A Pattern Restore Method for Restoring Missing Patterns in Server Side Clickstream Data

I-Hsien Ting, Chris Kimble, and Daniel Kudenko

Department of Computer Science, The University of York
Heslington, York YO10 5DD, United Kingdom
{derrick, kimble, kudenko}@cs.york.ac.uk

Abstract. When analyzing patterns in server side data, it becomes quickly apparent that some of the data originating from the client is lost, mainly due to the caching of web pages. Missing data is a very important issue when using server side data to analyze a user's browsing behavior, since the quality of the browsing patterns that can be identified depends on the quality of the data. In this paper, we present a series of experiments to demonstrate the extent of the data loss in different browsing environments and illustrate the difference this makes in the resulting browsing patterns when visualized as footstep graphs. We propose an algorithm, called the ***P****attern* ***R****estore* ***M****ethod* (PRM), for restoring some of the data that has been lost and evaluate the efficiency and accuracy of this algorithm.

1 Introduction

Clickstream data is a principle resource for the analysis of user's browsing behavior on a website. The process of analyzing clickstream data and taking actions as a result of that analysis can be viewed as a circular process (see Figure 1) [7]. There are many open problems in this process. For example, the data integrity problem between the first step and the second step, the problem of measuring the "interestingness" of a pattern between the second and the third step, the problem of assessing the significance of a pattern between the third step and the fourth step, and the final problem of how to close the loop.

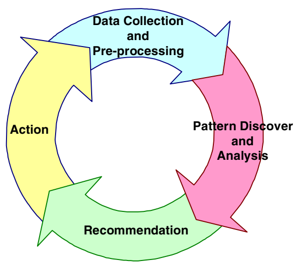

Data Collection and Pre-processing
Pattern Discover and Analysis
Recommendation
Action

Fig. 1. The process of web usage mining for E-commerce [Adopted form 7]

The quality of the pattern discovery and analysis step depends on the quality of the data that is produced after the data pre-processing step [1]. Missing data in server side logs is a common problem in this respect. For example, clickstream data can be lost due to the use of the “back” button in the browser, which does not show in the server side clickstream data. Consequently, some of the user’s browsing patterns may be misinterpreted due to the missing data.

In this paper, we describe a series of experiments that show how clickstream data is lost at the server side. We examine the effects of different browsers, different ISPs and different proxy servers and propose a pattern restoration method (PRM) to overcome the caching problem caused by use of the browser’s "back" button.

The structure of this paper is as follows. Some related literature is discussed in section 2. Section 3 briefly describes the concept of a footstep graph, the experiment design and results. Section 4 describes the concept and the process of the PRM algorithm in detail and section 5 evaluates the efficiency and accuracy of the PRM algorithm. Section 6 concludes the paper.

2 Data Pre-processing for Web Usage Mining

In web usage mining, there are three main sources of usage data. These are client side data, intermediate data and server side data [8]. Client side data is normally collected by installing an agent in the client user’s computer or using embedded script in the web page. No data is lost in this process and the data is complete [5].

However, server side data is the most common source of data for web mining, as it is easy to collect. Unfortunately, raw server side data contains much noise and is usually incomplete [6]. Therefore, research has focused on pre-processing server side data to improve it’s quality. The pre-processing is an essential step but one that consumes a large proportion of the processing time in web usage mining [4]. Colley et al. [3] have developed a series of steps for data pre-processing for web usage mining. These include data cleaning, user identification, session identification and data formatting.

In recent years, the use of "bots" (search engine robots and web crawlers) has become widespread and a single bot can visit a website many times. However, these visits do not represent “real” users and are simply seen as noise in the server logs. Therefore, some systems try to find and delete these requests as an initial step in pre-processing [10]. In addition to this noise, incomplete information is also a problem when mining server side data. In [2], two problems that cause the loss of information in server side clickstream data have been pointed out.

The first is that there may be data missing in the server side data due to a frame-based website. Such a website may cause errors in user browsing time measurements, because every page of the framesets is treated as a single page and an independent request in the server side data. This problem has been solved by Spiliopoulou et al., who use a referrer-based heuristic algorithm to reconstruct the incomplete session record [9]. Others have used the web site structure to reconstruct incomplete session data [1].

