Path-Source Oriented Session Identification Based on Linked Referrers and Log Indexing

Chenhan Liao[†] and Jianguo Zheng[††],

Dong Hua University, No.1882 West Yan An Road, Shanghai, P.R.China

Summary
Web usage mining has been widely adopted in various fields such as optimizing site structure, user-behavior analysis, personalized web services and system performance tuning. Although much research has been done against web log mining algorithms and log pre-processing techniques, the study of efficient retrieval of the structured contents for web log mining is seldom reported. In this paper, we first show that people are much more interested in discovering user navigation based on various path-sources. Then, we present a novel session identification algorithm Referrer Link based on discovering linked referrers to serve source-oriented path mining. Next, an efficient web log indexing and path extracting technique is introduced to provide structured web log data for general purpose log mining. The experimental results has shown that the accuracy of the mining results conducted against the sessions discovered by the proposed Referrer Link algorithm is 10% higher in average compared with Time-out approach.
Key words: Session identification, web log mining, path extraction, log indexing.

1. Introduction

From this section, input the body of your manuscript according to the constitution that you had. For detailed information for authors, please refer to [1]. Web logs, trace and store user requests, originally designed for system failure diagnosis. Eventually, its rich information had been utilized in diverse aspects such as intrusion detection [1][2][3], cache performance tuning [4][5], frequent access pattern analysis [6][7][8] and web page access prediction [9] etc.. Most web log mining algorithms work on the structured user sessions or extracted user navigation paths that are supposed to be delivered in log preprocessing stage. However, industrial-level log mining is facing the scalability issue, where just-in-time log preprocessing is demanded to supply well structured set of user navigation paths. Nevertheless, E-business churns out large volume of web logs due to the dramatically booming web services. Thus, dealing with Terabytes log data has become an inevitable challenge in near future.

To analyze user-behaviors, its carrier, well cleaned and structured user navigation path is needed before initializing any further mining procedures. This step is referred as log pre-processing in literatures, which cleans, extracts, transforms and loads structured web log contents for log mining. Interestingly, industrial level applications spend majority of time on data preprocessing procedure but well defined mining procedure. This is due to the nature of web services producing large volume of semi-structured, unclean and distributed log data. For example, log preprocessing is required to filter out business-irrelevant data such like web crawler requests and automatic-loaded multimedia requests. On the other hand, session identification is essential to indicate the unique user identity that has sent a certain group of requests.

Most log preprocessing technique extracts navigation paths based on the user sessions. A user session is considered to be a set of page accesses that occurs during a single visit to a Web site. Many systems use system-defined user sessions such as timeout, which uses all the sequentially accessed pages to form its' path. The sessions are defined either using a pre-determined time-slice window, normally 30 minutes [10], or a client-side cookie. However, the pre-determined time slice window varies in workloads and may contain mixed user-entry points. In addition, cookies can be disabled on client side and may not reflect accurate user behaviors. Thus, we argue that this method lacks reliability in terms of pages' consistency, where pages from different sources are mixed.

In [5], sessions are modeled as a child undirected graph, where each vortex denotes visited web page and arcs between two vortexes are weights. However, undirected graph is a complex data structure, normally needs to be stored in a sparse matrix. The computation complexity of undirected graph is $n \times \log n^2$, where n is the number of vortices. Suppose an e-business website contains 10,000 pieces products (normally much larger than this number), which 10 products forms 1 page. There would be 1,000 pages at least, not to mention other types of pages in this websites. Consequently, it requires $1000 \times \log 1000^2$ computing steps to model this huge graph.

Manuscript received June 5, 2011
Manuscript revised June 20, 2011

In real log mining applications, people are interested in inspecting user behaviors distinguished from various sources. For instance, we may want to know whether the online advertisements we have paid to Google is more effective than any other online media. Hence, the navigation paths are rather access sequences with a landing page having referrer: Google search engine. However, in fact, a user access sequence may contain multiple entries from different sources, which do not reflect users' real clicking-trace. To overcome this problem, we introduce a referrer-linking algorithm along with a log indexing algorithm to effectively retrieve accurate source-oriented user navigation path.

