On Improving Local Website Search Using Web Server Traffic Logs: A Preliminary Report

Qing Cui
Department of Computer Science
University of Kentucky
Lexington, KY, USA
Public Relations & Marketing
Eastern Kentucky University
Richmond, KY, USA
qing.cui@eku.edu

Alex Dekhtyar[*]
Department of Computer Science
University of Kentucky
Lexington, KY, USA
dekhtyar@cs.uky.edu

ABSTRACT

In this paper we give a preliminary report on our study of the use of web server traffic logs to improve local search. Web server traffic logs are, typically, private to individual websites and as such – are unavailable to traditional web search engines conducting searches across multiple web sites. However, they can be used to augment search performed by a local search engine, restricted to a single site.

Web server traffic logs, which we will refer to as simply logs throughout this paper, contain information on traffic patterns on a web site. By using this information, instead of pure link counts in the computation of PageRank, we can obtain a new local measure of web site importance, based on frequency of visits to a page, rather than simply on the amount of links.

In this paper we describe the architecture of a search engine we have built for the Eastern Kentucky University (EKU) website and some preliminary experiments we have conducted with it.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.5 [**Information Storage and Retrieval**]: Online Information Services

General Terms

Algorithms,Experimentation.

Keywords

PageRank, Markov Decision Process, probabilistic automata, web search, local web search, web traffic logs.

1. INTRODUCTION

Local web search, defined for the purpose of this paper as search among the web pages of a *single web site* has emerged as a distinct application in recent years and has gained considerable support in industry: from Google's facility allowing to restrict search results to a specific domain, to special-purpose local search engines running directly on the web sites. Local search engines on commercial web sites find products, documentation and other important information for the company's employees, customers and potential customers.

In this paper we describe one approach to improving local search by a special-purpose search engine deployed at the web site. The approach uses the information from the web site's logs to augment the traditional search mechanisms and determine the importance of individual web pages with respect to user queries.

Why use logs to augment local search? Consider the PageRank model described in [6, 1] on which early Google was based. PageRank is used to ensure that the top pages shown to the user for the queries are ”important.” In a nutshell, PageRank's notion of importance can be expressed as follows: ”if many important pages link to a page, then this page is also important.” The computation of PageRank is achieved by a relatively simple iterative procedure which distributes the weights among the URLs in the WWW graph. This computation assigns to each page a PageRank based on the current structure of the (observable) World Wide Web. In the absence of information about human traffic patterns on the web, the PageRank computation assumes that on each page, the user is as likely to follow a specific link, as any other link. As the success of Google shows, on the scale of the entire observable World Wide Web, this assumption leads to excellent results.

However, generally speaking, web traffic is not uniform. However, the exact information about web traffic is distributed across all web sites on the World Wide Web in a form of logs. This information is typically held under very close control by the web site owners and administrators, and is *not provided* to global web search engines. Things are different, when local search is considered. Here, a special-purpose local search engine can be installed directly on the web site under the command and control of the web site owners. Thus, there is no privacy/data ownership barrier between the local search engine and the log information. At the same time, anecdotal observations suggest that the information about the web pages that people *actually visit* as opposed to the web pages which have a lot of links pointing at them can be very useful in the search process.

[*]Work supported, in part, by the NSF grant ITR-0325063.

WIDM'05, November 5, 2005, Bremen, Germany.

In Fall 2005 semester, one of the authors was teaching CS 505 - a graduate level database class at the department of Computer Science at the University of Kentucky. In the middle of the semester, a query ”CS 505” posed to Google and restricted to the ”uky.edu” domain, returned as the first hit the CS 505 web page for the Spring 2005 semester. The following few links were to even earlier pages for the course and to the old materials from these pages. Current web page for CS 505 appeared only at the bottom of the Top 10. Both Fall 2005 and Spring 2005 web pages for CS 505 has similar material and provide the same strong match for the query ”CS 505.” However, the Spring 2005 page turned out to have a higher PageRank, because, being older it was better integrated with other web pages on the uky.edu site. At the same time, in the middle of the Fall 2005 semester, current CS 505 web page experienced significantly higher traffic than the older pages as reflected in the logs. Had the traffic information been available to the (appropriately tailored) PageRank computation, the Fall 2005 web page would have obtained the highest PageRank out of all pages matching the query, and would have been returned at the top, as one would want it to.