The second is the problem caused by web page caching, e.g. when the “back” function of a browser is used. When the "back button" is used, the browser does not send a request to the server and consequently the user’s browsing behavior is not recorded in server's logs. However, the user’s backward browsing is important information for web usage mining, and this kind of missing data problem will affect the quality of any patterns discovered.

3 The Missing Data Problem in Server Side Clickstream Data

In this paper, we propose a method for restoring this lost data by using referrer information from the clickstream data and website structure. However, first we present the results of a series of experiments to investigate the missing data problem. In order to visualize the user's browsing behavior, we use footstep graphs [11]. A footstep graph is based on a simple x-y plot: the x-axis in this graph denotes time and the y-axis represents the nodes (i.e. pages) on the user's browsing route. Horizontal distance in this graph represents the time between two nodes, and changes in the vertical axis indicate a transition from one node to another node (See figure 2). An *Upstairs* pattern in the footstep graph is found when the user only moves forward. A *Mountain* pattern occurs when a *Downstairs* pattern follows an *Upstairs* pattern. A *Valley* pattern occurs after the user goes back to a page they visited and then goes to another new page. A *Fingers* pattern in a footstep graph indicates that a user is in a browsing loop.

3.1 Design of an Experiment to Find the Scope of the Missing Data Problem

The goal of the experiment is to look at the difference between browsing patterns as they appear at the client and server side in order to identify what data is lost in different user browsing environments. Three different browsing environments were chosen for the experiments using different browsers, ISPs and proxy servers. Additionally, four browsing routes representing the *Upstairs* (figure 2a), *Mountain* (figure 2c), *Fingers* (figure 2e) and *Valley* pattern (figure 2g) were used.

Three widely used browsers were chosen for the experiments: Microsoft Internet Explorer, Netscape Navigator and Opera. Different users, using different ISPs to access the web site, were asked to follow a predetermined browsing route. The tests were repeated using different proxy server settings.

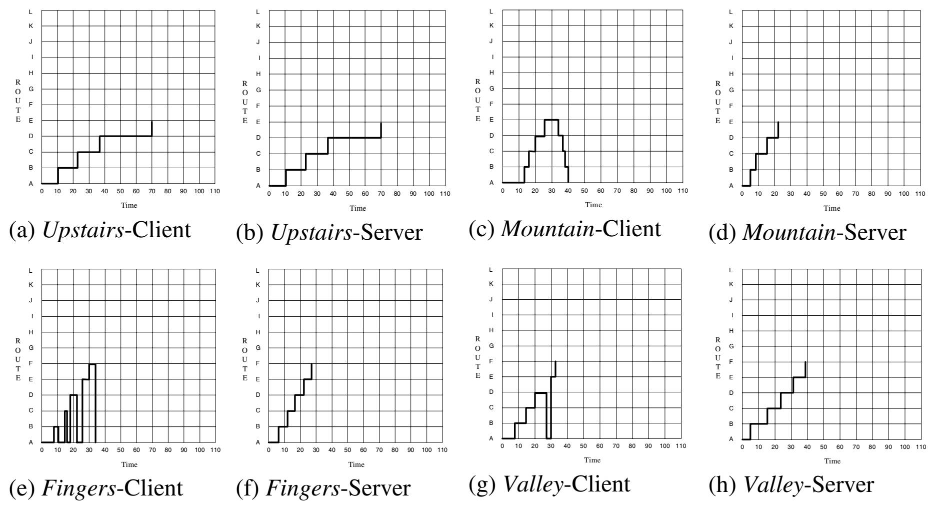
3.2 The Results

3.2.1 Using Different Browsers

The main purpose of this test is to see if there is any difference in the server side data when different browsers are used to browse a predetermined route. We used the default setting for the cache and did not set a proxy server in this part of the experiment.

Internet Explorer & Netscape Navigator. Internet Explorer & Netscape Navigator both use the same default cache setting. The default setting is that when the user uses the backward and forward function, the browser does not sent the request to the server but use the cached webpage. Thus, server side data is lost if the user clicks the backward and forward buttons. Figure2 shows the results of this experiment. There is no difference between the Upstairs patterns in the client side and server side data when using Internet Explorer and Netscape Navigator (See (a) and (b) in the figure2), but data for all other patterns are lost due to caching.

