A Comparison of Techniques for Sampling Web Pages

Eda Baykan
Ecole Polytechnique Fédéral de Lausanne (EPFL)
eda.baykan@epfl.ch

Sebastian de Castelberg
University of Applied Science Rapperswill
sdecaste@hsr.ch

Monika Henzinger
Ecole Polytechnique Fédéral de Lausanne (EPFL)
monika.henzinger@epfl.ch

ABSTRACT

As the World Wide Web is growing rapidly, it is getting increasingly challenging to gather representative information about it. Instead of crawling the web exhaustively one has to resort to other techniques like randomly sampling to determine the properties of the web. Unfortunately, no approach has been shown to sample the web pages in an unbiased way. Three promising web sampling algorithms are based on random walks [6, 2, 9]. They each have been evaluated individually, but on different data sets so that a comparison is not possible. In this paper we compare these algorithms by running them on the web with the same computation power and for the same amount of time. We then propose improvements based on experimental results.

Keywords

URL sampling, Random walks, PageRank, Information gathering from the web.

1. INTRODUCTION

The information accessible on the World Wide Web is increasing rapidly. This makes the web a rich resource for information gathering. However it is challenging to gather representative information about the web itself because of its huge size. To determine various statistics on the web like the percentage of pages in a given domain, we need effective tools to sample the web uniformly at random. Note that our goal is to have a random sample of *web pages* not a sample of the graph structure of the web.

In the literature there are three major approaches for sampling web pages and one approach for sampling hosts: Lawrence and Giles [8] tested random IP addresses to determine characteristics of hosts. However, it leaves the question open how to deal with multiple host sharing the same IP address or hosts being spread over multiple IP addresses. Additionally, it is not clear how to sample the pages accessible at a given IP address.

Bar-Yossef et al. [2] and Henzinger et al. [5, 6] proposed independently to use random walks on the web to sample web pages. Both papers present algorithms that in theory should lead to uniform random samples but that cannot be implemented in their "pure" form. Instead, the implementation needs to make some simplifications which causes various biases in the resulting samples. Basing on [2], authors of [9] proposed a different random walk which in theory should lead to uniform random samples. This walk can be implemented without any simplifications. Following the affiliation of the authors we denote in the rest of the paper the Bar-Yossef et al. algorithm as the *Berkeley algorithm*, the Henzinger et al. algorithm as the *Compaq algorithm* and the Rusmevichientong et al. algorithm as the *NEC algorithm.* In this paper we evaluate the samples generated by these algorithms after running them with the same computation power and with the same amount of time *on the web.*

Our experiments show the following new results: (1) Both the Berkeley and the NEC approach lead to a strong bias to highly connected hosts with few outedges. Over 20% of the pages in the Berkeley and in the NEC samples belong to the same host. The problem with *getting stuck* on hosts has not been shown before for these algorithms. Due to random resets the Compaq walk never got stuck. Adding random resets to the Berkeley and to the NEC algorithm should resolve this problem. (2) The Compaq samples exhibit a bias towards high outdegree nodes *on the web.* This was shown before for artificially generated graphs [9] but not for the web. (3) We experimented with different variants (subsampling techniques) for each algorithm, but we observed that they had a very small impact on the resulting samples. (4) Even though the NEC algorithm has a stronger theoretical justification, it does not seem to perform better in practice.

This paper is organised as follows: Section 2 describes the evaluated algorithms and their corresponding sampling techniques. Section 3 presents the experiments on the web and their results. We conclude with proposals for improving the algorithms in Section 4.

2. DESCRIPTION OF THE ALGORITHMS

We define the *web graph* as a graph where every web page is a node and every hyperlink is a directed edge between nodes. A memoryless random walk on the web graph is a Markovian chain that visits a sequence of nodes where transitions between nodes depend only on the last node of the walk and not on earlier nodes. Each visit to a node is called a *state* and results in one *step* progress of the walk.

All three algorithms consist of two phases: (1) A *walk phase*, where a memoryless random walk is performed on the web graph. To speed up both algorithms we use multiple threads in parallel which share the database. Thus, we are not running one random walk, but multiple random walks in parallel starting from the same initial state. These walks are not completely independent of each other since they share the database. However, the shared database only makes sure

.

that all threads "see the same graph", i.e., that the edges adjacent to a node remain same throughout all the walks[1]. Thus, we do not believe that the sharing of the database has negative effects on the quality of the samples returned by the algorithms.