Observations similar to the above, suggest to us that logs can be used to improve the results of the search by making them structured according to the recent traffic patterns on the site, rather than according to the current static structure of the site. In this paper we describe the preliminary results we have obtained. We have proposed a procedure that computes a PageRank for a web page according to the number of visits to the page and the traffic patterns on the site. We call the new measure LPageRank. We have used LPageRank in conjunction with a simple tf-idf-based retrieval procedure to build a local search engine. The search engine has been deployed at the Eastern Kentucky University (EKU) web site[1]. In this paper we report on the initial experiments that compare the search results of our LPageRank-powered method with two baselines: Google domain search and standard localized (to the eku.edu domain) PageRank implemented by us.

The rest of the paper is organized as follows. In Section 2 we describe LPageRank. In Section 3 we describe the architecture of the search engine we have built. In Section 4 we describe our initial experiments and provide the results.

2. LPAGERANK: USING LOGS IN LOCAL SEARCH

2.1 PageRank in a Nutshell

In [6] Page, Brin, Motwani and Winograd introduced PageRank, the mechanism for determining the overall importance of a web page. Intuitively, PageRank of a web page is an approximation of a probability to reach this page from some other page on the web[2]. PageRank models the page traversal as follows: at each step, a user is observing a web page with m outgoing links to distinct pages. At this point, the user either makes, with probability α a decision to type in a new URL, and goes to a random page in the collection, or, with remaining probability $1-\alpha$ follows one of the links on the page. PageRank assumes that the user is not biased in his/her choice of the link, thus, the probability to follow a specific link is $\frac{1-\alpha}{m}$. When considered globally, the leads to the following recursive formula for computing a PageRank (PR) of a page:

$$PR(A) = \alpha + (1-\alpha)\left(\sum_{B \in Parents(A)} \frac{PR(B)}{N(B)}\right),$$

where, $Parents(A)$ is the set of all web pages which link to A and $N(P)$ is the number of outgoing links to distinct pages found on page P. Typically, PageRank is computed iteratively, starting with a state in which it is uniformly distributed among all web pages, and continuing until a stabilization condition holds. Also, α, the probability is typically set to a small number between 0.05 and 0.15.

2.2 Probabilistic Graph of a Web Page Collection

PageRank implicitly models user behavior in terms of a *probabilistic graph* of the web. Indeed, Each page in the collection can be thought of as a node in a graph. A link from page B to page A can be modeled as a directed edge (B, A). Finally, PageRank computation assumes that, given a page B, the probability to follow some outgoing edge (A, B) is $\frac{1-alpha}{N(A)}$. The triple $G = \langle W, E, P\rangle$ where N is a set of nodes, $E \subset W \times W$ is a set of directed edges and $P : E \to [0, 1]$, s.t., $(\forall B \in N) \sum_{(B,A)\in E} P((B, A)) \leq 1$ is called a *probabilistic graph*. Probabilistic graph is one of many ways to visualize Markov Decision Processes (here, nodes play the role of states, and edges are state transitions with appropriate probabilities associated with them).

The probabilistic graph constructed (implicitly) by the PageRank computation assumes uniform probability distribution for outgoing edges for each node.

2.3 LPageRank

In a nutshell, $LPageRank$ is a PageRank computation based on a probabilistic graph of a web page collection that reflects traffic patterns obtained from the logs. We discuss how to construct a probabilistic graph of a web site given a log in Section 3. Here, suppose $G = \langle W, E, P\rangle$ is a probabilistic graph over a collection of web pages W. Then, LPageRank (LPR) of a web page is computed as follows:

$$LPR(A) = \alpha + (1-\alpha) \sum_{B \in Parents(A)} LPR(B)P\left((B, A)\right).$$

Note that $LPR(A) = PR(A)$ for graphs G, in which $P(B, A) = \frac{1-\alpha}{N(B)}$ for all edges (B, A), .

3. ARCHITECTURE OF A LPAGERANK-BASED LOCAL SEARCH ENGINE

In this section we give a brief description of the local search engine based on LPageRank, that we have built. While the search engine code is web site-independent, the search engine is currently available for use to search the web pages of Eastern Kentucky University (EKU). The search engine can be accessed at
http://w4.eku.edu/websearch.

Figure 1 shows the architecture of the search engine. It consists of three major components. The *site crawler* (Figure 1, top right corner) crawls the web site and builds the connectivity graph for it. In addition, it retrieves individual web pages, parses them, extracts keywords and builds vectors of keywords weights. The *log miner* (Figure 1, bottom right corner) collects information from the logs, and uses it to transform the connectivity graph of a website into the probabilistic graph, by assigning a probability to every edge in the graph. From this information, LPageRank is computed for

[1] One of the authors of the paper works as the web site administrator at EKU. The search engine is available as an alternative to the standard EKU search engine at http://w4.eku.edu/websearch.