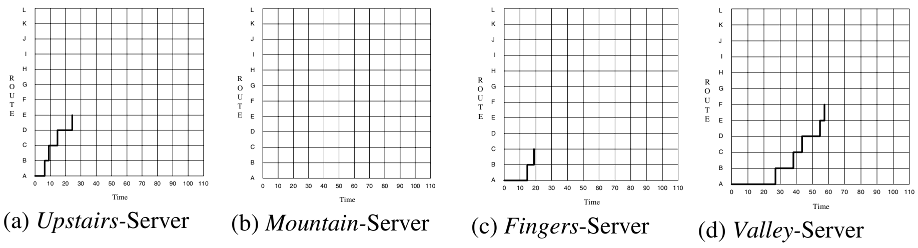
Opera. Opera browser does not send a request to the website server when the user opens a web page that has been browsed within the last five minutes. Thus, much of the server side information is lost. For example, figure 3b should be a Mountain pattern, but there is no server side data because all of the web pages had been requested before. Figure 3c has been affected in the same way.



(a) *Upstairs*-Client (b) *Upstairs*-Server (c) *Mountain*-Client (d) *Mountain*-Server

(e) *Fingers*-Client (f) *Fingers*-Server (g) *Valley*-Client (h) *Valley*-Server

Fig. 2. Browsing patterns when using Internet Explorer and Netscape Navigator



(a) *Upstairs*-Server (b) *Mountain*-Server (c) *Fingers*-Server (d) *Valley*-Server

Fig. 3. Server side browsing patterns when using Opera

3.2.2 Using Different ISPs

The second test investigated the server side patterns formed by using different ISPs. Again, users were asked to follow a pre-defined route through a given web site. Here only Internet Explorer or Netscape Navigator, without proxy server settings, were used. The commercial ISPs used in this experiment (Onetel, Hinet, SeedNet, BT and AOL) did not cache the experimental web pages and all of these ISPs created the expected patterns in server side data. However, some server side data was lost when using Loughborough University as an ISP as Loughborough appeared to cache web pages that had been browsed before. In this case, the user's browsing pattern at the server side appeared similar to the pattern produced by the Opera browser.

3.2.3 Using Different Proxy Servers

In the third set of experiments, we tested the impact of different proxy settings. The server side image of the user's browsing pattern depends on the proxy server's type. If the proxy server caches web pages for the user but does not send a request to the website server, some server side clickstream records will be lost and the server side pattern will be similar to the pattern when using the Opera browser. If the proxy server still sends a request to the website server, the server side pattern will suffer no losses.

4 The Pattern Restore Method (PRM) Algorithm

Having identified the potential scope of the problem, we now turn to a possible solution. Below we describe a method to restore the lost data that we call the pattern restore method (PRM). The basic idea behind the PRM algorithm is to use referrer information and the web site structure to restore the lost data. The PRM algorithm has two phases, the details of which are presented below.

4.1 The First Phase of the PRM Algorithm

4.1.1 The Process and Algorithm of the First Phase PRM Algorithm

Before we discuss the PRM algorithm, the general description of a clickstream format after pattern restoration is defined as:

C_i: [U_i, T_i, L_i, R_i, Mark]
C_i: C_i denotes the ith clickstream in one user session
U_i: Ui denotes the IP address of i^{th} clickstream in one user session (In our example, we use the IP address to identify the session. Therefore, all the U_i in one session are the same)
T_i: T_i denotes the timestamp of i^{th} clickstream in one user session
L_i: L_i denotes the requested URL of i^{th} clickstream in one user session.
R_i: R_i denotes the referrer information of the i^{th} clickstream in one user session
Mark: Mark is used to distinguish original from restored clickstream data.

The first phase of the PRM algorithm uses the referrer information to restore the missing clickstream data. First, the algorithm checks the first clickstream record in a session to check if it has any referrer information. If it does, and if the referrer belongs to the current site, some clickstream data must have been lost and needs to be restored. The restoration rule uses the referrer information of the first record as the URL of a new record. The referrer information of this new record is set to a null value, the time stamp is set to a suitable time (in this case 5 seconds earlier than the first record) and the record is marked as "Restored". The general description of the restored clickstream record in this step is **$C_{1'}$ [U_1, T_1-5, R_1, -, Mark]**. Figure 4 shows an example of original clickstream data in a session and Figure 5 shows the restored data for the same session.