During the walks we handle redirections and session ids as follows. When fetching a page the walk first follows all the redirects that it can follow and if the final URL in the redirect chain contains a question mark, it is truncated. If truncation leads to an error page or a new HTTP redirect, the walk undoes the truncation. In either case the inlinks of all the nodes in the redirect chain are added to the inlinks of the final URL if this URL was not visited before.

(2) A *sampling phase*, where the nodes that are visited by the walk are subsampled in a possibly biased way. The resulting set of nodes is considered a random sample of the web.

While sampling we ignore threads that had to be stopped during the crawl. If a thread tries to fetch URLs on the same host consecutively more than $3,000$ times, we put the walk to sleep for 20 minutes to avoid host overload. If this happens 12 times consecutively on the same host, we stop the walk as it seems to have a hard time leaving the host. We call this *getting stuck* in the host. This happened only for the NEC and the Berkeley algorithm.

All the samples consist of around $10,000$ web pages. Note that the Berkeley and the Compaq samples include unique web pages on the otherhand the NEC samples can include the same web page more than once.

2.1 Compaq Algorithm

Walk Phase: The Compaq algorithm tries to imitate the PageRank random walk [3] as closely as possible. During the walk the algorithm keeps track of all the visited pages as well as their outlinks. We call the visited pages and the URLs in their corresponding outlinks the *seen pages*. *Seen hosts* and *seen domains* can be defined accordingly. Note that in this paper we denote by *domain* the second level domain like `epfl.ch` or `berkeley.edu`.

Let S be the set of all seen pages. When choosing the next node to visit, the Compaq walk first flips a biased coin. With probability $d = 1/7$ it selects a random node out of S as next state. This is called a *random reset* or *random jump*. In our implementation the algorithm first selects a domain uniformly at random from all the seen domains, then it selects a host uniformly at random from all the seen hosts in that domain, and finally it selects a page uniformly at random from all the seen pages in that host. Note that this is different from the implementation of [5] which did not use domains in the random reset phase and which used the set of visited pages as S. Since the set of seen pages (about 40 million) is on the average a factor of roughly 40 larger than the set of visited pages (about 1 million), this modification allows us to more closely imitate the PageRank random walk which used S as the set of all the web pages.

With probability $(1 - d)$ the Compaq Algorithm chooses an outedge of the current node uniformly at random and selects the head of that edge as next state. If the chosen web page has no outlinks or it cannot be fetched, a random jump is performed.

Sampling Phase: Following [6] we use three different techniques to sample the Compaq walk. One is simply a random sample of all the unique visited nodes, called the *Random Sample*. However this will be biased towards high PageRank pages as they are more likely to be visited. The other two sampling techniques try to correct for this bias. The idea is to sample inversely proportional to PageRank. Since the PageRank of the whole web is not known, a *PageRank substitute* is used in the sampling. It is computed in one of two possible ways: (1) The PageRank of the subgraph of seen nodes during the crawl is computed and the unique visited pages are sampled inversely proportional to their PageRank values. This is called the *PR Sample*. (2) The *VisitRatio*, the ratio of the number of visits of a page to the number of steps in the walk is computed. The unique visited nodes are sampled with probability inversely proportional to their VisitRatio values. This is called the *VR Sample*.

[1]Since the web keeps on changing it is possible to find different links to a web page at subsequent visits.

2.2 Berkeley and NEC Algorithm

Both the Berkeley and the NEC algorithm make an undirected random walk on the web. The most important difference between these algorithms lies in the sampling phase. In the NEC walk the probability of visiting a page is proportional to that page's (degree+1) on the other hand in the Berkeley walk each page has an equal opportunity of being visited if run long enough.