[2] In [6], PageRank is computed with respect to different starting points: a specific web page, a web page from a preset collection of pages, a random web page, etc.

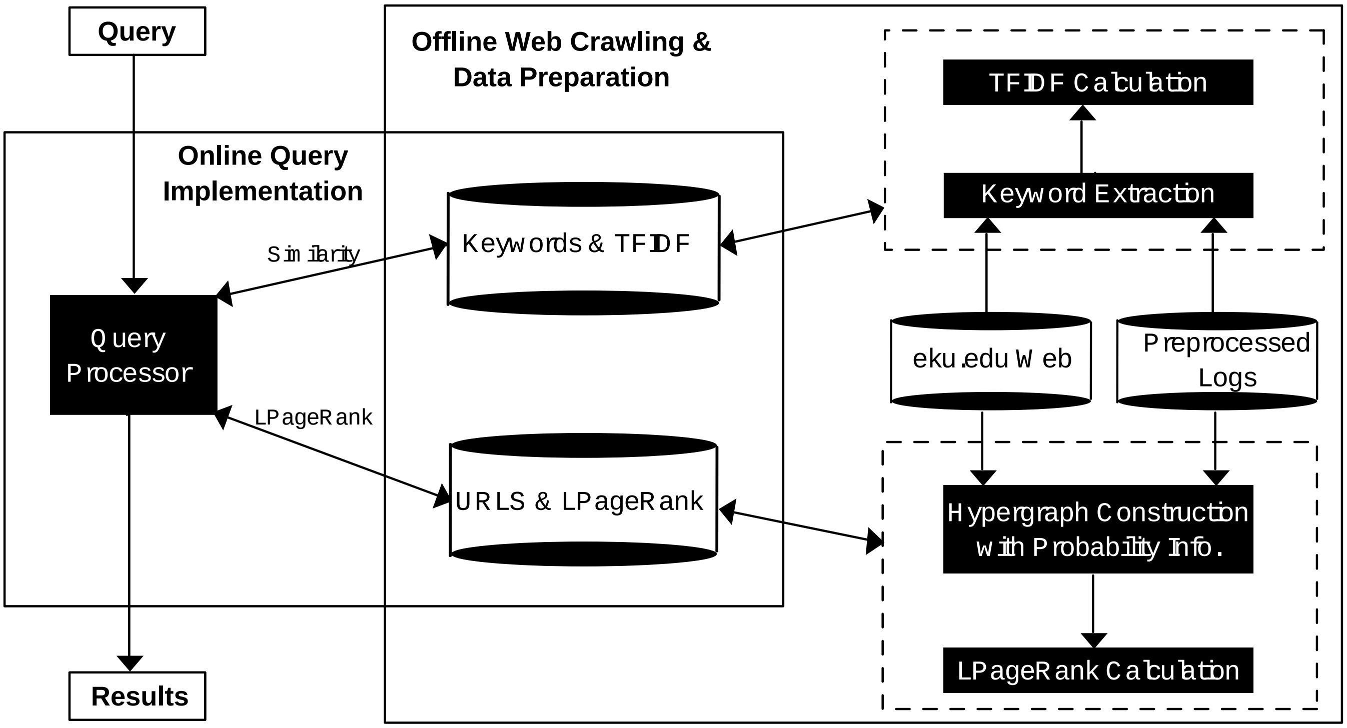

Figure 1: Project Components.

all indexed website pages. Finally, the *front end* of the search engine (Figure 1, left side) accepts user query and combines keyword-based retrieval with LPageRank to produce a ranked list of URLs matching the query. We briefly outline each component below.

3.1 Site Crawler

The *site crawler* component of the system performs two functions: (a) it constructs the connectivity graph for a given website and (b) is obtains the HTML for each page and indexes it.

The crawler takes as input a starting web page. We assume that the starting web page is a front page of a web site (such as www.eku.edu or www.acm.org). It crawls the website, by retrieving and indexing all HTML documents whose URL is from the same site as the starting URL. Thus, the crawler ignores all outside links both in traversal and building the connectivity graph.