The rest of this paper is organized as follows: section one reviews web log format and commonly used log cleaning technique. Section two represents the proposed session identification algorithm. Section three illustrates various path modeling technique and the proposed log index algorithm. Section four evaluates the performance of our log indexing algorithm. The last section concludes.

2. Log and Log cleaning

Log format may vary in different systems, but log contents are always similar since they share the identical purpose of diagnosing the system failure. There are mainly three types of log format, NCSA Common Log Format, NCSA Extended Log Format and W3C Extended Log Format. The following is a typical W3C Extended Log.

Table 1: W3C Extended Log Format

CIP	Time Stamp	Requested page	Method	Referrer	Status	Return size	Agent

Client IP	Time Stamp	Requested Page	Method	Referrer	Status	Return Size
116.225.71.49	20100415 16:29:53	/house-2347488.html	GET	-	200	25668
116.225.71.49	20100415 16:31:47	/house-2034559.html	GET	http://sh.fangyou.com/sale-a14-k2-l1-m5-n30.html	200	17766
116.225.71.49	20100415 17:24:22	/	GET	http://www.google.com.hk/search?	200	16312
116.225.71.49	20100415 17:25:41	/sale-j3.html	GET	http://sh.fangyou.com/	200	16345
116.225.71.49	20100415 17:26:13	/sale-j3-a14.html	GET	http://sh.fangyou.com/sale-j3.html	200	17302
116.225.71.49	20100415 17:27:48	/sale-j3-a14-c1.html	GET	http://sh.fangyou.com/sale-j3-a14.html	200	202
116.225.71.49	20100415 17:27:55	/index.php	GET	http://sh.fangyou.com/	200	15823
116.225.71.49	20100415 17:28:57	/sale-j3-a14-c1-k2-l1-m5.html	GET	-	200	17526
116.225.71.49	20100415 17:29:46	/sale-a14-c1-k2-l1-m5-j2.html	GET	http://sh.fangyou.com/sale-j3-a14-c1-k2-l1-m5	200	15031
116.225.71.49	20100415 17:30:07	/sale-a14-c1-k2-l1-m5-j1.html	GET	http://sh.fangyou.com/sale-a14-c1-k2-l1-m5-j2.html	200	15017
116.225.71.49	20100415 17:31:00	/sale-a14-c1-k2-l1-m5-j1-n2.html	GET	http://sh.fangyou.com/sale-a14-c1-k2-l1-m5-j1.html	200	14705
116.225.71.49	20100415 17:31:20	/sale-a14-c1-j1-k5-l0-m7.html	GET	-	200	17334
116.225.71.49	20100415 17:31:35	/sale-a14-c1-k5-l0-m7-j4.html	GET	http://sh.fangyou.com/sale-a14-c1-j1-k5-l0-m7.html	200	18099
116.225.71.49	20100415 17:32:12	/sale-a14-c1-k5-l0-m7-j4-n2.html	GET	http://sh.fangyou.com/sale-a14-c1-k5-l0-m7-j4.html	200	17788
116.225.71.49	20100415 17:32:29	/sale-a14-c1-k5-l0-m7-b5	GET	http://sh.fangyou.com/sale-a14-c1-k5-l0-m7-j4-b5.html	200	17176

Fig. 2 An Apache Log Sample

To illustrate what pages referred the current page, most logs have a field called referrer shown in Table.1, which shows the URL indicating where the request comes from. Suppose a user has sent 10 requests in series, however, their referrers can be different. The fact is that users often bounce out the current page and land in again from another source URL. This can yield multiple accessing sessions having various sources. The status field in Table.1 shows request status returned from servers, which can be used to debug system errors.

2.1 Session Identification Based on Referrer Link

One may argue that the definition of a user session is based on the existence of the physical links between two pages and time gap. If two requested pages have no physical links or the time gap between two requests exceed a certain threshold, then the request is supposed to be split into different sessions. Unfortunately, high level applications always concern the user navigation paths originated from specific origins such as search engines, direct browser input, online advertisements or any other online media. Observing and mining the path yielded from specific origins can help people better understand their online product promotion plan, improve user navigation experience and tune the online Ad effects.