Walk Phase: Consider the following random walk, on an *undirected graph*: From the current node choose an adjacent edge uniformly at random and select the other endpoint of the edge as next node. It can be proved that this random walk converges on a regular, undirected graph to a uniform distribution if run long enough. The web graph is neither undirected nor regular. To deal with the regularity the Berkeley algorithm performs the walk on a modified web graph by adding enough self loops to each node to increase its degree to max. Following [2] we set $max = 10,000,000$. For the NEC walk only one self loop is added to each node. To make the graph undirected the direction of the links is simply ignored. In the implementation this means that the inlinks as well as the outlinks of a page need to be determined when it is visited. This is done the first time a page is visited and the found links are stored in the database. At every further visit the links for that node are taken from database. As in the Compaq algorithm the outlinks are found by fetching the page. The inlinks are determined by querying a web search engine and retrieving the first up to 10 inlinks. This follows the approach in [2], where it was shown that it is useful to limit the number of inlinks found from a search engine. We did not ask the search engine for more inlinks as retrieving the inlinks is a slow operation. Inlinks from previously visited nodes of the walk are added to the ones retrieved from the search engine. We call the other endpoints of the inlinks and outlinks of a node the *siblings* of that node. As next state the Berkeley/NEC algorithm randomly chooses a sibling of a node or the node itself. Every node except the start node has always at least one sibling which can be visited, namely the node visited immediately before the current node. If a selected sibling cannot be fetched, a different sibling is selected uniformly at random as next state.

The number of self loops is the only difference in the walk phase between the Berkeley and the NEC algorithm at the implementation level. Thus we made one walk on the web without any self loop steps. After the walk for both algo-

Algorithm	Visited unique nodes	Seen hosts	Seen domains
Compaq	695,458	1,814,444	991,687
Berkeley and NEC	842,685	2,360,329	1,041,903

Table 2: The walks in the our experiment.

rithms we simulated self loop steps for each state. Since the walks are memoryless self loop steps do not have any effect on the progress of the walks.

Sampling Phase: The authors of [2] and [9] propose to uniformly sample the states of the walk after the first x states to reduce the starting state bias. The number of steps until the walk has reached a uniform distribution is called the *mixing time*. There are bounds on the mixing time that depend on the eigenvalue gap of the walk, which in turn depends on the structure of the web graph. Since the eigenvalue gap of the web is not known, for the Berkeley walk we decided to sample in two different ways: (1) *IgnoreFirst10k:* In each thread we determine a cutoff state by determining the first 10,000 states ignoring states reached by self loops. We remove all the states before the cutoff state and randomly sample states from the rest. (2) *FromLast10k:* For each thread we randomly sample from the last 10,000 states of each thread. When determining the last 10,000 states, the states reached by self loops are ignored. For each sample we remove duplicate nodes.

We sampled the NEC walk in three different ways. In each case web pages are sampled with probability inversely proportional to their (degree+1) in order to compensate the visiting bias. (1)*NEC IgnoreFirstQuarter:* We ignore the quarter of states of each thread including self loop steps. (2)*NEC IgnoreFirstHalf:* We ignore the half of states of each thread including self loop steps. (3)*NEC FromLastQuarter:* We only consider the last quarter of the states of each thread. In each sample we allow duplicate nodes, following [9].

3. RESULTS

We ran both walks for 240 hours on two identical machines equipped with a Intel Pentium 4 processor 3.0 Hz (Hyper-Threading enabled) and 4 GB of RAM. As database we used PostgreSQL 7.4.8. The implementations shared as much code as possible. All walks started from `www.yahoo.com` and used 50 threads. Three of the Berkeley and the NEC threads had to be stopped because of host overload before the end of the crawl. We removed their nodes and edges from the Berkeley and the NEC walk. None of the Compaq threads had to be stopped.

3.1 Comparison of the Algorithms

In order to evaluate the Compaq, the Berkeley and the NEC algorithms we compared the whole random walks done on the web and the samples taken at the end of these walks.

3.1.1 PageRank Biases

The Compaq walk tries to visit pages roughly according to their PageRank values. The most frequently visited pages should have high PageRank values.

Table 1 presents the top visited 10 URLs during the Compaq walk. We also give the PageRank as returned by the Google toolbar next to each URL. For one URL no PageRank is returned, all others have Toolbar PageRank 7 or above. We conclude that our walk succeeded to visit high PageRank pages more frequently than other pages. We observed no obvious bias towards high PageRank pages in the Berkeley and in the NEC walk and indeed the top visited URLs have Toolbar PageRank 0.

We also analyzed how the top visited URLs changed over the course of the walk. The rank of the start node decreased gradually as the walk proceeded. In the final 340k visited nodes it was not in the top 10 most frequently nodes anymore. This gives an indication that ignoring the first x nodes (for some appropriate choice of x) when sampling the Compaq walk might be a good idea to reduce the starting state bias.

In the Compaq walk visited hosts should have a bias to high PageRank hosts. In Table 3 we give the top visited hosts of the Compaq walks. The visit count of a host is the sum of the visit counts of its pages. The most visited hosts of the Berkeley walk and the NEC walk show no obvious bias to well-known, popular hosts.

