# NL Sampler - Random Sampling of Web Documents Based On Natural Language with Query Hit Estimation

Daniel Schuster, Alexander Schill

Chair of Computer Networks, Department of Computer Science, Technische Universität Dresden
Helmholtzstr. 10, 01062 Dresden, Germany

Daniel.Schuster@tu-dresden.de, Alexander.Schill@tu-dresden.de

## ABSTRACT

Random sampling of documents is a substantial supporting function for research in information science, content-related research (like content adaptation), or social sciences. Looking for an appropriate method to get a random sample of Microsoft Office files for research on presentation sharing applications, we found out, that the two main approaches Random Walk and Random Search are not appropriate to find formatted documents. Both approaches are designed for the purpose of large scale Web analysis and do not fit more special requirements.

In this paper, we adopt and extend the Random Search approach first described by Bharat and Broder to a more universal random sampling method based on natural language lexica called NL Sampler, that can be used in a wide range of application domains. It supports parameters like file type or DNS domain restrictions while preserving representativeness. We implemented and evaluated the approach and found a Zipf-like distribution of average hits per query which enables estimation of query hits for a certain set of parameters and thus can be used in a lot more application areas than the approaches previously published. Estimation functions are given for Microsoft Word and PowerPoint documents.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Clustering, Information filtering, Query formulation, Relevance feedback, Retrieval models, Search process, Selection process.*

## General Terms

Algorithms, Measurement, Performance, Experimentation.

## Keywords

Random Sampling, Search Engines, Content Analysis, World Wide Web.


*SAC'07*, March 11-15, 2007, Seoul, Korea.


## 1. INTRODUCTION

With the exponential growth of the Web in a means of both the total amount of accessible data as well as the amount of useful information, there has always been a strong interest in measuring the Web and its contents. Thus, the research question which drove the development of methods for Web random sampling was to compare the index size and quality of search engines like Google, Yahoo!, or MSN Search [1]. It is not possible to crawl the whole Web for this purpose, so naturally there is a need for random sampling of documents from the Web. The two main approaches of Web random sampling, i.e. using random search engine queries (Random Search) [2] or crawling the Web with random selection of links (Random Walk) [3] are based on this question. Such questions and more specific Web-related information science problems are subsumed under the term "webometrics" [4].

Besides this, there is a keen interest in random sampling in the field of content-related research such as content networking and content adaptation. E.g., Chandra [5] gathered image files from major websites to analyze the possibilities of scaling and compression of images for mobile Web access. De Lara et. al. [6] downloaded 12,500 Microsoft Office documents from 935 websites to explore possibilities for document adaptation.

We faced a similar problem when looking for an appropriate method to get a representative sample of Microsoft PowerPoint and pdf files with the purpose to convert and analyze them for research on presentation sharing applications [7]. Simply downloading files from different sites does not ensure the representativeness of the sample. Thus we try to answer the following research question: can we sample documents from a small subset of the documents available in the static Web while preserving representativeness?

We tried both the Random Walk as well as the Random Search approach to get such a sample. But a Random Walk does not find enough formatted documents as they are scarce compared to HTML pages. The Random Search approach has the same problems, as it is very unlikely to find a formatted document with a conjunctive random query of words from a large lexicon of words found by a Web crawl. Bar-Yossef and Gurevich recently proposed to use a pool of exact phrases instead [8], but this also doesn't solve the problem of how to sample within a small subset of Web documents.

Based on a detailed requirements analysis, we found out, that the Random Search approach can form a basis for such a sampling method, although it has the disadvantage that it can only find

documents containing text. Random sampling of images is thus not possible with this method.

Unlike existing methods, our approach uses natural language lexica ordered by frequency of occurrence to form random queries. This enables the flexible adjustment of input parameters (words per query, number of words to use from the lexicon) by query hit estimation to get a sufficiently large average number of query hits. We found a direct Zipf-like [9] relationship between these parameters and the number of query hits as returned by the search engine. Query hit estimation functions for Microsoft Word and PowerPoint documents are presented as the result of evaluation and analysis. Thus we are able to sample even scarce documents such as formatted documents or documents from certain domains.