We denote a user session from a unique ip with a specific path source as $S_i = \{r_i, p_{ri,1}, p_{ri,2}, p_{ri,i}, p_{ri,n}\}$, where r_i is the i-th path source (root) of the user session of the Ip, $p_{ri,i}$ is the i-th page visit generated from the i-th path source. The user sessions from a unique Ip having multiple various path sources can be described as the combination of the above "light weight" path source-oriented sessions denoted as $S_{ip_i} = \{S_{ip_i,1}, S_{ip_i,2}, S_{ip_i,i}, S_{ip_i,n}\}$, where subscript ip_i represents the i-th unique ip and subscript n is the number of distinct path sources. We then model the entire log file L as a set of user sessions regarding to the distinct IPs. Figure 3 represents an entire log using the above defined sessions.

Client-side cache and firewall may influence the session identification. If user presses a 'BACK' button on the browser, the client system will bring the cached previous page rather than yielding a new page request to the server. Besides, users behind the same firewall may share an identical IP address. Fig.3 shows the procedure of forming user sessions. We define a new session extraction algorithm based on identifying linked referrers. As shown in Fig.1, every requested page is followed by its referrer indicating the last page it has visited. Suppose there is an access sequence {A,B,C} in a time stamp order and page B is already cached on the client side, if page B is referenced again then there would not be a new request to the server. However, once a link of page D on page B is accessed, a new request of page D will be sent to the server since page D has not been cached on the client side yet. As a result, in the server log, the request of page D will be followed by its' referrer, page B. Hence, page B

can be denoted as a child page of page B and page B is automatically filled in the entire access sequence as {A,B,C,B,D}. Although we do not employ the site topology to supplementing full navigation paths, the physical links implied by request-referrer can be extracted based on the Referrer Link algorithm.

For a specified user entry, web log records its origin such as a search engine query. We denote an origin of a user entry as a start point shared by one or more in-site page visits. The origin can be an out-site request or an in-site request such as user saved in-site URL. For each request, we recursively track its referrers until we reach its origin. Such a set of linked pages is defined as a user session in our study.

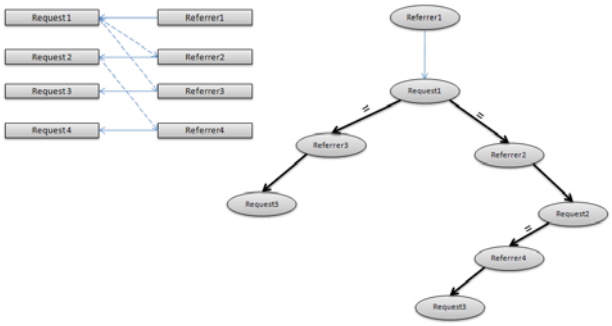

Fig. 3 Linking Referrers to Form User Sessions

From our experience, although this method can extract linked referrer-based path, it suffers the efficiency problem since the most sessions mainly have a forward navigation manner. In this case, every request referrer is checked to form the path, which may yield a lot unnecessary computation. To tackle this problem, we use a forward-backward method to reduce redundant computation. It first goes forward (in terms of the row number in the log) to check each corresponding referrer to see whether it links to the current requested URL. If the linkage cannot be constructed, it will go backward to check if any one of its former requests link to the current URL. This is because users may go back to any of their formerly visited pages and click some other links, which are sequentially recorded in terms of their arrival time. However, for some web workload, users seldom come back to the previously viewed pages but rather keep a forward navigation manner in general. The problem is that the recursive lookup of the page-referrer link is very time consuming since most of page visits come right after its referrer.

2.2 Transaction Identification

The goal of transaction identification is to create meaningful clusters of references for each user. The identified transactions derived from session files can be used for association mining [8], user clustering, page content clustering and etc. Since the extraction of user sessions directly influences the mining results, then the degree of how close the user sessions can represent real user behaviors become a crucial problem. For the evaluation purpose, we apply transaction identification algorithms on the session files produced by the proposed Referrer Link algorithm to see the mining results. In this paper, a user session is thought as a set of many transactions each consisting of a single page reference.

Cooley et al. [11] proposed a transaction identification method, called reference length. This method assumes that the amount of time a user spends on a page is correlated with whether the page is an “auxiliary” or “content” page for that user. By analyzing the histogram of page reference lengths, the authors found that the time spent on auxiliary pages is usually shorter than that spent on a content page, and also that the variance of the times spent on auxiliary pages is smaller than content pages. If an assumption is made about the percentage of auxiliary references in a log, then a reference length can be calculated that estimates the optimal cutoff between auxiliary and content references based on the histogram. Once pages are classified as either auxiliary or content pages, a session boundary will be detected whenever a content page is met.