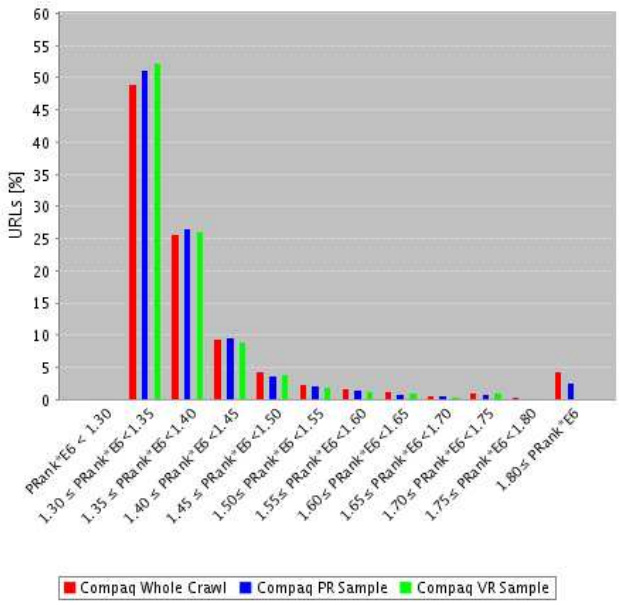

Figure 1: PageRank distribution in the subgraph crawled in Compaq Walk

Two of the Compaq samples used a PageRank substitute to reduce the PageRank bias. We call the PageRank of the subgraph crawled in the Compaq walk as the *subgraph PageRank*. Figure 1 shows the percentage of nodes in certain subgraph PageRank ranges for the whole crawled subgraph, for the Compaq PR Sample and for the Compaq VR Sample[2]. Since the Compaq PR Sample was created by sampling inversely proportional to the subgraph PageRank we would expect that nodes with low subgraph PageRank are more frequent in the sample than in the graph as a whole and very few nodes with high subgraph PageRank are in the sample. This is exactly what we see in Figure 1. Addition-

[2]The subgraph PageRank of a node can be very different from its PageRank in the whole web graph.

PR	Visit#	URL
8	929	http://extreme-dm.com/tracking/
10	810	http://www.google.com/
8	696	http://www.macromedia.com/shockwave/download/download.cgi
-	478	http://www.sitemeter.com/default.asp
10	364	http://www.statcounter.com/
7	336	http://www.mapquest.com/features/main.adp
10	312	http://www.microsoft.com/windows/ie/default.mspx
9	312	http://www.yahoo.com/
10	294	http://www.adobe.com/products/acrobat/readstep2.html
9	286	http://www.blogger.com/start

Table 1: Most visited 10 URLs of the Compaq walk in 2005.

VisitCount	Host
4,509	www.macromedia.com
3,262	www.amazon.com
2,848	www.google.com
2,246	www.microsoft.com
1,617	www.cyberpatrol.com
1,462	www.sedo.com
1,412	www.adobe.com
1,132	www.cafepress.com
1,069	www.blogger.com
929	extreme-dm.com

Table 3: Most visited 10 hosts by the Compaq walk in 2005

ally it shows that the VR Sample behaves very similar to the PR Sample. We do not include the Berkeley and the NEC samples in this figure since they are not sampled inversely proportional to a PageRank substitute.

3.1.2 VisitCount

Let the *VisitCount* of a node be the number of visits to the node including self loop steps. Note that there were no self loop steps in the Compaq walk in contrast to the Berkeley walk and to the NEC walk. The average VisitCount for the whole Compaq walk is 1.36. The Compaq VR Sample sampled the pages inversely proportional to the VisitCount, i.e., the average VisitCount in the sample should be close to 1. Indeed is 1.10. When averaged over 5 independent sets of samples their standard deviation is 0.004, showing that the Compaq VR Sample clearly reduced the average VisitCount.

We observed that each of the Compaq samples contain about 9,500 unique hosts while the Berkeley and the NEC samples each contain less than 1,000 unique hosts. As can be seen in Table 4 the Berkeley and the NEC samples contain about three times as many visited nodes on the host `fr.shopping.com` than from other hosts. Thus this is a significant bias towards nodes on that host. This is due to multiple threads "almost getting stuck" in that host.