We implemented the NLSample tool which we plan to make available as open source software. Besides sampling of text-based documents, it also downloads directly linked images in HTML pages and can thus also be used for random sampling of images.

The rest of the paper is organized as follows: In Section 2, we define requirements for random sampling of Web documents that consider both the area of webometrics as well as content-related research. Existing approaches are compared in Section 3 with respect to these requirements. Our approach is shown in Section 4 and evaluated in Section 5. We conclude the paper in Section 6 and discuss future research directions.

## 2. REQUIREMENTS

As mentioned above, a universal method for Web random sampling should be used in the domain of webometrics as well as for content-related research and social sciences. There are different possible application scenarios such as measuring search engine indexes, gathering content samples of images or formatted documents, or get a sample of documents from one or more specific domains and/or documents that contain special keywords such as product names. We thus define requirements for a universal sampling method that fits these application scenarios:

1. Universal – The method should support the sampling of as many file types as possible, especially markup files (html, xml), images (jpeg, png, gif), and formatted documents (pdf, doc, ppt, OpenOffice files).
2. Configurable - The procedure must be able to specialize on a certain portion of the Web, e.g. the .com top-level domain or any other set of hosts. Furthermore you should be able to specify keywords that have to be in the sampled documents.
3. Correct - The method has to work correctly, i.e. produce a representative and uniformly distributed random sample. Furthermore, the method should have a mathematical background, so that theoretical correctness of the approach can be proven considering e.g. a simplified graph model of the Web.
4. Repeatable - Every researcher with a keen interest in random sampling must be able to run the sampling procedure and produce repeatable results.
5. Economical - The sampling procedure must be economical to use, i.e. it has to consume few resources like processing time or network bandwidth.

The first functional requirement of universality (Req 1) is the primary motivation of our research. Existing methods are not appropriate to sample formatted Web documents, so we want to extend these methods to be universal.

Many studies do not cover the whole Web but specialize on a certain territory like commercial websites or all websites of a country only. Thus the sampling method has to be configurable (Req 2). More generally speaking, there has to be a flexible set of parameters for the sampling method.

As every research work has to be accountable, the requirement of correctness (Req 3) can be inferred consequently. This also leads to the requirement of repeatability (Req 4).

According to [10] the best sampling method would be to crawl and index all pages in the Web, and then select a page at random from the created index. As this is not possible due to resource limitations and the messy structure and size of the Web, the method has to be economical to use (Req 5). Random Walks are a first step in this direction, but still require several hundred of parallel crawling threads to achieve appropriate results. The Random Search approach is much more economical as it only requires one or two search engine queries per document in the sample.

## 3. EXISTING SAMPLING METHODS

The problem of random sampling from the Web has already been tackled in a number of research efforts. As mentioned above, the two main approaches are random walks on the Web and search engine random queries. These two and other approaches are described in the following with respect to the requirements specified in Section 2.

Random Search and Random Walk suffer a number of limitations, which can be found in the corresponding literature [2] [3] [10] [11]. Common to both methods are the limitation to static and public Web pages and the experimental bias, as the Web changes during the sampling.

### 3.1 Random Walk

The random walk approach was first proposed for Web random sampling by Henzinger et al. [3]. The theoretical background is to see the Web as a directed graph where Web pages as nodes are connected by links as edges of the graph.

The main idea behind the random walk approach is to model the typical behavior of a Web surfer. In this model, the surfer starts at a randomly chosen page and then chooses a link uniformly at random from the outlinks of each page he visits. Occasionally, he does a random restart by choosing a page at random from the Web again to avoid cycles. This ought to be the case after 5 to 10 steps. Hence, a Markovian random walk on the Web graph is performed, which means that the decision only depends on the current state and not on previous states, too.

This simple model is also used as the basis for the PageRank measure [12]. A site that is potentially visited more often by such an idealized Web surfer is assigned a higher PageRank. This introduces a bias to a sampling method based on this model, as a random sample of pages is usually not uniform but PageRank distributed.

