

Discovering Paths Traversed by Visitors in Web Server Access Logs

Alper Tugay Mizrak

Department of Computer Engineering
Bilkent University
06533 Ankara, TURKEY
E-mail: mizrak@cs.bilkent.edu.tr

Abstract

One of the most difficult and complicated problems for the Web Masters is obtaining feedback for their web sites. Web server access logs are originally intended for being used by Web Masters. In fact these files are the ideal source of feedback.

On the other hand these files may reach to extreme sizes in a very short time periods and it becomes impossible to draw results without using advanced tools. As a result web server access logs are nice raw material for Data Mining applications but advanced techniques are necessary.

In this paper I handle the problem of discovering paths traversed by visitors of a web site. I have been developing the path discovery tool, PathFinder2K, mining web server access logs for paths traversed by visitors. I present the design of the PathFinder2K, examine the performance of the algorithm, and give some outputs of the tool.

1. Introduction

What are web server access logs?

Web server access logs are the files containing one entry for each request answered by web servers. They are originally intended for use by Web Masters to obtain feedback for their web sites. However they may grow by hundreds of megabytes per day and may reach to a size of

several gigabytes in a month for a popular web site. When you take this fact into account, it turns out that there is no one in the world such that he/she can scrutinize these files with the naked eye and draw conclusions.

As a result web server access logs are nice raw material for Data Mining applications.

For web server access logs, the common log format is used as the standard format. Basically servers record the information into access, referrer, agent and error log files:

Access Logs:

The following information is recorded in the access logs:

- Domain name/IP address: Name of the computer accessing a file on the server, or IP address of the particular machine. *Pc511dis.cs.bilkent.edu.tr*
- Login name: If the visitor logs in to access the files than the login name of the visitor is also stored. - -
- Timestamp: The second that the requested file was transmitted. *[14/Jun/1999:11:01:07 +0300]*
- File name: The name of the requested file and the performed action (e.g. GET, POST, HEAD). *"GET /~guvenir/Oku HTTP/1.0"*
- Code: This is a three digit number indicating the result of the request. *200*
- Size of the file: The size of the transmitted file (in bytes). *4568*

Referrer Logs:

The last file accessed by the visitor is recorded in this file, indicating that the visitor reaches to the current file through a link in the previous file.

E.g.:

```
http://www.cs.bilkent.edu.tr/~guvenir -->
/~guvenir/Oku
```

Agent Logs:

The browser and the platform used by the visitor are recorded into this file.

E.g.:

Mozilla/4.0 (compatible; MSIE 4.01; Windows 98), for the visitor uses Microsoft Internet Explorer on a computer with the platform Windows 98.

Error Logs:

Detailed information of the access that causes error is recorded.

Combined Access Logs:

There is information loss while recording the information of each request into separate files: access log, referrer log, agent log. By adjusting the setting of the server all information can be recorded into one file called combined access log. By this way it is possible to cross-tabulate the information stored separately before, e.g. accessed file with platform or referring page with browser. Combined access log format is likely to become industry standard henceforth.

E.g.:

```
pc511dis.cs.bilkent.edu.tr - - [14/Jun/1999:11:01:07
+0300] "GET /~guvenir/Oku HTTP/1.0" 200 4568
"http://www.cs.bilkent.edu.tr/~guvenir" "Mozilla/4.0
(compatible; MSIE 4.01; Windows 98)"
```

Previous Work

In recent years many successful data mining applications on web server access logs have been developed. These applications give information about the system performance, web

traffic, trends contingent to time, users preferences, etc.

On this topic more than a hundred programs can be found on the internet at first sight. Some of them are AccessWatch, Analog, Emily, Getstats, NetIntellect, Surf Report, WebStat, WebTracker, WebTrends, WWWstat, W3Perl. Most of these programs are free and their codes are available. They give the categorized hierarchical information, graphical interpretation of statistics, demographics, geographical distribution of visitors beside the simple statistical information, such as a summary of hits, access and transferred bytes, a report of popular browser and platforms, a report of top requested files and referrers, hourly and daily statistics of server load and a report of top referring domains. However most of the results are far from the conceptual level of human beings and most of them uses template layout format while few of them can be customized.