Each HTML document obtained during the crawling operation is then piped through a standard sequence of tools to build a vector of keyword weights: parser – lexical analyzer – stopword remover – stemmer – vector constructor. During this process, we build the vocabulary $V = \{k_1, \ldots, k_M\}$ of keywords (terms) found on the website. We use standard *tf-idf* term weighting schema: $w_i = tf_i(d)log(N/df(i))$, where $tf_i(d)$ is a (normalized) frequency of a keyword k_i on page d, while $df(i)$ is the document frequency (i.e., the number of documents on the website in which it appears) of k_i.

3.2 Log Miner

The *log miner* component of the system also performs two tasks: (a) completion of construction of the probabilistic graph of the web site and (b) computation of LPageRank. First, it takes This input to the log miner is a log or a collection of logs from the website of interest. The log consists of a sequence of HTTP requests. While different web servers put slightly different information in the logs, we can assume that the log entry has the following format: (TimeStamp, IPaddress, RequestedURL, BytesSent, RequestStatus) (we ignore all other information found in the log).

The miner performs the following operations with the logs:

1. Log cleanup. The input logs are cleaned of all irrelevant information, such as failed page requests and inlined image requests. Also a few straightforward steps [2, 8] are implemented to identify and filter out the robot references through the logs:

   (a) Any web requests that reference robots.txt are purged away;

   (b) We got a list of known web robots such as Yahoo! Slurp, Google Robots, Baidu Spiders, Microsoft Spider, etc. When a log entry matches one of the known spiders, it is purged from the list.

   (c) Finally, in the sessionization process, we identify the robots activity by looking at the number of requests in a short period of time. If the number of requests is more than the predefined threshold value, we conclude that user is not a regular user, it is either a robot, web spider or programmed web crawler.

2. Session detection. A session is a sequence of URLs requested by the same user within reasonable time from each other. We use standard session detection conventions[5, 7, 4]: the user is assumed to be uniquely defined by an IP address recorded in each http request within the time-frame of a single session. We use a threshold of 30 mins between two consecutive web page requests to determine the end of a session.

3. Session analysis and construction of the probabilistic graph of the website. The probabilistic graph is constructed as follows. We start with the connectivity graph build by the site crawler. Each session extracted from the log is then mapped onto the graph. Each time an edge (B, A) is followed in a session (we assume that if B, A is part of the session sequence, then the user clicked on a link to A while observing B in the browser), we increase the weight of the edge (B, A) in the graph by 1. After all sessions are analyzed in this manner, we normalize the edge weights by the total sum of all weights on the outgoing edges of each node.

4. LPageRank computation. Given the probabilistic graph constructed in the previous step, the LPageRank computation is then performed. We take the value of $\alpha = 0.15$, and start with a uniformly distributed LPageRank LPR_0. We than compute each next iteration as follows:

$$LPR_{i+1}(A) = \alpha + (1 - \alpha) \sum_{B \in Parents(A)} LPR_i(B) P\left((B, A)\right).$$

This process stops when LPR_i and LPR_{i+1} are different by no more than some small number ϵ. At present implementation, ϵ is set to 0.000001.

3.3 Front End

Both the site crawler and the log miner components of the search engine work off-line, preparing the data for on-line use. *Front end* is the on-line component of the search engine, responsible for accepting the user query, parsing it and converting to the same vector representation as the web pages, and conducting the actual retrieval of the relevant web pages.

Given a page vector d and a query vector q, their similarity is computed as the cosine of the angle between them in the M-dimensional space:

$$sim(d, q) = cos(d, q) = \frac{d \cdot q}{|d||q|}.$$

The similarity between d and q is then weighted by the LPageRank of d to produce the final relevance score:

$$relevance(d, q) = LPR(d) sim(d, q).$$

All pages with non-zero scores are retrieved and returned in batches of 10 or 20 URLs.

4. EXPERIMENTAL RESULTS

We have conducted some preliminary experiments to test the LPageRank-augmented retrieval. In this section we describe the obtained results.

4.1 Experimental Setup

The search engine described in Section 3 had been implemented and deployed on the web site of Eastern Kentucky University. It can be accessed and tested at http://w4.eku.edu/websearch. To test its performance, we have considered two alternative local search methods: Google's domain-restricted search, with domain restriction eku.edu and local PageRank. For the former baseline, we simply pass the query to Google's domain-restricted search and retrieve the links that Google returns. For the latter baseline method, we use the local search engine we have implemented, and substitute computation of LPageRank (and consequently, the entire log mining component) with the straightforward computation of standard PageRank as described in [6]. PageRank is computed only for the pages on the local website (eku.edu, in our case), all outside links found on the web pages are ignored.