(1) 61.59.121.221, 16:02:54, http://www-users.cs.york.ac.uk/~kimble/research/ research. .html, http://www-users.cs.york.ac.uk/~kimble/
(2) 61.59.121.221, 16:03:00, http://www-users.cs.york.ac.uk/~kimble/teaching/teach.html, http://www-users.cs.york.ac.uk/~kimble/

Fig. 4. The clickstream data before the first phase of the PRM algorithm

(1) 61.59.121.221, 16:02:49, http://www-users.cs.york.ac.uk/~kimble/, -, Restored
(2) 61.59.121.221, 16:02:54, http://www-sers.cs.york.ac.uk/~kimble/research/research.html, http://www-users.cs.york.ac.uk/~kimble/
(3) 61.59.121.221, 16:03:00, http://www-users.cs.york.ac.uk/~kimble/teaching/teach.html, http://www-users.cs.york.ac.uk/~kimble/

Fig. 5. The clickstream data after the first step of the first phase of the PRM algorithm

The next step of the first phase of the PRM algorithm compares the referrer URL in each clickstream record with the target URL from the previous record. If they are the same (such as record 1 and 2 in figure 5), it means that the user moved directly from one URL (e.g. record 1 in figure 5) to another (e.g. record 2 in figure 5). Thus, it is assumed that there is no missing data and no need to perform pattern restoration. However, if the target and referrer are different, it means that some data must have been lost before the current record (e.g. between record 2 and 3 in figure 5).

In this case, the algorithm will restore a record using the information in the current record. The IP address of the restored record is set to the same as the current record and the timestamp is set to be midway between the time between the current record and the previous record (i.e. midway between the times of record 2 and 3 in figure 5). The target URL of the restored record is taken from the referrer of the current record and the referrer of the restored record is set to null. Finally, the marker of this record is set to "Restored". The general description of the restored clickstream record in this step is **$C_{i'}$ [U_1, $(T_{i-1}+T_i)/2$, R_i, -, Mark]**.

(1) 61.59.121.221, 16:02:49, http://www-users.cs.york.ac.uk/~kimble/, -, Restored
(2) 61.59.121.221, 16:02:54, http://www- sers.cs.york.ac.uk/~kimble/research/research.html, http://www-users.cs.york.ac.uk/~kimble/, **Original**
(3) 61.59.121.221, 16:02:57, http://www-users.cs.york.ac.uk/~kimble, -, Restored
(4) 61.59.121.221, 16:03:00, http://www-users.cs.york.ac.uk/~kimble/teaching/teach.html, http://www-users.cs.york.ac.uk/~kimble/, **Original**

Fig. 6. The clickstream data after the second step of the first phase of the PRM algorithm

Record (3) in figure 6 is an example of a restored record created by comparing records (2) and (3) in figure 5. The algorithm will continue until the last record in a session has been processed. The pseudo code of the first phase of the PRM algorithm is given in figure 7.

```
1:  IF $R_1$ is not null then
2:          IF $R_1$ belongs to website THEN Restore $C_{1'}$ [$U_1$, $T_1$-5, $R_1$, -, Mark]
3:  END IF
4:  FOR i=2 to n
5:          IF $L_{i-1} \neq R_i$ THEN Restore $C_{i'}$ [$U_1$, $(T_{i-1}+T_i)/2$, $R_i$, -, Mark]
6:  END FOR
```

Fig. 7. The pseudo code of the first phase of the PRM algorithm

4.1.2 Server Side Browsing Patterns After the First Phase of the PRM Algorithm

After the first phase of the PRM algorithm, *Fingers* patterns and *Valley* patterns have been restored. Figure 8(a) shows an example of a *Fingers* pattern recorded in a client side; Figure 8(b) shows the corresponding pattern in the server side log. The pattern now looks like an *Upstairs* pattern because data from pages browsed using the back function has been lost. Figure 8(c) shows the restored pattern after the first phase of the PRM algorithm has been run. Figures 9(a) to 9(c) show a similar restoration process with a *Valley* pattern.