One of these novel applications is the Adaptive Web Sites which improve themselves by learning from user access patterns and make popular pages more accessible, highlight interesting links, connect related pages, cluster similar documents together, etc.[4,5].

Motivation

On the conceptual level web server access logs entries are the footprints of the visitors. Examining these tracks may lead us to a valuable information: the most popular paths traversed by visitors.

Counts for the links give valuable information to Web Masters such as which links are popular, why the other links are followed less, whether they should be emphasized more. This information lets us to examine how the visitors interact with the web site, which links are followed to reach their target web page.

Moreover the next step of this research is guessing the link that is likely to be traversed, predicting the final target of the visitor and we can use this information in order to classify the visitors and discover the preferences of users.

One of the poor points of the recent applications is that they do not give information about the paths traversed by the visitors.

I handle the problem of discovering paths traversed by visitors of a web site. I have developed the path discovery tool, PathFinder2K, mining web server access logs for paths traversed by visitors.

Outline

The following sections are organized as follows: In the next section, I describe the design of the PathFinder2K which is the data mining tool on web server access logs. Some samples of output are given, the performance of the algorithm is examined, and the result of the experiments are presented in the Section 3. Finally in Section 4, the conclusion is mentioned and future work is discussed.

2. Design of PathFinder2K

A path is the sequence of linked pages visited by a single user in a single visit. PathFinder2K extracts the *possible* paths from the web server access logs. I use the word "*possible*" because of several reasons that make it impossible to detect all the paths exactly. I discuss why it is difficult to discover the exact paths in the Section 4 *discussion* part.

Extraneous information is also stored in the log files such as the requests for image files, sound files. The solution is very simple for this: Simply ignore these entries and do not take these records into account.

Another difficulty comes from the definition of visit: A sequence of accesses from a single client. We can indicate the several accesses done by a particular machine in succession. However the critical question is the following: How much must the time pass in order to separate two successive accesses from the same client as different visits? As most of the current programs use 20-30 minutes as threshold to determine the separate visits, I assume a delay of 30 minutes in successive accesses indicates two different visits.

PathFinder2K Algorithm

PathFinder2K is designed for only combined access logs. For the systems recording the information of each request into separate files, the algorithm may be modified such that the entries in access log and referrer log are match firstly and then the information should be used in path finding algorithm.

The domain name, the timestamp, the referring page, and the finally the name of the file accessed are necessary for the following algorithm. The domain name is used for differentiating the requests done from different domains, and the timestamp is for separating the request into visits even if they are from the same domain. Eventually the referring pages and the name of the accessed files are for determining the paths traversed by the visitors.

As the program needs many string parsing operations, I choose Perl programming language to implement PathFinder2K. Because Perl makes it very easy to do operations with regular expressions which give us extreme power to do pattern matching on text documents. Moreover dynamic hash table data structure of Perl is also a big advantage.

Algorithm:

In this part I explain the algorithm I used in implementing PathFinder2K:

```
for each record in the web server access log
  1) parse the attributes of the record
  2) if it is an image or sound file
    2.1) get the next entry
  else
    2'.1) concatenate the referrer and target
          to the entry corresponding domain
          in the hash table
    2'.2) for each entry in the hash table
          exceeding the threshold value
      2'.2.1) find the possible paths for
              the particular visit and
              delete it from the hash table
```

The web server access log file is scanned and for each record in the file the following procedures are applied:

1) parse the attributes of the record

The attributes of the domain name, the timestamp, the referrer page and the file accessed are necessary so this information is extracted from the records.

More over small adjustments are done such as: %7E and %7e are changed as ~ symbol, the extra slashes are removed, and redundant :80 port specification is removed.

2) if it is an image or sound file
2.1) get the next entry

Extraneous information such as the requests for image files, sound files are simply ignored and we skip these kind of entries.