Nevertheless, near-uniform distributed samples can be fetched using Random Walk when the visited pages are not added

uniformly at random to the sample but with a propability approximately inversely proportional to their PageRank [10] [11]. This increases complexity and need for resources as at least medium-scale Web crawling (100 or more threads) is needed to achieve appropriate results.

Concerning our requirements, the Random Walk approach clearly fulfills the requirement of correctness (Req 3), as there is the mathematical model of the random Web surfer. It is also repeatable (Req 4), although this often requires a large amount of resources. Regarding Req 2 (configurable), the search for documents with keywords or from special domains always requires a Random Walk over the whole Web to be representative. Afterwards, documents that are not of interest have to be filtered out. For special parameters the resulting sample might contain an insufficient number of documents. For the same reason the method is not appropriate for arbitrary file types (Req 1), because they are scarce compared to HTML pages. After all, the method is not economical to use (Req 5), because it requires multi-threaded crawling of Web pages which is often not possible due to resource limitations.

## 3.2 Random Search

As stated above, the ideal random sampling technique would be based on a large crawl of the entire Web, that indexes all pages found and so a sample of pages could easily be chosen uniformly at random from this index. Search engines maintain such an index of pages, but they do not allow direct access to it.

Bharat and Broder [2] used random queries to overcome this problem. They built up a lexicon of words from a Web crawl starting at yahoo.com that contains 400,000 words and their respective frequencies. Then they constructed random queries out of the lexicon and selected an URL at random from the first 100 results returned by the search engine. In the following, we refer to this method as the *BB Sampler*.

Besides the well-known limitations of the BB Sampler as mentioned in [2], there remain some open questions. Only two words are used in conjunctive queries and are selected according to their estimated frequency to get sufficient search results. But no data is reported how this affects the distribution of the sample.

Bar-Yossef and Gurevich recently proposed to use a pool of exact 5-term phrases instead of conjunctive or disjunctive combination of search words – the Pool-based Sampler (*PB Sampler*) [8]. Their primary focus was to overcome what they call *large documents bias*. The original Random Search approach tends to prefer large, content-rich documents as they contain more words indexed by the search engine. Another bias they deal with is the *unrelated words bias*, i.e. if words are combined that are completely unrelated, only documents that are actually word lexica can be found by the method.

As we also use conjunctive search phrases, the bias of our method towards large documents is the same as the bias of the BB Sampler. But unlike the two Random Search approaches mentioned, we use word lists of most frequently used words of specific languages and have a more limited lexicon size. This reduces the possibility of unrelated words. Furthermore, the unrelated words bias can be reduced even more if less words from the lexicon are used which is one of the input parameters of our method. While the PB Sampler is obviously better in reducing the unrelated words bias, it is not appropriate to sample formatted documents as this would produce massive *query underflows* (i.e. 0 hits are returned by the search engine).

Regarding the requirements in Section 2, the BB Sampler, the PB Sampler, as well as our Sampler (the NL Sampler) are correct (Req 3), assuming the lexicon contains a collection of words representative for the basic population of documents indexed by the search engine. Their strongest points are repeatability (Req 4) and economical use of resources (Req 5). They need only a small number (often 1) of search requests per object in the sample. While it can be argued that the search engine consumes much more resources indexing all the Web pages, Req 5 should be refined to represent only the researcher's point of view.

The user also has the possibility to enhance the random search with own parameters such as "filetype:pdf" or "site:stanford.edu" and thus have a near-universal (Req 1) and configurable (Req 2) method, but restricted to documents containing text. In the case of additional parameters, the average number of hits has to be increased to avoid query underflows. This can only be done with our NL Sampler, as it introduces estimation functions for the average number of hits per search engine query. This is only possible with natural language lexica.

## 3.3 Other Approaches

An overview of other approaches for random sampling of Web sites can be found in [4]. A method often used in content networking is to crawl selected documents of one or more hosts as used in [5]. This may be desirable for certain research questions, but we also need a sampling method which incorporates the whole Web.

Log file analysis [13] suffers much the same problems. Web server logs and web proxy logs each only cover a small portion of the Web with special distributions of requests. As an example, Web proxies of academic institutions are tended to give a preference towards research-related Web sites.