ID	Query
1	2005 spring courses
2	campus dining
3	campus map
4	cares report
5	Chemistry department
6	corbin campus
7	dining services
8	disability services
9	employment
10	financial aid
11	fitness and wellness center
12	GBU 201
13	homecoming court
14	lambda chi alpha
15	library hours
16	loan
17	McNair program
18	Qing Cui
19	ROY KIDD STADIUM
20	shuttle bus schedule
21	student id card
22	student jobs openings
23	teaching certificate
24	tuition fee
25	undergraduate admissions
26	undergraduate catalog

Table 1: Queries used in the experiment.

Our search engine indexes around 42,000 web pages in the eku.edu domain. We index only HTML and text files. Both LPageRank- and PageRank-based retrieval work over the same collection of pages. For this experiment, we took the collection of logs from the eku.edu web servers for the month of April 2005: we wanted to use regular semester traffic - April is the last full month of the Spring 2005 semester; final exams, break between Spring and Summer semesters, graduation and many other events occurring in May change traffic patterns on the www.eku.edu web site significantly.

We do not know exactly the number of web pages in the eku.edu domain indexed by Google, however, Google reports around 114,000 documents matching the domain-restricted query

```
"eku site:eku.edu".
```

However, this number includes files in formats other text and HTML. To correct for this difference in indexes, we remove all non-text and non-HTML URLs from Google's results.

We have selected a test dataset consisting of 26 simple keyword queries shown in Table 1. 24 of these queries have been selected among the queries posed to the *main* EKU site search engine by users[3]. We have assembled a list of queries asked during a single 24-hour period, and selected from it queries related to the academic life. During the experiments, we added two more queries, "Chemistry department" and "Qing Cui" to test retrieval of information about academic departments and individuals, to complete the dataset. Prior to its selection no query other than "chemistry department" had been tested on our search engine.

Each query was tested as follows. When entered in the query engine's main page, the query was sent to all three search engines used in the test. We have selected first twenty (20) answers returned by each engine. The lists of answers were united, sorted in lexicographical order, and duplicates were removed. The resulting list

[3] This is the engine that is accessbile from http://www.eku.edu page directly.

was shown to one of the authors of the paper, who had not seen results from individual query engines. The author inspected, in turn, each presented link and provided binary ”relevant”/”not relevant” feedback. A web page was typically deemed relevant to the query, if the tester could immediately get his information need satisfied (at least partially) by the content of the page, or if the page contained an easy-to-find link to the page satisfying the information need. Other pages were deemed not relevant. Any broken link was scored as not relevant.

Because of rather large size of the website, we could not determine the ”ground truth”, i.e., the set of all relevant pages for each query. Therefore, we could not and did not test for recall. Our two main measures of the method performance are precision and average expected precision among the Top 20 answers. In addition we have collected information about query coverage - how many URLs in the Top 20 lists returned by different methods matched. The results of this experiment are described below.

4.2 Results

Figures 2 and 3 show the precision and average expected precision recorded for all methods and queries. The mapping between the query numbers and queries is the same as in Table 1. In Tables 2 and 3 we provide summaries of pairwise comparisons between LPageRank-based retrieval and the baseline methods. In these tables, the ”Wins” column contains the number of queries from the test dataset for which LPageRank exhibited a better value of the measurement than the baseline method, ”Ties” counts the number of queries with the same value of the measurement, and ”Losses” is the number of queries on which the baseline method had a better measurement value. From the two graphs and two tables we can see the following:

- Google’s domain-restricted search shows overall the best performance on the test dataset in both precision and average expected precision. LPageRank outperforms Google on either of the measures only on about one out of every three queries.
- LPageRank outperforms PageRank in both precision and average expected precision.

In Tables 4 and 5 we show the amount of intersection between Google’s Top 20 and that of LPageRank and between PageRank and LPageRank respectively. We show the total number of common URLs in the Top 20s, the number of common relevant URLs, and, for reference, show the number of relevant URLs for each method (all numbers except for query 20 are out of 20 retrieved URLs. For query 20 Google returned 8 URLs and LPageRank returned 11). As we can see from this table, out of possible 508 matches (20*25+8), only 118 URLs, or about 23% were common to the two methods. Almost two thirds (78, or 66.1%) of the common URLs were those deemed relevant. Google found a total of 223 relevant links, while LPageRank retrieved 206, for a total of 351 (226+206 -78) relevant links observed. The percentage of common links among all observed is thus 22.2%. It is around 35% for the URLs returned by Google, and around 38% for the URLs returned by LPageRank. At the same time, PageeRank is much closer to LPageRank. For each query, more than 10 out of 20 URLs were common, and most of the relevant links were common.