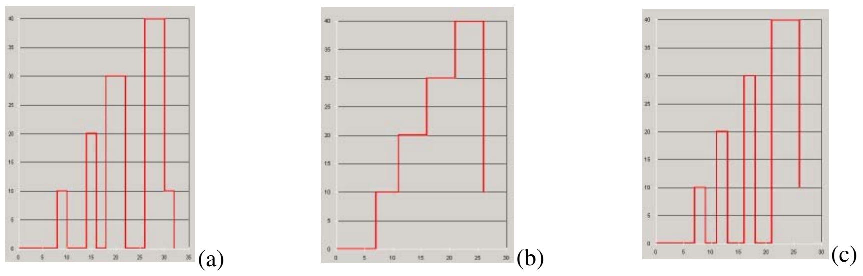


Fig. 8. (a) The *Fingers* pattern in client side data (b) The *Fingers* pattern appears as an *Upstairs* pattern in server side data (c) The restored *Fingers* pattern from the server side data

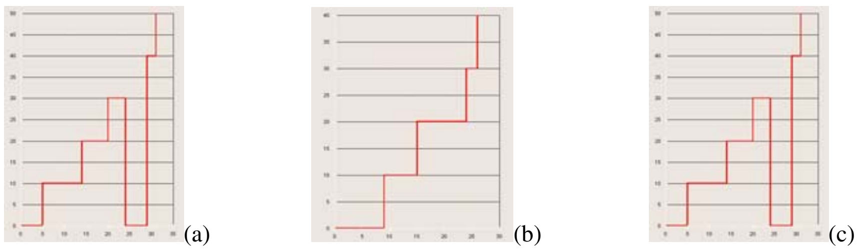


Fig. 9. (a) The *Valley* pattern in client side data (b) The *Valley* pattern appears as an *Upstairs* pattern in server side data (c) The restored *Valley* pattern from the server side data

Although the algorithm works with *Fingers* and *Valley* patterns, *Mountain* patterns cannot be restored using this method. Figure 10(a) shows a *Mountain* pattern in the client side data and the Figure 10(b) shows the corresponding pattern in server side data. The data lost by backward browsing cannot be restored and the pattern after the first phase of the PRM algorithm looks the same as in figure 10(b).

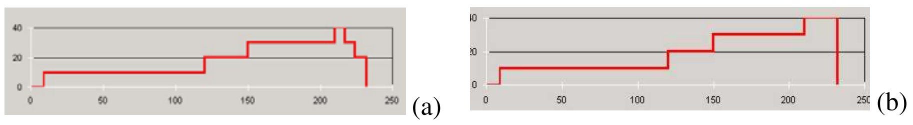


Fig. 10. (a) The *Mountain* pattern in client side data (b) The *Mountain* pattern as it appears in the server side data and/or after the first phase of the PRM algorithm

4.2 The Second Phase of the PRM Algorithm

4.2.1 The Process and Algorithm of the Second Phase of the PRM Algorithm

As described above, the *Downstairs* part of *Mountain* patterns cannot be restored by the first phase of the PRM algorithm. Therefore, a second phase is needed to replace the missing data. The second phase of the PRM algorithm uses information from the website's link structure to insert the "most probable" browsing path of the user. To do this, a list of pages and the link structure for the site must be maintained.