2') if it is not an image or sound file

So then the record stands for a possible edge in a path traversed by the particular visitor. And the following operation are done one by one:

2'.1) concatenate the referrer and target to the entry corresponding domain in the hash table

A hash table is used for storing the requests of a particular visit temporarily. The key field of the hash table entries is the domain name and the each request for a file from that domain is concatenated with each other and also with the last request's timestamp and stored in the table corresponding to that domain.

2'.2) for each entry in the hash table exceeding the threshold value
2'.2.1) find the possible paths for the particular visit and delete it from the hash table

I check if there is any connection from a certain domain that exceeds the time threshold (30 minutes). For these entries, the algorithm finds the possible paths traversed using the information of each request from the same domain.

In Figure 1, the requests of a visitor in the web server access log are shown in the format of:

$X \rightarrow Y$, the file Y is accessed using a link in page X.

If the requests in the access log are :

$A \rightarrow B, B \rightarrow C, B \rightarrow D,$

then in reality the visitor traversed probably the path:

$A \rightarrow B \rightarrow C \rightarrow B \rightarrow D.$

Because of caching the backing operation of $C \rightarrow B$ is not reflected into the access log.

We have to take this fact into account. So then the PathFinder2K finds the following paths:

$A \rightarrow B \rightarrow C, A \rightarrow B \rightarrow D.$

REQUESTS	FOUND PATHS
$A \rightarrow B$	
$B \rightarrow C$	$A \rightarrow B \rightarrow C$
$B \rightarrow D$	$A \rightarrow B \rightarrow D$

Figure 1.

Running Time of the Algorithm:

1) parse the attributes of the record
2) if it is an image or sound file
2.1) get the next entry

First step takes $O(1)$ time when we take the length of each records in the access log is bounded by a constant number. The lines 2 and 2.1 are also $O(1)$ because of single comparison.

2'.1) concatenate the referrer and target to the entry corresponding domain in the hash table

As we are using hash table data structure, it takes $O(1)$ time to concatenate and insert it into hash table.

2'.2) for each entry in the hash table exceeding the threshold value
2'.2.1) find the possible paths for the particular visit and delete it from the hash table

In my experiment the maximum number of entries in the hash table is 99 for a certain time. If we assume that the number of entries in the

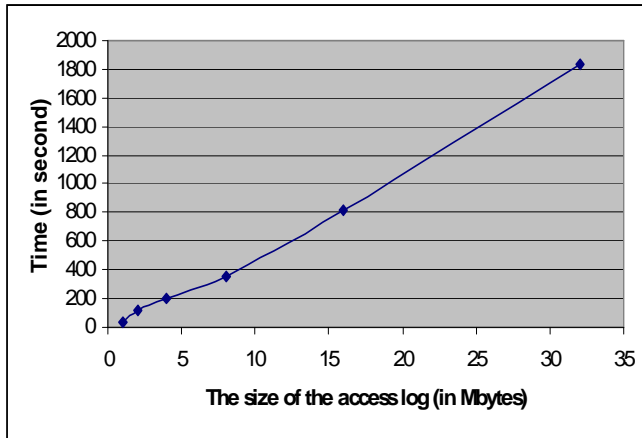


Figure 2.

hash table is bounded with a constant and use amortized analysis method this step is also $O(1)$ time but with a high hidden constant.

Finally, as it takes $O(1)$ time for each record in the web server access log, the total time for this algorithm is $O(N)$, where N is the number of records record in the web server access log.

3. Experiments with PathFinder2K

In this part I report the results of my experiments. I use the access log of web server of Computer Science Department in Bilkent University. This site is a typical site of a CS department. And receives more than 20000 hits per day.

In Figure 2, the relation between the size of the web server access log and the time to process that data is given. As I indicate that it is linear algorithm, the time changes linearly with respect to the size of the log file.

After all, it turns out that on the average it takes 50 sec to process 1 Mbytes data in web server access log.

In Figure 4, a sample of the output of the PathFinder2K is given. When I examine the relation between the size of the log file and the number of paths found, I see that the number of the new discovered paths decreases in the course of time. This result can be seen more clearly in the Figure 3.