4.3 Discussion

From the experiments we have conducted we can make two positive conclusions. First, with everything else being equal, LPageRank outperforms regular PageRank and provides more relevant links and better quality lists of links to the users.

LPageRank vs. Google	Wins	Ties	Losses
Precision	9	2	15
Average Expected Precision	8	0	18

Table 2: Summary of results: LPageRank vs. Google

LPageRank vs. PageRank	Wins	Ties	Losses
Precision	13	10	3
Average Expected Precision	15	2	9

Table 3: Summary of results: LPageRank vs. PageRank

Query	Common URLs	Common Relevant	Google Relevant	LPageRank Relevant
1	2	2	3	2
2	1	0	1	2
3	5	2	9	5
4	5	5	13	11
5	6	2	8	8
6	3	3	14	11
7	3	1	7	1
8	2	2	14	7
9	5	5	18	13
10	6	5	15	11
11	6	2	11	6
12	6	3	8	13
13	12	6	11	8
14	11	10	12	18
15	3	3	11	16
16	6	5	15	9
17	12	11	16	13
18	0	0	12	19
19	4	0	2	1
20	3	0	0	1
21	6	2	3	7
22	1	1	3	4
23	0	0	4	2
24	4	4	6	6
25	3	2	4	4
26	3	2	3	8

Table 4: Common URLs between LPageRank and Google.

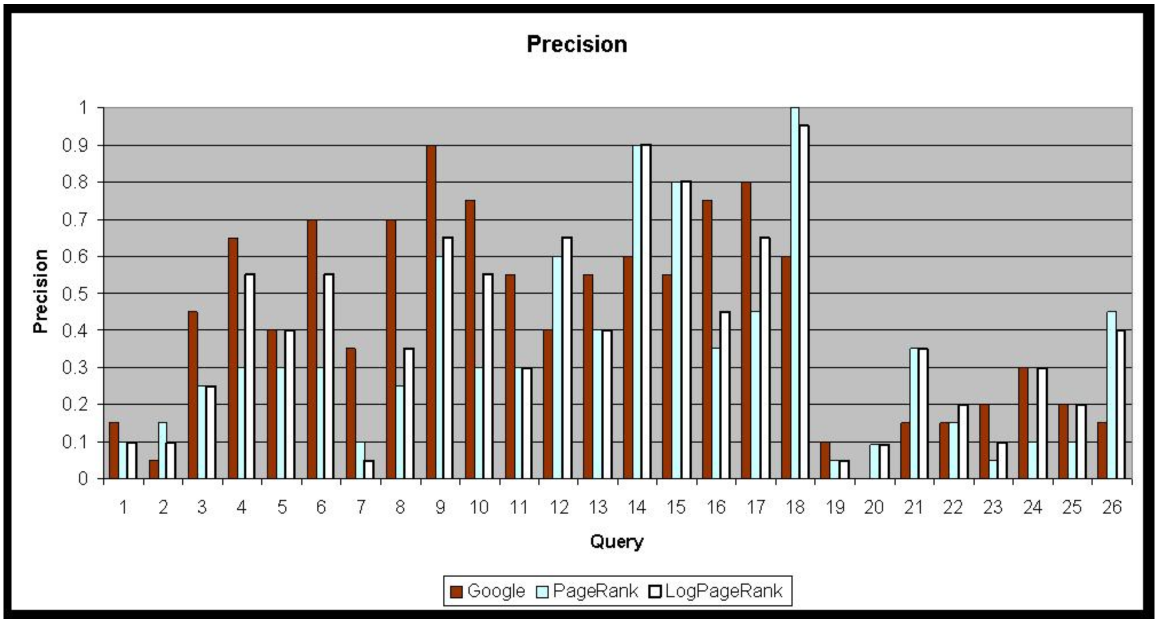

Figure 2: Precision results.