Another transaction identification method, referred to as maximal forward reference presented by Chen et al. [8]. In this approach, each session is defined as the set of pages from the first page in a request sequence to the final page before a backward reference is made. Here, a backward reference is naturally defined to be a page that has already occurred in the current session. One advantage of the maximal forward reference method is that it does not have any parameters that make assumptions about the characteristics of a particular data set. However, it has the significant drawback that backward references may not be recorded by the server if caching is enabled at the client site.

2.3 Indexing Log

To extract source-oriented sessions efficiently, we index the requests to reduce the computation overhead. Rather than scanning the entire log file, we inversely index the requests using their position in the log file and the corresponding time stamps. Consequently, the scan process of forming Referrer Link based path is confined in a relatively small range rather than the entire log file. The table below shows the index structure utilizing both line numbers and time stamps. Each IP address is assigned in a hash table, where the hash bucket stores the tuples representing a paired value: (line number, time stamp). The index is constructed using Trie tree [13], which is commonly used for indexing in many fields such as Nature Language Processing (NLP). In NLP, for example, Trie tee is employed to store and search words efficiently, where words in the same branch share common prefix but split

when different letters occurs. The Figure below shows a Trie tree structure for indexing requests.
It can be observed that user navigation path also implies a parent-child (page) manner. However, it is different with word indexing because a child page may have multiple parent pages where physical links exist. In addition, users can click back to any of its ancestor pages or accessing the client side-cached pages during a session. To solve this issue, we define each tree node in a Tire tree as a (r, t) tuple, where r denotes the row number of the requests in a log file and t denotes the time stamp of the request's arrival. Rather than using linked list to represent a Trie tree in word indexing, we use nested hash tables to index log requests, where the first level hash keys are the IPs, and the hash buckets stores the line number and the time stamp, the line number in the hash bucket also links to other line numbers. The Figure below shows the nested hash table structure storing the session information.

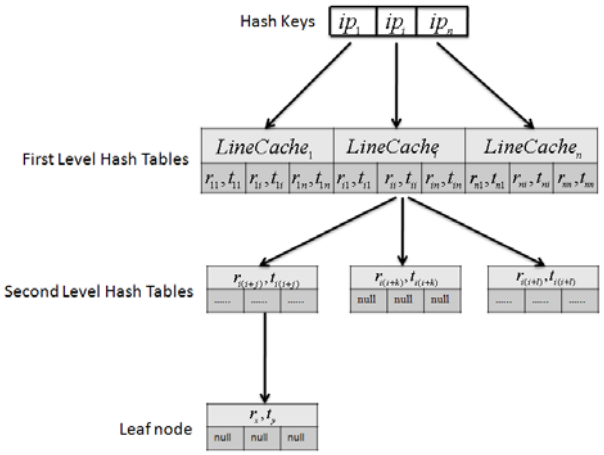

Fig. 6 A Trie Tree Structure for Indexing User Navigation Path

Based on the proposed Referrer Link method, the source-oriented user navigation path are retrieved and then modeled as above Trie structure. Same level tree nodes represent the requests directed from the same parent page (URL), where the time stamp can be used to identify the order of the requests on the same level. The construction of the Trie is described in detail as follows:

Construct Trie

Step1.) Hash IPs (this is for fast counting of IPs in a log file)
Step2.) For each IP, cache lines in a buffer (each IP corresponds to a bunch of requests)
Step3.) Find all entries matching the source (This is because that a user may come in the site multiple times via same path source)

Step4.) For each entry, in the succeeding lines of the line containing the source, iteratively check the referrer with their previous line to see whether the referrer matches the previously requested pages. If they match, insert the line number and time stamp in the tree after the previous line node. If they do not match, iteratively check the previously requested page with the referrer.
Step5.) Continue steps 2 until all IPs are finished.