Besides this, there have been considerations to generate random IP addresses [14] or URLs [15]. But both approaches have proven to be of little usefulness as can be seen in [4].

# 4. NL SAMPLER

The architecture of the NL Sampler is shown in Figure 1. It consists of the 3 components Randomizer, Search Engine Agent, and Download Agent. The components are chained together to get the intended result but work independently from each other.

A sampling transaction consists of several random search queries with the same parameters but different random words. Parameters for the transaction are the number of documents to sample and/or the maximum number of search engine requests to use for this purpose. A word frequency ordered natural language word lexicon (like the 10,000 most frequently used words in English) has to be selected and additional parameters like file type, domain restriction, or keywords may be given. The result of the sampling transaction is a random sample of documents and several log files.

In a first step, the *Randomizer* takes as input the lexicon and the randomizer parameters i.e. the number of random search words in each query and the span of words to use from the lexicon. The Randomizer randomly selects words from the lexicon (using a uniform distribution) and eliminates double hits, so that random search phrases are constructed. As already mentioned, the lexicon

has to be ordered by frequency of occurrence in natural language. This is essential for the query hit estimation described later in Section 5.

This difference to the BB and PB Sampler has big impact on the selection of documents. The random selection is now based on natural language and as such separated from characteristics of the basic population.

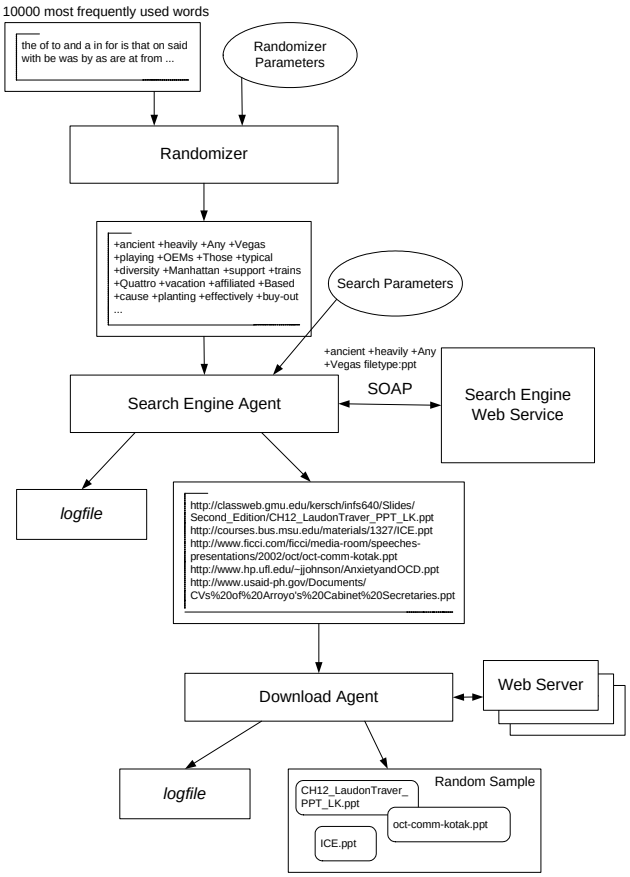


**Figure 1: NL Sampler architecture.**

The *Search Engine Agent* combines each random search phrase with the search parameters like domain or file type restrictions. An URI is randomly selected out of the search results using a uniform distribution. As ordinary search engine queries are used for this purpose, the behavior of the sampler can easily be adapted to the user's needs. E.g., the NL Sampler gathers a sample from a search engine query like "groupware filetype:pdf", which has approximately 500,000 hits. The last 499,000 of them are not accessible, as search engines restrict search results to the first 1,000 hits.

The Search Engine Agent now repeats the search and changes the randomly generated part for each request. Thus, web pages are randomly selected out of the set of web pages specified by the search parameters. Thus, we get a sample of pdf files containing the word "groupware" out of the 500,000 available documents.

Random sampling of documents without additional search words generates a sample out of the basic population of all Web documents accessible through the search engine interface. As most search engines do not allow automated access of their Web interfaces anymore, we use search engine Web Services like the Google APIs [16] instead.