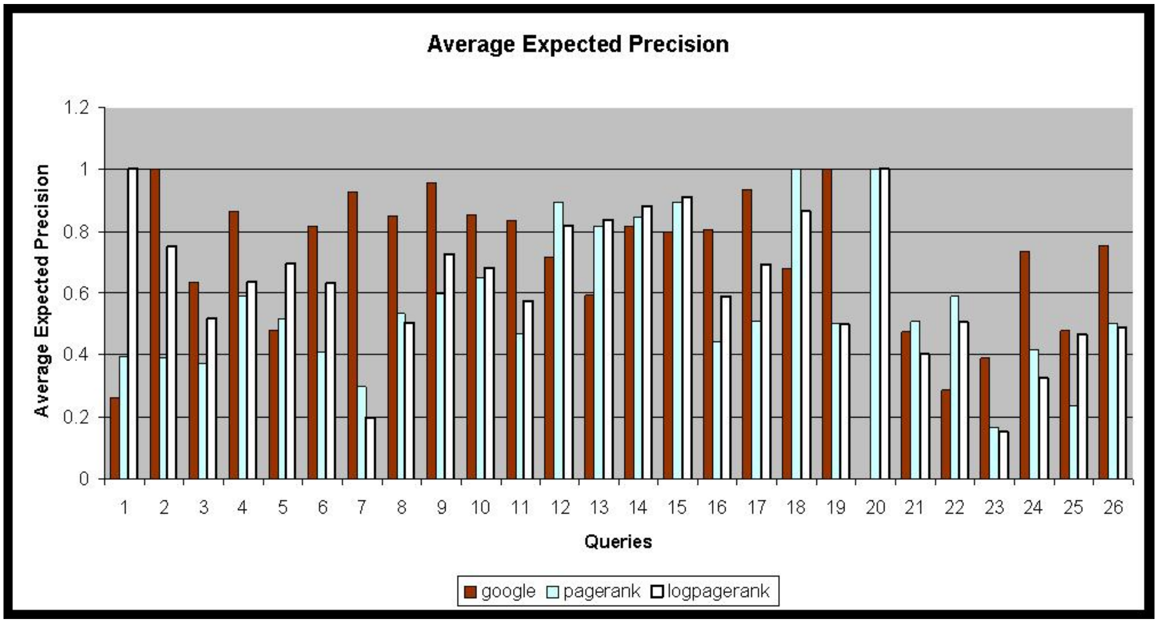

Figure 3: Average Expected Precision results.

Query	Common URLs	Common Relevant	PageRank Relevant	LPageRank Relevant
1	10	2	2	2
2	11	2	3	2
3	16	5	5	5
4	12	6	6	11
5	16	6	6	8
6	12	6	6	11
7	10	1	2	1
8	13	5	5	7
9	17	11	12	13
10	14	6	6	11
11	17	6	6	6
12	19	12	12	13
13	18	8	8	8
14	19	17	18	18
15	16	14	16	16
16	11	5	7	9
17	15	9	9	13
18	17	17	20	19
19	19	1	1	1
20	11	1	1	1
21	17	6	7	7
22	14	3	3	4
23	17	1	1	2
24	13	2	2	6
25	15	2	2	4
26	15	4	9	8

Table 5: Common URLs between LPageRank and PageRank.

Second, retrieval by LPageRank appears to be significantly different than retrieval by domain-restricted Google. Only 22% of all relevant links observed were common to both methods. This number would go up somewhat, but not significantly, if we considered as the same different URLs pointing to the same physical file. The majority of relevant links was still retrieved by only one of the two methods. Even more drastically differ the lists of irrelevant links returned by the methods - with less than 10% of common URLs. Without knowing the underlying proprietary mechanisms employed by Google, it is hard to assess the exact reasons, however, some ideas can be seen from observing the behavior of LPageRank. For many of the queries, LPageRank retrieved some pages with very low similarity scores, but extremely high LPageRanks - i.e., heavily traffic-ed pages. Google, we assume, has no information of which pages get heavy traffic, and the standard PageRank of these pages might not be as high.

On the other hand, we must note, that LPageRank combined with tf-idf is not enough to better Google's domain-restricted search. We can see two potential reasons for Google's better performance. First, while the exact matching formulas for Google are proprietary secrets, ever since [1] we know that Google combines PageRank with more than just tf-idf-based retrieval. Google's engine analyzes position of keywords on in the HTML, text of links pointing to the page, and, possibly many other factors, ignored by tf-idf. Second, we have observed that Google has indexed potentially a much larger number of pages in the ”eku.edu” domain. While some of the documents indexed by Google are not HTML or text files, and while some other documents are no longer present on the eku.edu web server, it is reasonable to assume that Google's list of eku.edu pages is larger. In part, this can be explained by our crawling mechanism - we will never find any web pages NOT reachable from the top page of the website, www.eku.edu. At the same time Google's crawler is global - and can detect these pages if they are linked from outside sources.