The pseudo code of the proposed indexing algorithm is presented as follows:

```
Input: A Cleaned Log File and Pre-defined Path Sources
Output: A Trie tree indexing the Log requests
Method:
1. T= Empty Trie                                  /*Initialize a Trie Tree*/
2. for each Pre-defined Path Source:
3.     UniqueIP.hash={}                           /*Initialize a Hash Table*/
4.     for each line in log:
5.         UniqueIP.hash{ip:(LineNumber,TimeStamp)}   /*hash unique IPs with corresponding line number and time stamp*/
6.     T.Add(Path Source)
7.     for each IP in UniqueIP.hash:              /*caching corresponding lines for referrer linking*/
8.         for LineNumber in sorted UniqueIP.hash[IP]:
9.             LineCache.add(LineNumber)
10.    for Line in LineCache:
11.        if Line.referrer==Path Source:         /* find all entries matching the path source*/
12.            T.insert(Line.request); Next_referrer=Line.request;
13.            StartLine=FirstLine+1 ;            /*from the next line do forward referrer linking*/
14.            for NextLine in range(StartLine, Len(LineCache)):
15.                LineNo+=1;
16.                if NextLine.referrer==Next_referrer:
17.                    T.insert(NextLine.request); Next_referrer=NextLine.request;
18.                else:                          /*from the previous line do backward referrer linking*/
19.                    for PreviousLine in range(LineNo, StartLine):
20.                        LineNo-=1;
21.                        if PreviousLine.request==NextLine.referrer:
22.                            T.insert(PreviousLine.referrer); Next_referrer=NextLine.request;
23.                            Break ;            /*stop when the first backward matching detected */
24. Return T
```

Fig. 7 Pseudo code of log indexing algorithm

Experimental Results

The web logs we used for the experiments were obtained from three city-sites (Shanghai, Beijing and Shenzhen) of www.fangyou.com whose main business scope is to serve the real estate information of 60 major cities in China to both agents and customers. For the purpose of privacy and commercial information protection, we have encoded all explicit user accesses. The duration of the log is from 1st October 2009 to 31st December 2009. The table below shows the statistic of the web log used in our experiment. In order to normalize the features of these three workloads, we take the average of the four figures: log size, request number, session length and unique IP. We also define 25 different sources as the origins of user navigation paths to be extracted in the experiment.

Table. 4 Basic log statistics of three city-sites

Sites	Log size	Request number	Session length	Unique IP	Path Source Defined
Shanghai	1.35GB	10,400,000	4.8	98,425	25
Beijing	890MB	6,074,000	5.7	62,317	25

Shenzhen	760MB	5,020,144	4.1	41,213	25

In Table.4, log size denotes the average size of the file over 60-day duration while the request number is the average number of requests received by the servers per day. Session length denotes the average number of pages viewed within a user session. We implemented the proposed session identification algorithm Referrer Link applied to both transaction identification algorithms: Referrer Length and Maximal Forward Reference. The evaluation between the two algorithms, Referrer Link and Timeout, is conducted by performing both transaction identification algorithms on the sessions identified by the proposed Referrer Link and Timeout. Then, we conducted association rule mining against transactions generated from the session files produced by both Referrer Link algorithm and Timeout method. Two interestingness measures presented by Huang et atl. [24] were considered for the purpose of evaluating the proposed Referrer Link algorithm in our experiments.

1. Confidence. The confidence of a rule or pattern is denoted as $P(B \mid A)$. For association rules, $P(B \mid A)$ denotes the probability that event B occurs in a session conditioned on the occurrence of event A. With this measure, rules are ranked according to their confidence value as the main key and their support value as the secondary key.

2. Mutual information (MI). In probability theory and information theory, the mutual information of two random variables is a quantity that measures the mutual dependence of the two variables. The mutual information between events A and B is defined as $MI(A,B) = \log_2 \frac{P(AB)}{P(A)P(B)}$. Informally, mutual information compares the probability of observing A and B together (the joint probability) with the probabilities of observing A and B independently. [24] After the rules are ranked according to an interestingness measure, we then evaluate the interestingness of the top-ranking rules by interpreting the domain experts from www.fangyou.com who is capable to determine whether the rules are interesting or not. In this way, for each ranked list of discovered association rules generated by an interestingness measure from a session file, we can calculate the rule precision which is the percentage of the interesting rules yielded in the top 10, top 20 or top 30 rules. Table.5 shows the results of two identification algorithms in terms of rule precision in top 10, 20 and 30 ranked rules discovered based on Confidence interesting measure. It can be seen that Referrer Link-based transactions give higher rule precision than Timeout method under both transaction identification algorithms. It can increase rule precision of both transaction identification algorithms by 10% in average compared to Timeout. Interestingly, M.F.R has lower rule precision with both Timeout and Referrer Link. The first reason is that some backward references may be ignored once they are already cached on client side. Therefore, the transactions generated by M.F.R may suffer accuracy problem. However, applying the proposed Referrer Link algorithm can compensate the loss of cached pages so that the mining results between M.F.R and Reference Length do not pose significant difference.