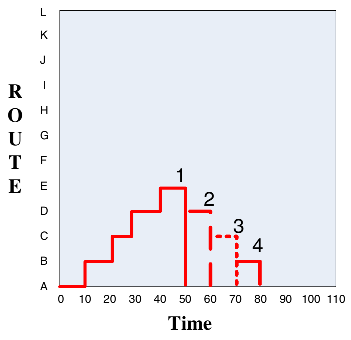


Fig. 11. An example of the second phase of the PRM algorithm

Figure 11 shows a simple example of how this algorithm works; the *Downstairs* part of *Mountain* pattern begins at node E. Firstly, the algorithm will check the link structure of node E. If node E has a direct link to node A, then the restored pattern

```
1:  FOR i=2 to n
2:  IF $R_i \neq L_{i-1}$ THEN
3:          IF $L_{i-1}$ has link to $L_i$ THEN
4:                  Restore $C_i$ [$U_i$, $T_i$, $L_i$, $L_{i-1}$, Mark]
5:          ELSE
6:                  FOR j=1 to m
7:                  IF $L_{i-1-j} \neq L_i$ THEN
8:                          IF $L_{i-1-j}$ has link to $L_{i-1-j+1}$ and has link to $L_i$ THEN
9:                                  k=k+1, Restore $C_i^k$ [$U_i$, $T_i$, $L_{i-1-j}$, $L_{i-1-j+1}$, Mark]
10:                                 Restore $C_i$ [$U_i$, $T_i$, $L_i$, $L_i^k$, Mark]
11:                                 Exit for
12:                         ELSE IF $L_{i-1-j}$ has link to $L_{i-1-j+1}$ THEN
13:                                 k=k+1, Restore $C_i^k$ [$U_i$, $T_i$, $L_{i-1-j}$, $L_{i-1-j+1}$, Mark]
14:                         END IF
15:                 END IF
16:                 END FOR
17:                 FOR all restored clickstream k=1 to p $T_i^k=(T_i-T_{i-1})/p$
18:         END IF
19: END IF
```

Fig. 12. The pseudo code of the second phase of the PRM algorithm

will look like pattern 1 in figure 11. If node E does not have a direct link to node A, it means some data must have been lost. In this case, the algorithm will search back from the nodes before node E to find the closest node that has a direct link to it. For example, the closest node to node E in figure 11 is node D, which has a direct link to node E. The algorithm then checks the link structure of node D. If there are direct links to node A, node D will be restored and the pattern is shown as 2 in figure 11. The algorithm will continue to loop until it finds a node that has a direct link to node A or the node is equal to A.

The pseudo code for the second phase of the PRM algorithm is shown as figure 12. Here, m denotes the total records before the current record, and k is used to count the total number of restored records in a single restoration session.

4.2.2 The User's Browsing Pattern After the Second Phase of the PRM Algorithm

After the second phase of the PRM algorithm, some of the lost clickstream data can be restored. For example, figure 13(a) shows how a *Mountain* pattern appears in the server side logs; figure 13(b) shows the restored *Mountain* pattern.

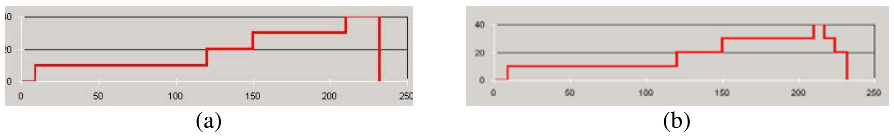


Fig. 13. (a) The original *Mountain* pattern as it appears in the server side data (b) The restored *Mountain* pattern after the second phase of the PRM algorithm

Unfortunately, there is a weakness of the second phase of the PRM algorithm: when attempting to restore a *Valley* pattern, some errors may occur.

Figure 14(a) shows how a *Valley* pattern appears in the server side log. After the second phase of the PRM algorithm, the pattern becomes that shown in figure 14(b). However, the original pattern was as shown in figure 14(c), not figure 14(b).

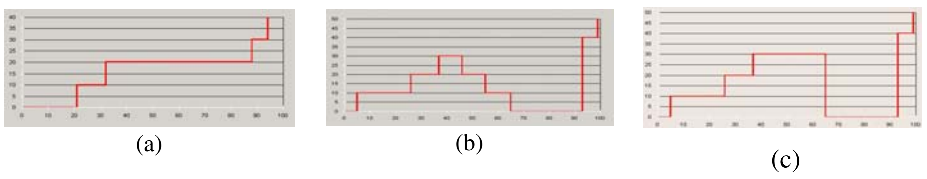


Fig. 14. (a) The server side data (b) The restored *Valley* pattern (c) The original *Valley* pattern

5 Evaluation of the PRM Algorithm

To evaluate the performance of the PRM algorithm, we tested the processing time of the PRM algorithm under different restoration rates (i.e. the number of restored records / total number of records after restoration) and the restoration accuracy under different user's browsing patterns. The evaluation was performed under Windows XP operation system, 512Mb RAM, 2.4 GHz Intel Pentium 4 CPU.