This paper outlines only the very preliminary results of our study of LPageRank. These results are encouraging and show that the use of LPageRank with appropriate retrieval methods may improve retrieval results. They also suggest two avenues for further improvement. First, we must subject our search engine to more rigorous testing, involving independent users rather than authors. Second, we must work to improve the IR component of the search engine, make it resemble more the actual methods used in existing web search engines. We also note, that LPageRank can be used as the means of analysis of website traffic and changes in it over time.

5. RELATED WORK

Probabilistic automata (a.k.a. Markov Decision Processes) for representing web traffic have been considered since the end of the 90ies [5, 2, 7, 3, 4]. This work, however, did not use the obtained web models to facilitate web search.

Page, Brin, Motwani and Winograd [6] have proposed PageRank as the means of determining the ”importance” of web pages. PageRank went on to become part of Google search engine [1].

Xue et. al [9] propose an approach to constructing implicit links by mining user's access patterns, and then apply a modified PageRank algorithm to re-rank web-pages for small web searches based on the extracted association rules.

Zhu et. al [10, 11, 12] propose PageRate, a method similar to ours, for using logs and MDPs to determine page importance. Their work on the theory of PageRate has influenced our approach. At the same time, to our knowledge, LPageRank is the first log based method for local search implemented into a fully operational search engine.

6. CONCLUSIONS

In this paper we have studied the use of logs to augment single website search engines – a type of search engine almost ubiquitously present on all major commercial, academic and interest-based websites. Our preliminary results indicate, that our proposed method, LPageRank outperforms standard PageRank when coupled with a simple retrieval method (tf-idf). We have also discovered, that despite showing worse performance than domain-restricted Google search, our LPageRank-based search engine retrieves significantly different results, and thus shows complimentary to Google behavior. Plans for further study of LPageRank have been briefly outlined as well.

7. ACKNOWLEDGEMENT

We would like to thank the following people: Dustin Tennill, System Administrator in Eastern Kentucky University for the support for Database and Operating System set up and Ron Yoder, Director of Web Communications in Eastern Kentucky University for various support and encouragement throughout the project.

8. REFERENCES

[1] Brin, Sergey and Page, Lawrence (1998). The anatomy of a largescale hypertextual Web search engine. In *Proc. of WWW7*, pages 107-117, Brisbane, Australia.

[2] R. Cooley, B. Mobasher, and J. Srivastava (1999): Data Preparation for Mining World Wide Web Browsing Patterns, J. Knowledge and Information Systems, vol. 1, no. 1, 1999, pp. 5-32.

[3] X. Fu, J.Budzik, and K. J. Hammond. (2000): Mining Navigation History for Recommendation. In Proceedings of the 2000 International conference on Intelligent User Interfaces, New Orleans, LA, January 2000. ACM Press.

[4] Bamshad Mobasher, Honghua Dai, Tao Luo, Miki Nakagawa. (2002): Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization. Data Mining and Knowledge Discovery, 6,61-82,2002.

[5] Bamshad Mobasher, Namit Jain (1997): Web Mining: Pattern Discovery from World Wide Web Transactions, March 1997.

[6] Page, Lawrence, Brin, Sergey, Motwani, Rajiv and Winograd, Terry. (1998) The PageRank Citation Ranking: Bringing Order to the Web, University of Stanford Technical Report 1999-66, http://dbpubs.stanford.edu/pub/1999-66.

[7] Jaideep Srivastava, Robert Cooley (2000): Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data, ACM SIGKDD, Jan 2000.

[8] Pang-Ning Tan, Vipin Kumar (2000): Modeling of Web Robot Navigational Patterns, WebKDD 2000: Web Mining for E-Commerce, Boston,. MA, August 20, 2000.

[9] Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Wei-Ying Ma, Hong-Jiang Zhang, Chao-Jun Lu (2003): Implicit Link Analysis to Small Web Search, in Proceeding of 26th Annual International ACM SIGIR Conference, Toronto, July 2003.

[10] Zhu, Jianhan, Hong, Jun, Hughes, John G. (2001): PageRate: counting Web users' votes. In *Hypertext'01 - Proceedings of the Twelfth ACM Conference on Hypertext and Hypermedia*. August 14-18, 2001, Aarhus, Denmark. p.131-132.

[11] Zhu, Jianhan, Hong, Jun, Hughes, John F. (2002): Using Markov models for web site link prediction. In *Hypertext'02 - Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia*. June 11-15, 2002, College Park, Maryland, USA. p.169-170.

[12] Zhu, Jianhan (2003): Mining Web Site Link Structure for Adaptive Web Site Navigation and Search. *Ph.D thesis*, University of Ulster at Jordanstown, UK.