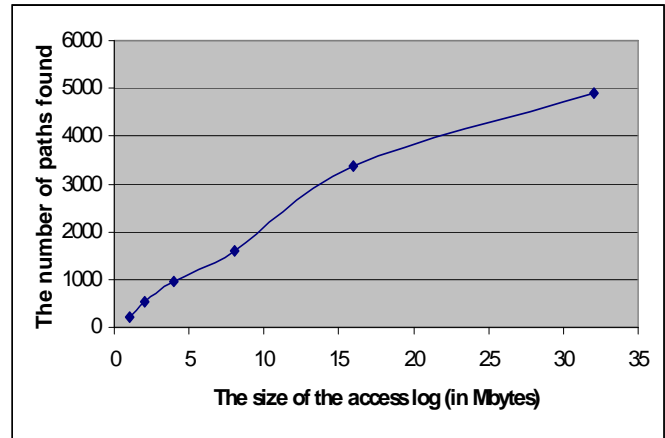


Figure 3.

4. Discussion and Conclusions

As I mentioned above it impossible to detect all the paths exactly. I said that PathFinder2K extracts the *possible* paths from the web server access logs because of several reasons:

It is not easy to handle the actions that are not reflected to the web serve access logs, such as backing or forwarding. In order to enhance performance all browsers use caching technique for storing the recently requested files, and if these files are requested again the locally stored files are served immediately without contacting the actual web server. As a result it is impossible to determine the exact action that are taken by the visitors.

Another similar problem is the proxies. Proxy servers copy the documents that are mostly requested to themselves and subsequently requested these files are given the users from their storage. This service speeds up the internet for their users. So in recent years many people use proxy server for fast internet access. But this prevents the web logs recording the actual actions of users.

Moreover when we take the difficulty of determining the separate visits into account, it turns out that the knowledge discovered from the web server access logs are not exactly true. This fact is not valid for only PathFinder2K but also all data mining applications on these log files.

STATISTICS FROM WEB SERVER www.cs.bilkent.edu.tr

Period: April 2, 2000 at 7 pm - April 29, 2000 at 5 am

PATHS

Paths with at least 2 references were included in this list.