This seems like a fundamental flaw in the Berkeley approach: Consider an undirected graph of n nodes, consisting of a complete graph of $n/2$ nodes and attached to one of these nodes is a chain of $n/2$ nodes. Berkeley random walk on this graph has a very good change to get stuck in the complete subgraph when run long enough. To avoid this problem the authors of [2] added self loops to make the graph regular. As a result the walk is equally likely to "get stuck"

Berkeley IgnoreFirst10k		
Number of pages	Percentage of pages	Host
2,405	24.4%	fr.shopping.com
791	8%	www.hostpooling.com
537	5%	www.friday.littledusty.org
Berkeley FromLast10k		
3,804	40.72%	fr.shopping.com
666	7%	classifieds.fr
346	4%	www.rechtschutzversicherung.de
NEC IgnoreFirstQuarter		
2,486	24.91%	fr.shopping.com
918	9.20%	www.hostpooling.com
754	7.56 %	www.friday.littledusty.org
NEC IgnoreFirstHalf		
2,713	27.11 %	fr.shopping.com
1,057	10.56 %	www.friday.littledusty.org
1,049	10.48 %	www.hostpooling.com
NEC FromLastQuarter		
3,054	30.52%	fr.shopping.com
1,011	10.11 %	www.hostpooling.com
788	7.87 %	www.friday.littledusty.org

Table 4: The hosts with the most pages in the Berkeley and in the NEC samples.

.

on the chain as in the complete subgraph. However, the fundamental problem of "getting stuck", i.e., staying within a small part of the graph, is not solved. The NEC approach of adding just one self loop does not solve the problem of getting stuck either. The Compaq walk avoids this problem by performing random jumps. Indeed, the host with the largest number of pages in any of the Compaq samples has 31 nodes in the sample. This host is `www.amazon.com`. Performing random jumps after a certain number of steps would probably also be a good idea for the Berkeley walk and the NEC walk.

3.1.3 HopCount

We computed the HopCount as follows: When a node is visited for the first time, its HopCount is set to be one larger than the HopCount of the previous visited node in the walk. If there was just one thread, the HopCount would express the length of the path used to visit that node with self loops removed. Note that this does not have to be the shortest path in the web graph and the HopCount is only a rough indication of the distance from the start node. Ta-

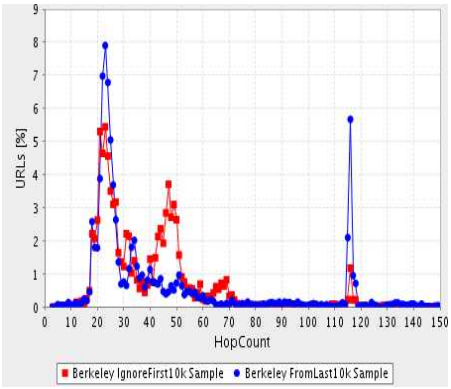

Figure 2: Berkeley HopCount distribution

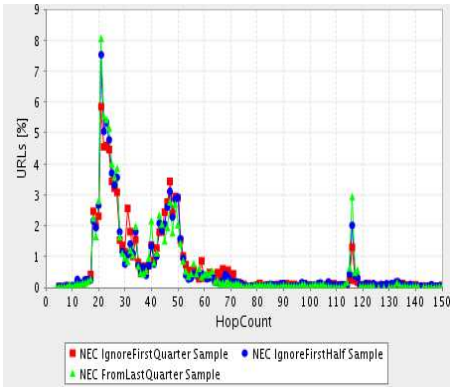

Figure 3: NEC HopCount distribution

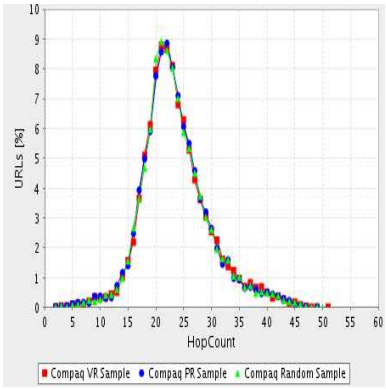

Figure 4: Compaq HopCount distribution

	HopCount		OutDegree	
Sample	Avg	Max	Avg	Max
Compaq VR	23.63	60	59.06	26,423
Compaq PR	23.51	61	60.22	62,021
Compaq Random	23.59	59	57.81	11,138
Berkeley IgnoreFirst10k	115.91	24,565	5.90	338
Berkeley FromLast10k	389.92	24,660	6.91	1,041
NEC IgnoreFirstQuarter	141.45	24,498	6.27	786
NEC IgnoreFirstHalf	167.80	24,630	6.24	1,057
NEC FromLastQuarter	185.22	24,681	6.03	520