Once the URLs are sampled, they can be downloaded by a *Download Agent* like HTTrack [17] which supports downloading of formatted documents as well as HTML pages with associated images.

# 5. IMPLEMENTATION AND EVALUATION

## 5.1 The NLSample system

We implemented the NL Sampler using C# and SharpDevelop. The NLSample tool uses the Google Search APIs and the lexica of the "Wortschatz" project of Leipzig University [18]. This project offers lists of the 10,000 most frequently used words in English, German, Dutch, and French. Thus the system is able to search for documents in these 4 languages.

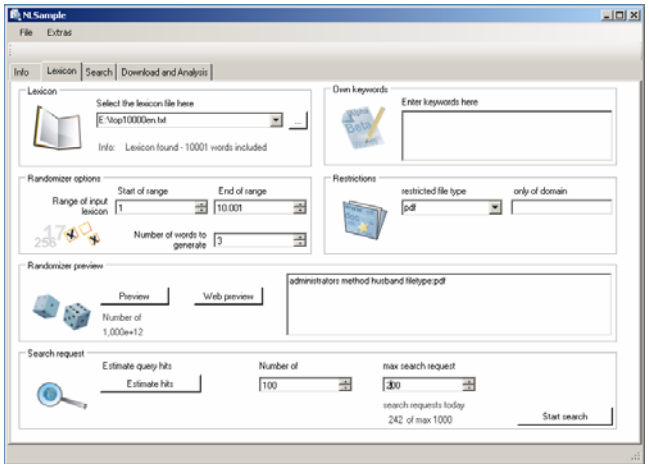


**Figure 2: Screenshot 1 - input parameters.**

Figure 2 shows the start screen of the system where the user can insert the Randomizer parameters and search parameters. A lexicon must be selected and the word span to use from this lexicon has to be specified. In the example, we look for a random sample of pdf documents in English. All 10,000 words of the lexicon are used and each random query consists of 3 words that are combined to a conjunctive search phrase. We specify to find 100 documents by using a maximum of 200 search engine requests (some requests may return no result). The number of requests is limited to 1,000 per day per user by the usage policy of Google Search APIs.

Then the sampling can be started as shown in Figure 3. URIs from the search results are grouped by domains of the host part of the URI. Once these requests are finished, a selection of the documents can be downloaded by the integrated download agent and thus form the actual random sample. For each sample a log file is written which contains the random search request, original URI and the number of total documents for each query.

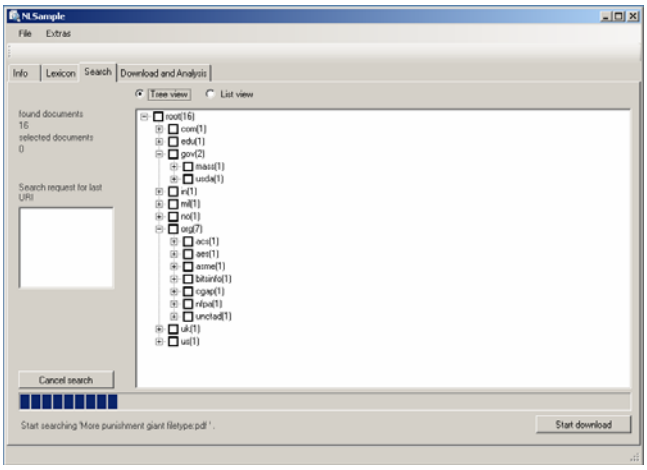


**Figure 3: Screenshot 2 - the sampling process.**

## 5.2 Evaluation

We evaluated the system by performing sampling transactions for "filetype:ppt" and "filetype:doc" over a period of 2 months. Each sampling transaction consisted of 200 to 1,000 queries. We analyzed the average number of hits returned by each query dependent on the number of search words in each query and the number of words used from the lexicon.

We observed that it is difficult for users to know which randomizer parameters to choose for certain search parameters to get appropriate results. If users want to have a uniform distribution of documents in the sample, they should use as many random search words per query as possible. On the other hand, if they specified special search queries with file type and domain restrictions, even 2 additional random search words may result in query underflow. Besides reducing the number of search words, it is also possible to reduce the number of words to use from the lexicon.