Table. 5 Rule Precision of Confidence interestingness measure

Approach	Parameters	Shanghai/ Rule precision			Beijing/ Rule precision			Shenzhen/ Rule precision		
		Top 10 rules	Top 20 rules	Top 30 rules	Top 10 rules	Top 20 rules	Top 30 rules	Top 10 rules	Top 20 rules	Top 30 rules
Timeout-based M.F.R	20 min	50%	45%	50%	60%	63%	67%	60%	65%	60%
	25 min	60%	77%	67%	65%	70%	73%	70%	87%	80%
	30 min	80%	80%	77%	80%	83%	83%	80%	75%	77%
Referrer Link-based M.F.R	n/a	90%	95%	93%	90%	99%	97%	100%	99%	93%
Timeout-based Reference Length	20 min	60%	65%	85%	70%	75%	85%	60%	85%	7%
	25 min	70%	80%	85%	80%	80%	85%	80%	80%	90%
	30 min	80%	85%	80%	80%	75%	70%	80%	75%	93%
Referrer Link-based Reference Length	n/a	100%	95%	90%	90%	85%	90%	100%	95%	97%

Another interestingness measure Mutual Information (MI) was borrowed to rank the rules discovered. The results of two different session identification algorithms are shown in Table.6. It has shown that the Referrer Link algorithm is capable to produce high quality sessions for further mining procedure, especially for M.F.R algorithm. The difference of mining results between M.F.R and Reference Length is smoothed due to the supplemented sessions given by Referrer Link. Xiangji et al. [25] and Cooley et al. [11] all reported in their work respectively that Reference Length performs better than M.F.R in general. However, the advantage of M.F.R is that it does not require any assumed parameters representing data set characteristics. Thus, utilizing M.F.R on the session files generated by Referrer Link can simplify the mining procedure and reduce the total cost.

Table. 6 Rule Precision of Mutual Information interestingness measure

Approach	Parameters	Shanghai/ Rule precision			Beijing/ Rule precision			Shenzhen/ Rule precision		
		Top 10 rules	Top 20 rules	Top 30 rules	Top 10 rules	Top 20 rules	Top 30 rules	Top 10 rules	Top 20 rules	Top 30 rules
Timeout-based M.F.R	20 min	30%	20%	20%	30%	25%	33%	40%	30%	30%
	25 min	40%	35%	47%	50%	55%	43%	60%	47%	50%
	30 min	40%	45%	50%	60%	60%	47%	60%	75%	63%
Referrer Link-based M.F.R	n/a	100%	85%	83%	90%	85%	93%	80%	85%	77%
Timeout-based Reference Length	20 min	70%	75%	83%	60%	75%	77%	70%	75%	83%
	25 min	90%	85%	77%	70%	85%	83%	80%	70%	77%
	30 min	90%	80%	83%	80%	75%	80%	85%	80%	83%
Referrer Link-based Reference Length	n/a	90%	85%	95%	100%	95%	93%	90%	85%	93%

Conclusion and Future Works

In this paper, we proposed and discussed a novel session identification algorithm Referrer Link along with a web log indexing algorithm to serve fast source-oriented user navigation path retrieval. By employing Referrer Link and the Trie-based log indexing structure, the extracted sessions can better reflect user navigation behaviors such as frequent page-bounce and avoid client-side cache effect. In our future work, we will combine the cookie mechanism along with the source-oriented paths to define user sessions more precisely. In addition, the further experiments will be also conducted between the proposed indexing approach and other database engines. Although modern database engines all have indexing mechanisms, we believe that the normal database engines are not appropriate for log storage and preprocessing since the size of log files sometimes can grow unpredictably large and the Structured Query Language (SQL) is not satisfied to deal with complicated preprocessing jobs in most circumstances. Hence, the proposed indexing algorithm can be employed to speedup log preprocessing, especially navigation path extraction, where log files are very large.