5.1 The Evaluation of the Processing Time of the PRM Algorithm

The tested number of original records ranged from 10 records (about 2 Kbytes file size) to 51200 (About 30 Mbytes file size). The results are shown in figure 15.

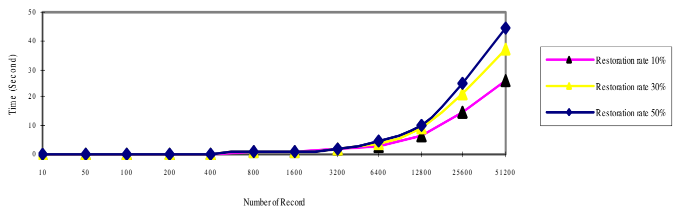


Fig. 15. Processing time of the first phase of the PRM algorithm

This shows that the rate of increase in processing time is similar under different restoration rates. When running the first phase of the PRM algorithm to process 51200 records, the processing time for 10% restoration rate is about 26 seconds; for 30% it is about 36 seconds and for 50% it is about 44 second. The processing time needed is not particularly high, even when given a large amount of data.

The same testing method is also be used to evaluate the second phase of the PRM algorithm; the results are shown in figure16. Here, the processing time with a restoration rate of 10 % is quite low and does not increase very much as the number of records processed increases.

When the restoration rate has reached 30%, the time / number of records slope has increased substantially. The reason for this is that the second phase of the algorithm needs to check the site structure. When the restoration rate and record number increases, the algorithm needs to check the structure table many times. Thus, the processing time increases as the restoration rate and the number of records increases.

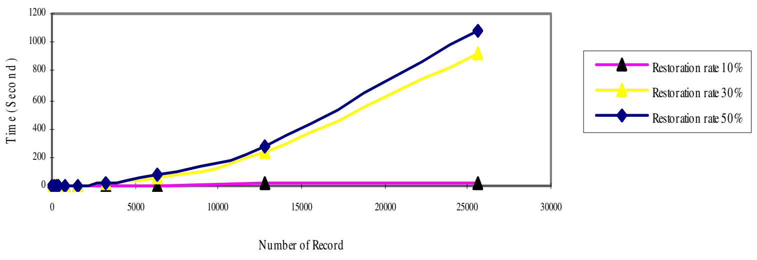


Fig. 16. The evaluation of processing time of the second phase of the PRM algorithm

5.2 The Evaluation of the Accuracy of the PRM Algorithm

In order to evaluate the accuracy of the algorithm, we designed ten different browsing routes that produced different *Fingers*, *Valley* and *Mountain* patterns, run the PRM algorithm on each of the resultant sets of server side data, and compared the result to the original pattern.

Table 1. The accuracy of restoration of the first phase of the PRM algorithm under different user's browsing pattern

	Finger			***Valley***			***Mountain***		
Route	Correct	Error	Rate	Correct	Error	Rate	Correct	Error	Rate
1	9	0	1	9	0	1	5	4	0.556
2	10	0	1	10	1	0.909	5	4	0.556
3	7	1	0.875	9	0	1	5	4	0.556
4	7	0	1	9	0	1	6	4	0.600
5	8	1	0.889	7	1	0.875	6	3	0.667
6	9	0	1	6	1	0.857	7	4	0.636
7	11	1	0.917	8	0	1	5	5	0.500
8	8	0	1	9	0	1	5	4	0.556
9	9	0	1	7	0	1	4	4	0.500
10	11	0	1	10	0	1	6	2	0.750
Avg.			0.968			0.964			0.587

Table 1, shows the restoration accuracy of the first phase of the PRM algorithm when restoring *Fingers*, *Valley* and *Mountain* patterns. Although the restoration accuracy for the *Fingers* and *Valley* is high, restoring *Mountain* patterns is only about 60% accurate, as the first phase of the PRM algorithm cannot deal with the *Downstairs* part of *Mountain* patterns.

