure for how representative the resource descriptions obtained by sampling are for real-world usage would be very useful. This remains an open problem, also for full document sampling, even though some attempts have been made to solve it [4].

An other research direction is the snippets themselves. Firstly, how snippet generation affects modeling performance. Secondly, how a query can be generated from the snippets seen so far in more sophisticated ways. This could be done by attaching a different priority to different words in a snippet. Finally, 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 Dolf Trieschnigg, 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.

7. REFERENCES

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

[2] Bailey, P., Craswell, N., and Hawking, D. Engineering a multi-purpose test collection for web retrieval experiments. *Information Processing & Management 39*, 6 (2003), 853–871.

[3] 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.

[4] Baillie, M., Carman, M. J., and Crestani, F. A topic-based measure of resource description quality for distributed information retrieval. In *Proceedings of ECIR* (Apr. 2009), vol. 5478 of *Lecture Notes in Computer Science*, Springer, pp. 485–497.

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

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

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

[8] 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.

[9] 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.

[10] Eiron, N., and McCurley, K. S. Analysis of anchor text for web search. In *Proceedings of SIGIR* (New York, NY, US, July 2003), ACM, pp. 459–460.

[11] Harman, D. Overview of the first trec conference. In *Proceedings of SIGIR* (New York, NY, US, June 1993), ACM, pp. 36–47.

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

[13] 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.

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

[15] Kilgarriff, A., and Rose, T. Measures for corpus similarity and homogeneity. In *Proceedings of EMNLP* (Morristown, NJ, US, June 1998), ACL-SIGDAT, pp. 46–52.

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

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

[18] 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.

[19] Paltoglou, G., Salampasis, M., and Satratzemi, M. Hybrid results merging. In *Proceedings of CIKM* (New York, NY, US, Nov. 2007), ACM, pp. 321–330.

[20] Thomas, P., and Hawking, D. Evaluating sampling methods for uncooperative collections. In *Proceedings of SIGIR* (New York, NY, US, July 2007), ACM, pp. 503–510.

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