Journal of Computer Science 2 (9): 704 - 709, 2006
ISSN 1549-3636

MFTPM: Maximum Frequent Traversal Pattern Mining with Bidirectional Constraints

Jiadong Ren, Xiaojian Zhang and Huili Peng
College of Information Science and Engineering, YanShan University, Qinhuangdao 066004, China

Abstract: An important application of sequential mining technique is maximal frequent traversal pattern mining, since users' traversal pattern and motivation are latent in session sequence at some time segment. In this paper, a Frequent Traversal Pattern Tree structure with dwell time (FTP-Tree) is designed to store, compress the session database, and simplify the configuration of dwell time thresholds during mining. A novel algorithm based on bidirectional constraints, called Maximal Frequent Traversal Pattern Mining (MFTPM) is presented, which traverses quickly FTP-Tree and discovers maximal frequent traversal patterns from the session sequences. Experimental results show that MFTPM can significantly reduce the average execution time and the storage space for mining maximal frequent traversal patterns. Our performance study shows that MFTPM performs muth better than previous approaches in the time constraint environment.

Key words: Dwell time ; FTP-Tree ; Session ; Maximal frequent traversal pattern

INTRODUCTION

Mining Frequent Traversal Patterns (FTP) is an important task in web usage mining. Web usage mining makes sense of data generated by observing web surf sessions or behaviors and finds the relationship among different users' accesses. Sessions, users' behaviors, and traversal data on each web server can be extracted from the web logs. Analysis of traversal sequence data can obtain trends of users' interests and provide useful information for server performance enhancements, restructuring a web site, and directly marketing in e-commerce. Since web contains mass usage data, it is indubitable that we can obtain abundant FTP. However, some FTP are not only numerous but also meaningless. Many traditional algorithms generally mine the candidate traversal sequences to get FTP. If a traversal sequence of length L is a FTP, then 2^{L-1} candidate subsequences must be enumerated one by one. If the length of sequence is too long, we have to spend considerable execution time and waste vast disk space. In addition, autonomic mining FTP in web environment is unrealistic. So, we needn't mine FTP but Maximal Frequent Traversal Patterns (MFTP), since the most information of FTP is contained in MFTP. In [1], J.Han introduces that data mining is an interactive process, and decision-makers should directly take part in the process through query language or GUI for mining MFTP. Thus we design a method to settle those problems, which decision-makers can give some time parameters to constrain every page in each traversal sequence of sessions. Although the method possibly limits the frequency of some pages, we can discover more interesting MFTP.

There exist many algorithms for mining FTP and MFTP, such as GSP[2], MSPS and SPADE etc. But these algorithms mostly aim at the whole database. SPADE[3] are often fit for small databases. If the database is too large and the minimal support is very low, SPADE will generate large numbers of candidate sequences, which are too big to be loaded into memory. GSP is an efficient algorithm for mining large database. However, the length of the longest frequent sequences determines the number of scanning database it requires. Consequently, if there exist very long frequent sequences and if the database grows huge, the I/O cost of GSP could be very large. Although MSPS[4] based sampling technique can reduce much more search space and based pruning technique can remove many candidate subsequences, it possibly loses lots of useful information of traversal sequences, and only gets approximate result. In addition, the rate of sample directly affects the precision of mining MFTP. In this case, we propose a new MFTPM algorithm for mining MFTP.

The rest of the paper is organized as follows: The definitions of dwell time, session FTP, and MFTP are described in the section 2. In the Section 3, we construct the FTP-Tree. MFTPM, performance evaluation and experimental results are described in the section 4 and the section 5 . Finally, Section 6 draws a conclusion researched and describes the future work.

PROBLEM DEFINITION

Corresponding Author : Jia-Dong Ren, College of Information Science and Engineering, Yanshan University, China

Let $P = \{P_1, P_2,\ldots, P_n\}$ be the complete set of web pages. Let DB be the traversal sequence database to be mined, DB is a set of sessions, $DB= \{S_1, S_2,\ldots, S_m\}$ and $S_i = <(P_1, t_1)(P_2, t_2)\ldots(P_r, t_r)>$ where $S_i \subset DB$, $P_j\in S_i$, $1\leq i\leq m$, $1\leq j\leq r$, t_j is the time of requesting P_j. Each record in DB includes session identifier (Sid), traversal sequence and timestamp. We specify $t_1 = 0$ in order to compute dwell time conveniently, where t_1 is the requesting time of each entry page in every session. Consider two sequences $s_a=<a_1,a_2,\ldots,a_n>$ and $s_b=<b_1,b_2,\ldots,b_m>(n\leq m)$. If there exists integers $1\leq i_1<i_2<\ldots\leq m$ with $a_1=b_{i1}$, $a_2=b_{i2},\ldots,a_n=b_{im}$, then s_a is a subsequence of s_b, and s_b is a super-sequence of s_a sequences. s_a is called a prefix of s_b if and only if (1) $b_i=a_i$ for $1\leq i\leq n\leq m$; (2) $a_n\subseteq b_m$; and (3) all pages in $(b_m$-$a_n)$ are alphabetically ordered after those in a_n. Given a session database DB formed by traversal sequences, the support count of a traversal sequence s_i is denoted by Count (s_i), where Count (s_i) denotes the number of session in DB that contains s_i. The length of a traversal sequence s_i is the number of pages in the sequence. A traversal sequence of length k is called a k-traversal sequence.

[Definition] Dwell time is the actual time that a user spends on a content page in a sequence of session.

Let P_i, P_{i+1} be two adjacent pages in a sequence of session. T_i is the time of request of P_i, T_{i+1} is the time of request of P_{i+1}. Suppose T_3 is the time of loading P_i, T_4 is the time of loading the ancillary files, and T_0 is the dwell time of P_i. According to Fig. 1, $T_3=T_1$-T_i, $T_4=T_2$-T_1.