Table 2. The accuracy of restoration of the second phase of the PRM algorithm under different user's browsing pattern

	Fingers			***Valley***			***Mountain***		
Route	Correct	Error	Rate	Correct	Error	Rate	Correct	Error	Rate
1	9	1	0.900	7	2	0.778	8	1	0.889
2	10	0	1	8	3	0.727	9	0	1
3	6	2	0.750	7	2	0.778	7	2	0.778
4	7	0	1	9	2	0.818	9	1	0.900
5	7	0	1	8	3	0.727	8	1	0.889
6	9	0	1	7	2	0.778	11	0	1
7	10	0	1	8	2	0.800	8	2	0.800
8	8	1	0.889	9	3	0.750	7	2	0.778
9	9	0	1	7	2	0.778	6	2	0.750
10	11	0	1	10	3	0.769	8	0	1
Avg.			0.954			0.770			0.878

In table 2, the restoration accuracy rate of *Mountain* patterns has been increased by about 30%, but the restoration accuracy rate of *Valley* patterns has been decreased by about 20%. The reason for this is that the second phase of the PRM algorithm can deal with the *Downstairs* part of the *Mountain* pattern but some *Valley* patterns are now restored as *Mountain* patterns (see figure 14).

6 Conclusion

This paper describes a series of experiments to examine the differences between click-stream data on the client side and on the server side. We found that due to caching, some data is lost. We subsequently developed and evaluated an algorithm for restoring the lost data. The evaluation showed that the algorithm could restore most of the lost patterns. Although not perfect, we believe that the method we describe can be used as the last part of the data pre-processing step in web mining. Consequently, we believe that by using the PRM algorithm the accuracy of the pattern discovery can be increased.

References

1. Berendt, B., Mobasher, B., Nakagawa, M. and Spiliopoulou, M.: The Impact of Site Structure and User Environment on Session Reconstruction in Web Usage Analysis. Proceedings of the WebKDD 2002 Workshop, Edmonton, Alberta, Canada, July (2002) 159-179
2. Clickstream Technologies Plc.: Technical White Paper: A clickstream Though-leadership Paper, http://www.clickstream.com/docs/cswhitepaper.pdf (Access date: 6 September 2004)
3. Cooley, R., Mobasher, B. and Srivastava, J.: Data Preparation for Mining World Wide Web Browsing Patterns, Knowledge and Information System, Vol. 1, No. 1 (1999) 5-32
4. Eirinaki, M. and Vazirgiannis, M.: Web Mining for Web Personalization, ACM Transactions on Internet Technology, Vol. 3, No. 1 (2003) 1-27
5. Fenstermacher, K. D. and Ginsburg, M.: Mining Client-Side Activity for Personalization, Proceedings of the Fourth Workshop on Advanced Issues in Electronic Commerce and Web Information Systems, Newport Beach, California, USA, June (2002) 26-28
6. Kohavi, R.: Mining E-commerce Data: The Good, the Bad, and the Ugly, Proceedings of the KDD 01 Conference, San Francisco, CA, USA (2001) 8-13
7. Lee, J., Podlaseck, M., Schonberg, E., Hoch, R.: Visualization and analysis of clickstream data of online stores for understanding web merchandising, Journal of data mining and knowledge discovery, Vol.5 (2001) 59-84
8. Pierrakos, D., Paliouras, G., Papatheodorou, C. and Spyropoulos, C. D.: Web Usage Mining as a Tool for Personalization: A Survey, User Modeling and User-Adapted Interaction, Vol. 13 (2003) 311-372
9. Spiliopoulou, M., Mobasher, B., Berendt, B. and Nakagawa, M.: A Framework for the Evaluation of Session Reconstruction Heuristics in Web Usage Analysis, INFORMS Journal of Computing, Special Issue on Mining Web-Based Data for E-Business Applications, Vol. 15, Issue. 2 (2003) 171-190
10. Tan, P. N., and Kumar, V.: Discovery of the Web Robot Sessions Based on their Navigational Patterns, Data Mining and Knowledge Discovery, Vol. 6 (2002) 9-35
11. Ting, I. H., Kimble, C., Kudenko, D.: Visualizing and Classifying the Pattern of User's Browsing Behavior for Website Design Recommendation, Proceedings of First International Workshop on Knowledge Discovery in Data Stream (ECML/PKDD '04), Pisa, Italy, 20-24 September (2004) 101-102