Table 5: Statistics about the HopCount and the Outdegree distributions

ble 5 presents the average and maximum HopCount for the all samples. The sample type seems to have only a minor influence on the HopCount for the Compaq samples. We observed high maximum HopCounts for the Berkeley and the NEC samples when compared to the Compaq samples. The large average HopCount in the Berkeley and in the NEC walks has two reasons: (1) The Berkeley and the NEC walks do not perform random resets and thus they have a better chance of following long paths created by calendars and advertising pages. (2) We ignored the beginning states in the samples and at later stages of the walk nodes with higher HopCount tend to be visited. This can be clearly seen by the increase in the average when switching from the Berkeley IgnoreFirst10k sample to the Berkeley FromLast10k Sample and when switching from the NEC IgnoreFirstQuarter to the NEC FromLastQuarter sample.

The HopCount distribution for the Berkeley samples and the NEC samples is given in Figures 2 and 3. These figures contain spikes which are due to almost getting stuck in some hosts. In both figures for the all samples, there is a spike at 23 due to `fr.shopping.com` whose average HopCount is 23. 7% of the nodes in the BerkeleyFromLast10k sample are from `classifieds.fr`. Since 78% of the nodes of this host have HopCount 116, this results in a spike of size 5.5% at HopCount 116 in figure 2. For the NEC samples the spike at HopCount 116 is again due to skewed distribution to `classifieds.fr`. In Figure 3 the spikes at HopCount 47 and 49 are because of hosts `www.hostpooling.com` and `www.iconnectghana.org` which are among the hosts with the most pages in the NEC samples. In comparison the Compaq plot is quite smooth as can be seen in figure 4.

3.1.4 Outdegree Distribution

As was shown in the literature [1, 7, 4] the outdegree of the nodes in the web graph follows power laws. Seeing a power law outdegree distribution in a sample would be a strong indication that the sample is indeed random. Table 5 gives statistics about outdegrees. It shows that the Compaq walk is biased towards high outdegree nodes more than the Berkeley walk and the NEC walk. Note that the different sampling techniques for the Compaq walk do not compensate for this bias. In Figure 5 we plot the outdegree

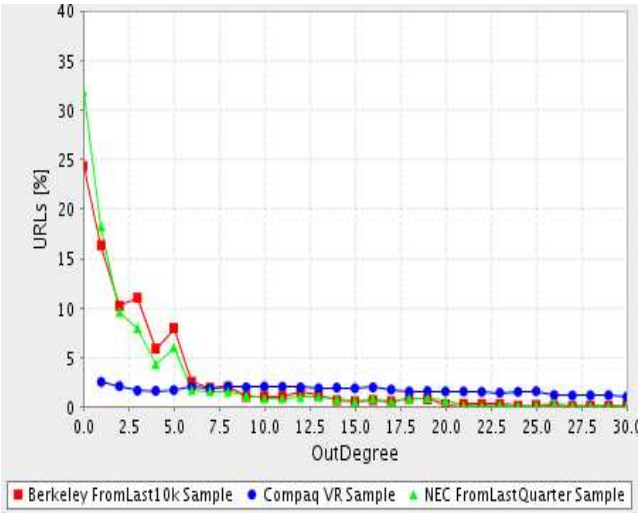

Figure 5: Outlink distribution

distribution for one of the samples of each algorithm as there is barely any difference between the different samples of an algorithm. The outdegree distribution for each the Berkeley and the NEC sample is somewhat similar to the expected web outdegree distribution where there are more nodes with low outdegree. The almost flat curve up to outdegree 20 and the gradual decrease afterwards indicate that none of the Compaq samples, sample the nodes reliably according to their outdegree.