**Definitions.**

We adopt the formal setup of [8] and define $Q$ as the space of queries supported by the search engine. EVAL(q) is an evaluation function, which maps every query $\mathbf{q} \in Q$ to a result set $R_q$, i.e. an ordered sequence of documents. The *volume* of **q** is the number of results for this search query: VOL(q) = $| R_q |$. We further define the transaction $T$ as a set of queries **q** with the same search parameters (but different randomizer parameters). We define the volume of $T$ VOL($T$) as the number of documents accessible with any combination of randomizer parameters. We further define HITS($T$) as the average volume of all queries in this transaction:

$$HITS(T) = \frac{\sum VOL(q)}{|T|}, q \in T \qquad (1)$$

The volume of **q** is estimated by the search engine on every request and returned in the result. Our goal is to find a function which takes the parameters as input and estimates the HITS($T$) value.

**Experiment results.**

Figure 4 shows the results of various test runs with the search parameters "filetype:ppt", i.e. with the purpose to sample Microsoft PowerPoint documents out of all static documents indexed by Google. Each point in the diagram corresponds to a run with 200 queries and certain randomizer parameters. The randomizer parameters were modified for each next run to identify the influence on HITS($T$).

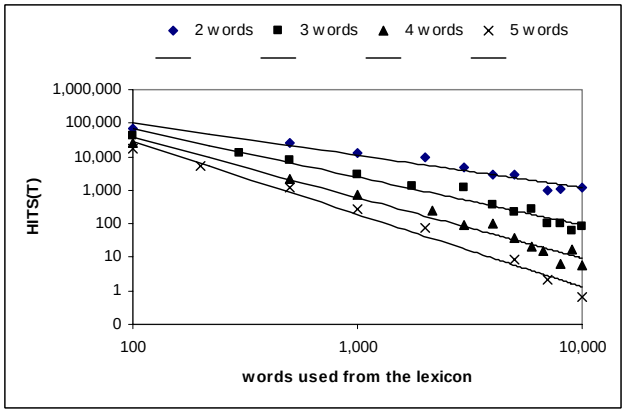


**Figure 4: HITS($T$) of "ppt" dependent on words used from the lexicon.**

As it is shown in the diagram, we have a Zipf-like distribution for HITS($T$) which can be seen in the near-linear regression lines in the log-log scale for 2, 3, 4, and 5 search words dependent on the number of words used from the lexicon. The original meaning of Zipf's law as stated in [9] was the ranking of words against their frequency of occurrence in natural language. As we stated in Section 4, the input lexicon of the NL Sampler should be in natural language and ordered by this frequency. So the input lexicon shows a Zipf distribution. This distribution of the input lexicon results in a Zipf-like distribution of HITS($T$) as shown above.

It is not surprising that the more popular search words we use (i.e. less words from the lexicon) the more documents we find using these words. But the fact, that the resulting distribution of average search engine hits is Zipf-like gives us the crucial starting-point for query hit estimation.

It has already been proven, that website popularity also follows a Zipf-like distribution [19]. But this has no influence on the distribution of HITS($T$). The (estimated) total number of websites returned by the search engine is not related in any way to the mean or maximum popularity of these sites. Only the ranking within the result set depends on popularity (e.g. PageRank).

## 5.3 Query Hit Estimation

To get an estimating function for HITS($T$), we have to divide the parameters in randomizer parameters and search parameters. The search parameters are a search string like "groupware filetype:pdf" which can be sent as a query to the Google Web Service. The result returns the estimated total number of documents for the query. We assume that this is equal to VOL($T$). For search parameters without search words like "filetype:pdf" we use a query with the character "e" to get the volume of $T$. E.g., a search request "e filetype:pdf" returns approximately between 250 million and 300 million hits.

The randomizer parameters constitute the number of possible random search queries which is equivalent to the combinations of search words in the lexicon and words in each query. Thus we define:

$$C_q = \left( \frac{sizeoflexicon}{wordsperquery} \right) \quad (2)$$

as the number of possible combinations of words per query and number of words to use from the input lexicon (i.e. size of the lexicon).