Acknowledgement

This project was supported by the National Natural Science Foundation of China (70971020) and the Natural Science Foundation of Shanghai (06ZR14004).

Reference

[1] Wenke Lee and Salvatore J. Stolfo, Data Mining Approaches for Intrusion Detection, in the 7th USENIX Security Symposium, 1998.

[2] S Mukkamala, GI Janoski, AH Sung, Intrusion detection using support vector machines, in Proceedings of the High Performance Computing Symposium-HPC, pp.178-183, 2002.

[3] C Kruegel, G Vigna, Anomaly detection of web-based attacks, in Proceedings of the 10th ACM conference on Computer and Communications security, pp.251-256, 2003.

[4] Qiang Yang, Haining Henry Zhang, Tianyi Li, Mining web logs for prediction models in WWW caching and prefetching, In the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD'01, August 26 - 29, 2001.

[5] F. Bonchi, F. Giannotti, C. Gozzi, G. Manco, M. Nanni, D. Pedreschi, C. Renso and S. Ruggier, Web log data warehousing and mining for intelligent web caching, Data & Knowledge Engineering Volume 39, Issue 2, pp.165-189, November 2001.

[6] Robert Cooley, Mukund Deshpande, Pang-Ning Tan, Jaideep Srivastava, Web usage mining: discovery and applications of usage patterns from Web data, ACM SIGKDD Explorations Newsletter Volume 1, Issue 2 pp.12-23,2000.

[7] Jian Pei, Jiawei Han, Behzad Mortazavi-asl, Hua Zhu, Mining Access Patterns Efficiently from Web Logs, Knowledge Discovery and Data Mining Current Issues and New Applications, Volume 1805/2000, pp. 396-407,2000.

[8] M.S. Chen, J.S. Park, and P.S. Yu, Data mining for path traversal patterns in a Web environment, In Proceedings of the 16th International Conference on Distributed Computing Systems, pages 385-392, 1996.
[9] Magdalini Eirinaki, Michalis Vazirgiannis, Web mining for web personalization, ACM Transactions on Internet Technology (TOIT), Volume 3, Issue 1, pp. 1 – 27, 2003.
[10] Weinan Wang and Osmar R. Zaïane, Clustering Web Sessions by Sequence Alignment, In Proceedings of the 13th international workshop on database and expert systems applications, pp. 394-398, 2002.
[11] Cooley, R., Mobasher, B. and Srivastava. J, Data Preparation for Mining World Wide Web Browsing Patterns, Knowledge and Information Systems, Volume 1 (1), pp.5-32,1999.
[12] Mehrdad Jalali, Norwati Mustapha, Ali Mamat, Md. Nasir B Sulaiman, A new clustering approach based on graph partitioning for navigation patterns mining, in Proceedings of 19th International Conference on Pattern Recognition, 2008.
[13] Donald Knuth, The Art of Computer Programming, Volume 3: Sorting and Searching, Second Edition. Addison-Wesley, 1997. ISBN 0-201-89685-0. Section 6.3: Digital Searching, page 492.
[14] Jinlin Chen, An UpDown Directed Acyclic Graph Approach for Sequential Pattern Mining, international, IEEE Transactions on Knowledge and Data Engineering, pp.913-927, Vol. 22, no. 7, July 2010.
[15] Ma, Justin, Saul, Lawrence K., Savage, Stefan, Voelker, Geoffrey M., Identifying Suspicious URLs: An Application of Large-Scale Online Learning, In Proceedings of the 26th International Conference on Machine Learning, Montreal, 2009.

Chenhan Liao received his PhD from school of engineering Cranfield University in 2009. He is now a post doctoral in Glory Sun management school Dong Hua University. His research interests include data mining, data warehousing and information retrieval.

Jianguo Zheng is a professor in Glory Sun management school Dong Hua University. His research interests are Evolutionary algorithms, data mining, artificial intelligence and information management.