3.1.5 Top-Level Domain Distribution

	2005								2000	
	Compaq Sample			Berkeley Sample		NEC Sample			Compaq	Berkeley
Domain	Random	PR	VR	Ignore First10k	From Last10k	Ignore FirstQuarter	Ignore FirstHalf	From LastQuarter	VR Sample	Sample
com	63.04	62.50	63.11	66.53	63.79	64.65	62.38	62.15	45.62	49.15
edu	0.71	0.48	0.75	0.12	0.07	0.03	0.04	0.06	9.84	8.28
org	9.85	10.13	9.77	10.95	2.94	13.25	15.33	13.05	9.12	6.55
net	6.14	6.32	6.35	8.24	3.58	7.78	7.69	7.27	4.74	5.60
jp	0.37	0.52	0.48	0.12	0.73	0.31	0.43	0.26	3.87	2.87
gov	0.57	0.45	0.40	0.04	0.06	0.07	0.01	0.05	3.42	2.08
uk	3.1	3.17	3.14	0.30	0.48	0.41	0.57	0.35	2.59	2.75
us	0.71	0.67	0.51	0.50	0.27	0.64	1.08	1.02	1.77	1.12
de	3.31	3.19	3.37	3.00	7.78	3.19	3.35	3.98	3.26	3.67
ca	0.94	0.87	0.82	0.10	0.06	0.04	0.04	0.07	2.05	1.58
fr	0.45	0.36	0.32	1.94	9.45	1.87	2.99	4.85	0.99	1.01

Table 6: The top level domain distribution in 2005 and 2000

In Table 6 we give the distribution of top level domains in summer 2005 as determined by the samples of Compaq, Berkeley and NEC algorithms. The Berkeley FromLast10k sample and all the NEC samples show large bias towards `.fr` resulting from the high frequency of `classifieds.fr` in the samples. Still there is a large agreement on the percentage of pages for top level domains which have high percentages. For comparison we show the distributions published in the two papers [2, 6]. Those samples are the results of the crawls run on the web in 2000 from the USA. We made our crawls in 2005 from Switzerland and with the changes explained in section 2. We believe that most of the differences are due to the rapid changes in the web over the last 5 years.

4. CONCLUSIONS

In this paper we compared the Berkeley, the Compaq and the NEC algorithm under conditions that are as equal as possible. The Berkeley algorithm and the NEC algorithm seems to more correctly reflect the outdegree distribution of the web than the Compaq algorithm. The Berkeley and the NEC walk more frequently revisits nodes (even when we ignore self loops). The Berkeley and the NEC algorithm has a serious problem with "getting stuck" in hosts. We tried to avoid this by stopping the walk when it could not leave a host for a large number of steps. However, a better approach might be to perform a random reset every x steps, like the Compaq algorithm. An appropriate value for x needs to be determined experimentally. The improved NEC algorithm and the improved Berkeley algorithm can give better results than their old versions for the top level domain distribution since their samples won't be biased to specific hosts. It is quite possible that the improved NEC algorithm and the improved Berkeley algorithm will reflect the outdegree distribution of the web more correctly than the Compaq algorithm.

5. ACKNOWLEDGEMENTS

We would like to thank Varun Kanade for the implementation of PageRank Algorithm, Christopher Iu Ming-Yee for useful discussions, and Fabien Salvi for his technical assistance during our research project.

6. ADDITIONAL AUTHORS

Additional authors: Stefan F. Keller (University of Applied Science Rapperswill, email: `sfkeller@hsr.ch`) and Markus Kinzler (University of Applied Science Rapperswill, email: `mkinzler@hsr.ch`).

7. REFERENCES

[1] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(509), 1999.

[2] Z. Bar-Yossef, A. Berg, S. Chien, J. Fakcharoenphol, and D. Weitz . Approximating aggregate queries about web pages via random walks. *Proceedings of 26th International Conference on Very Large Databases*, 2000.

[3] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Proceedings of the 7th International World Wide Web Conference*, pages 107–117, April 1998.

[4] A. Broder, R. Kumar. F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, J. Wiener. Graph structure in the Web. *Proceedings of the 9th International World Wide Web Conference*, pages 309–320, June 2000.

[5] M.R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. Measuring index quality using random walks on the web. *Proceedings of the 8th International World Wide Web Conference*, pages 213–225, May 1999.

[6] M.R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. On Near-Uniform URL Sampling. *Proceedings of the 9th International World Wide Web Conference*, pages 295–308, May 2000.

[7] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for emerging cyber communities. *Proceedings of the 8th World Wide Web Conference*, 1481–1493, April 1999.

[8] S. Lawrence and C. L. Giles. Accessibility of Information on the Web. *Nature, 400:107-109, 1999*

[9] P. Rusmevichientong, D. M. Pennock, S. Lawrence, and C. L. Giles. Methods for sampling pages uniformly from the world wide web. *Proc. of AAAI Fall Symposium on Using Uncertainty Within Computation*, pages 121–128, 2001.