We are now able to summarize both types of parameters in a characteristic value,

$$C_d = \frac{C_q}{VOL(T)} \quad (3)$$

where $C_d$ is the number of possible query combinations per document in the basic population that is accessible with the search parameters of the transaction.

To find the influence of $C_d$ on HITS($T$) we print the same data as in Figure 4 but now dependent on $C_d$ (Figure 5).

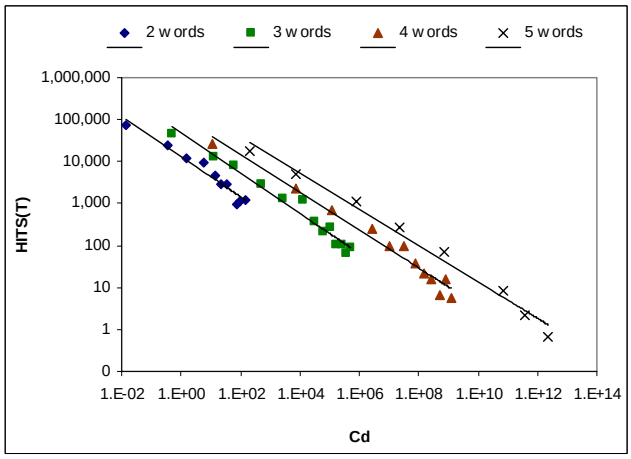


**Figure 5: HITS($T$) of "ppt" dependent on $C_d$.**

As can be seen in the diagram, we now have almost parallel linear regression lines. Parallel lines in this scale can be mapped onto each other by multiplication. We found out, that this factor is the reciprocal of *wordsperquery* to the power of 10. This leads us to the definition of the weighted number of query combinations per document ($W_d$):

$$W_d = \frac{1}{10^{wordsperquery}} \cdot C_d \quad (4)$$

All influence of the randomizer parameters can be modeled by $W_d$. Figure 6 shows this as all runs of the first experiment series are now on one regression line. Additionally a second series of experiments has been carried out with the search parameters "filetype:doc". This leads to another estimating function which can be seen in Figure 6. The estimating functions for these two file types are specified in the diagram and show a correlation bigger than 95 %. We thus have identified estimating functions for these two file types which can be easily extended to other file types by further experiments.

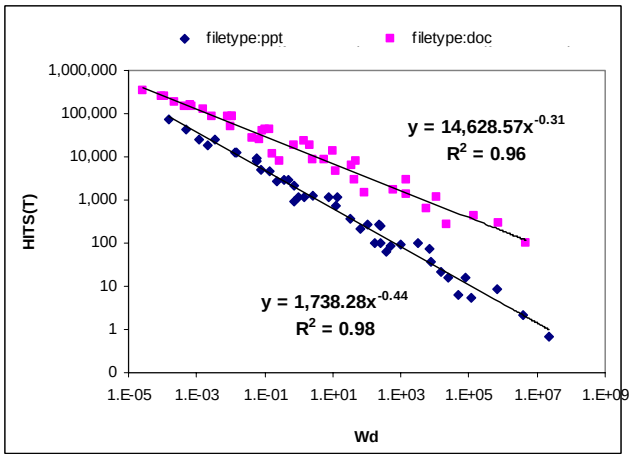


**Figure 6: HITS($T$) of "ppt" and "doc" dependent on $W_d$.**

## 6. CONCLUSIONS

We surveyed related works in the field of random sampling of Web documents and revisited the random search approach first published by Bharat and Broder. The NL Sampler described in this paper is an extension of the random search approach to get random samples of Web documents with a flexible set of parameters. It is now possible to sample any document type from the Web that contains text while consuming only few resources. The sampling of images is also possible by downloading the images linked in sampled HTML pages. We have implemented the NL Sampler within the NLSample system and plan to make this available as an open source tool for scientific and private use.