No. of references	Length (links)	Path
7	2	/~guvenir/courses/ => /~guvenir/courses/CS558/ => /~guvenir/courses/CS558/resources.html
7	2	/courses.html => /~guvenir/courses/CS558/ => /~guvenir/courses/CS558/resources.html
7	1	/~guvenir/courses/ => /~guvenir/courses/CS550/
7	1	/courses.html => /~guvenir/courses/CS315/
7	1	/~guvenir/courses/CS558/ => /~guvenir/courses/CS558/review-form.txt
7	1	/~guvenir/Oku/ => /~guvenir/Oku/oku.zip
7	1	/~guvenir/CATT/index1.html?Sozluk=>Turkish?%3C%3D%3E?English?Dictionary => /~guvenir/CATT/TalkingPictures/
6	4	/ => /courses.html => /CS492/ => /~guvenir/courses/CS492/ => /~guvenir/courses/CS492/Projects.htm
6	4	/ => /faculty.html => /~guvenir/ => /~guvenir/toc.html => /~guvenir/courses/
6	4	/ => /faculty.html => /~guvenir/ => /~guvenir/toc.html => /~guvenir/courses
6	3	/faculty.html => /~guvenir/ => /~guvenir/main.html => /~guvenir/courses/
6	3	/~atolga/ => /~atolga/me.html => /~guvenir/courses/CS476/ => /~guvenir/courses/CS476/quizzes.html
6	3	/courses.html => /CS492/ => /~guvenir/courses/CS492/ => /~guvenir/courses/CS492/Projects.htm
6	3	/faculty.html => /~guvenir/ => /~guvenir/toc.html => /~guvenir/courses/
6	3	/~guvenir/courses/ => /~guvenir/courses/CS558/ => /~guvenir/courses/CS558/DataSets/ => /~guvenir/courses/CS558/DataSets/Begendik
6	3	/faculty.html => /~guvenir/ => /~guvenir/toc.html => /~guvenir/courses
6	3	/~guvenir/main.html => /~guvenir/courses/ => /~guvenir/courses/CS558/ => /~guvenir/courses/CS558/DataSets/
6	3	/~guvenir/courses/ => /~guvenir/courses/CS558/ => /~guvenir/courses/CS558/DataSets/ => /~guvenir/courses/CS558/DataSets/Begendik/
6	2	/~atolga/me.html => /~guvenir/courses/CS476/ => /~guvenir/courses/CS476/quizzes.html
6	2	/CS492/ => /~guvenir/courses/CS492/ => /~guvenir/courses/CS492/Projects.htm
6	2	/~guvenir/courses/CS558/ => /~guvenir/courses/CS558/seminar.html => /~guvenir/courses/CS558/SeminarPapers/Efficient_Text_Categorization.ps
6	2	/~guvenir/courses/ => /~guvenir/courses/CS492/ => /~guvenir/courses/CS492/schedule.htm
6	2	/~guvenir/ => /~guvenir/main.html => /~guvenir/publications.html
6	2	/~guvenir/toc.html => /~guvenir/courses/ => /~guvenir/courses/CS492/
6	1	/~guvenir/CATT/TalkingPictures/ => /~guvenir/CATT/TalkingPictures/govde.html
6	1	/~guvenir/courses/CS315/lex-yacc/ => /~guvenir/courses/CS315/lex-yacc/float.l
6	1	/~guvenir/CATT/ => /~guvenir/CATT/jTuSC/
6	1	/~guvenir/CATT/ => /~guvenir/CATT/Conversation/
6	1	/t-index.html => /~guvenir/CATT/
6	1	/~guvenir/CATT/ => /~guvenir/CATT/TalkingPictures/
6	1	/~guvenir/toc.html => /~guvenir/publications.html
6	1	/~guvenir/CATT/index1.html?Sozluk=>Turkish?%3C%3D%3E?English?Dictionary => /~guvenir/CATT/GrammarTutor

Figure 4.

In addition as the usage of internet grows rapidly, web agents become widespread. Web spiders, indexers, link checkers download almost all files that they can reach and leave their footprints in the log files.

Future Work:

As a future work I can propose the following ideas:

- Predicting the user's next action:

The more advanced research on this topic may lead us to predict the user's next action and we may emphasize the links that are likely to be traversed by the user.

- Predicting the user's final aim:

Instead of the next action of the user, we can strive for predicting the final goal of the users. By this way we can offer several targets that are likely to be reached by the users in the very beginning.

- Classifying the users according to the paths that they traversed:

The information of the paths traversed by the users may be used to classify the user.

References

- [1] M. S. Chen, J. S. Park, and P. S. Yu. Data mining for path traversal patterns in a web environment. In Proc. 16th Int'l Conf. Distributed Computing Systems, pages 385-392, May 1996.
- [2] K. Keeker. Improving web site usability and appeal. In <http://msdn.microsoft.com/msdn-online/workshop/management/planning/improving-site-usability.asp>, 2000.
- [3] M. Perkowitz, O. Etzioni. Adaptive web sites: Automatically learning from user access patterns. In Proceedings of the Sixth Int. WWW Conference, Santa Clara, California, April 1997.
- [4] M. Perkowitz, O. Etzioni. Adaptive web sites: Automatically synthesizing web pages. AAAI98.1998.
- [5] J. Riphagen. In Search of the elusive user: Gathering information on web server access. In <http://www.ncsa.uiuc.edu/Edu/trg/webstats/>, 2000.
- [6] L. Tauscher, S. Greenberg. Revisitation Patterns in World Wide Web Navigation. In Proc. CHI 97 Conference on Human Factors in Computing Systems, Atlanta, March 1997.
- [7] O. R. Zaïane, M. Xin, J. Han. Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. In Proc. ADL'98 (Advances in Digital Libraries), Santa Barbara, April 1998.