Unlike previous attempts, we use frequency-ordered natural language lexica which enables query hit estimation. Estimation functions for Microsoft PowerPoint and Word files are given in the paper based on experimentation and analysis. The NLSample tool can be used in different application domains like content-related research, webometrics, marketing research, or extended Web search. Due to new policies of search engines, Web services APIs have to be used for random search.

A quantitative comparison of our approach with the BB and PB sampler or the Random Walk approach has not yet been carried out but will be part of our future work, although the goal of our work is different from these previous works. These are still the reference methods to sample representative samples out of all web pages while our approach is best suited to find samples out of small subsets of Web documents such as formatted documents.

Besides this, more experiments are needed in the future to refine query hit estimation. This has already been done for the search parameters "filetype:ppt" and "filetype:doc", which means random sampling of ppt- and doc-files out of all such documents indexed by Google. It would be interesting to see how other search parameters influence query hit estimation.

## 7. REFERENCES


[1] Sullivan, S., Sherman, C., Search Engine Watch Website, searchenginewatch.com, 2006.

[2] Bharat, K., Broder, A., A technique for measuring the relative size and overlap of public Web search engines, 7th International World Wide Web Conference (WWW7), Brisbane, Australia, 1998.

[3] Henzinger, M., Heydon, A., Mitzenmacher, M., Najork, M., Measuring Index Quality using Random Walks on the Web, 8th International World Wide Web Conference, WWW8, Toronto, Canada, 1999.

[4] Thelwall, M. V. L., Björneborn, L., Webometrics, Annual Review of Information Science and Technology 39, 2005.

[5] Chandra, S., Gehani, A., Ellis, C. S., Vahdat, A., Transcoding Characteristics of Web Images, Multimedia Computing and Networking (MMCN'01), San Jose, CA, USA, 2001.

[6] de Lara, E., Wallach, D. S., Zwaenepoel, W., Opportunities for Bandwidth Adaptation in Microsoft Office Documents, 4th Usenix Windows Systems Symposium, Seattle, Washington, USA, 2000.

[7] Schuster, D., Kuemmel, S., Towards Adaptive Distribution of Multimedia Content within Collaborative Conferencing Sessions, 9th International Conference on Internet and Multimedia Systems and Applications, Honolulu, Hawaii, 2005.

[8] Bar-Yossef, Z., Gurevich, M., Random Sampling from a Search Engine's Index, World Wide Web Conference (WWW 2006), Edinburgh, Scotland, 2006.

[9] Zipf, G. K., Human Behavior and the Principle of Least Effort, Addison Wesley, Cambridge, MA, USA, 1949.

[10] Henzinger, M. R., Heydon, A., Mitzenmacher, M., Najork, M., On Near-Uniform URL Sampling, 9th International World Wide Web Conference, Amsterdam, The Netherlands, 2000.

[11] Rusmevichientong, P., Penncock, D. M., Lawrence, S., Giles, C. L., Methods for Sampling Pages Uniformly from the World Wide Web, AAAI 2001 Fall Symposium on Using Uncertainty within Computation, North Falmouth, MA, USA, 2001.

[12] Page, L., Brin, S., Motwani, R., Winograd, T., The PageRank Citation Ranking: Bringing Order to the Web, Stanford University Database Group, Stanford, CA, USA, 1998.

[13] Mahanti, A., Williamson, C., Eager, D., Traffic Analysis of a Web Proxy Caching Hierarchy, IEEE Network Magazine, Special Issue on Web Performance, May/June 2000, 2000.

[14] Lawrence, S., Giles, C. L., Accessibility of information on the Web, intelligence, 11/1, pp. 32-39, 2000.

[15] Thelwall, M., Commercial web sites: Lost in cyberspace? Internet Research: Electronic Networking and Applications, 10/2, pp. 150-159, 2000.

[16] Google, Inc., Google APIs, http://www.google.com/apis/, 2006.

[17] Roche, X., et. al, HTTrack Website Copier, http://www.httrack.com/, 2006.

[18] Universität Leipzig, Word lists of wortschatz project, http://wortschatz.uni-leipzig.de/html/wliste.html, 2006.

[19] Nielsen, J., Zipf Curves and Website Popularity, http://www.useit.com/alertbox/zipf.html, 1997.