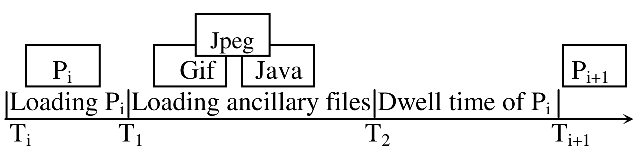

Fig. 1: Dwell time of P_i

The dwell time of P_i can be calculated by finding the difference between the requests of P_i and P_{i+1}, and subtracting the time required loading P_i and the ancillary files from the value (using equation (1) to calculate T_0). But the time required to loading streaming media files like real audio and mpeg may not be considered for the dwell time computation of P_i.

$$T_0=T_{i+1} -T_i -[(T_1 -T_i)+(T_2 -T_1)] \text{ or } T_{i+1} -T_i -(T_3+T_4) \qquad (1)$$

$$T_0=T_{i+1} -T_i - (T_1- T_i) \text{ or } T_{i+1} - T_i -T_3 \qquad (2)$$

This paper will take equation (2) to compute T_0. In the case, the time of loading ancillary files (T_4) doesn't be considered. Decision-makers can give two time thresholds according to their purpose. The two thresholds are minimal dwell time λ_1 and maximal dwell time λ_2. We can utilize λ_1 and λ_2 to constrain dwell time T_0 of every page, and remove the unreasonable pages from the sequence s.

Let P_i be a page in traversal sequence s_i, $P_i.t_0$ be its dwell time of P_i. *FTPset* denotes a complete set, which consists of the frequent traversal patterns.

(1) If $P_i.t_0<\lambda_1$, then $P_i\notin s_i$, $P_i\notin FTSset$ ($s_i \subseteq DB$). This case indicates the content of P_i doesn't satisfy the want of the user, or the page is error.

(2) If $P_i.t_0>\lambda_2$, then $P_i\notin s_i$, $P_i\notin FTSset$ ($s_i \subseteq DB$). This case indicates the user may exit from the web site, or he revisits the P_i that bad been saved in the cache of browsers. We know P_i in cache doesn't leave behind any information in web logs.

As we know, there are many time granularities used to describe dwell time and timestamp, such as hour, minute and second and so on. We choose minute as the unit of dwell time, this is because hour is too coarse and second too detailed for mining MFTP, which was introduced in [5].

[Definition] A session is a page sequence ordered by timestamp in usage data record, or is a visit performed by a user from the time when he enters the web sites to the time he leaves.

Before mining FTP and MFTP, different sessions for the same user should be reconstructed. During the reconstruction, two time constraints are very crucial. One is that the duration for any session can't exceed a defined threshold. The most commonly used timeout threshold is *30min*, which was proposed in [6]. The other is that the time gap between any two continuously visited pages can't exceed another defined threshold. The most commonly used threshold is *10min*, which was presented in [7].

[Definition] Traversal sequence s_i is a Frequent Traversal Pattern (FTP) if and only if Count $(s_i)\geq Min\_sup$ and $\lambda_1\leq s_i.T_0\leq 30min$ ($\lambda_2=30min$), where *Min_sup* is a user specified support threshold, *30min* is upper limit time threshold of a session sequence. A FTP of length k is called a k-FTP.

[Definition] Given a traversal sequence set V, s_i is a Maximal Frequent Traversal Pattern (MFTP) if and only if $\nexists s_i'$ s.t. $(s_i'\in V)\wedge(s_i'\subseteq s_i)\wedge(s_i'\neq s_i)$, where s_i' is an assumed sequence.

[Property] If s_i is a MFTP, and s_i is not contained in any other traversal sequence in V. We use MFTPS to represent the set of all maximal frequent traversal sequences in V.

CONSTRUCTION OF TREE STRUCTURE

In this section, a new in-memory data structure called FTP-Tree is constructed. FTP-Tree is a tree structure, which must satisfy three necessary conditions. First, the tree consists of one root labeled as "*null*", a set of sequence-prefix subtree as the children of the root, and header table. Second, each node in FTP-Tree includes four fields: *P.name*, *Node.count*, *Node.link*, and *Session:T_i*. *Session:T_i* registers which sessions will contain the same page with dwell time, where T_i denotes some dwell time of the page. Hence we can easily make use of an equation to express the relation between *Session:*T_i and *Node.count*, represented by *Node.count*= $\sum$ *Page*.(*Session:*T_i). Third, each entry of the header table includes three fields: *Page*, *Page.count*, and *Node-link*, where *Node-link* indicates the pointer pointing to the first node in FTP-Tree, and the node carrying the same name with the Page. The following algorithm explains the steps of constructing FTP-Tree.

Algorithm 1: construct FTP-Tree (T, s)
Input: Traversal Sequences $s_1, s_2, \ldots, s_m, \lambda_1, \lambda_2, T_0$
Output: FTP-Tree// Store and compress database
Function FTP-Tree (T, s)
(1) While ($P_j \neq null$) and ($\lambda_1 \leq P_j.T_0 \leq \lambda_2$) Do {//$P_j \in s_i$, $s_i \subset \{s_1, s_2, \ldots, s_m\}$
(2) If ($P_j.name = A.name$) Then{//A is the ancestor of T
(3) $A.count = A.count + 1$
(4) $A.count = \sum A.(Session{:}T_i) + Session{:}P_j.T_0$
(5) $T = A$
(6) Else If($P_j.name = C.name$)Then{//C is the child of T
(7) $C.count = C.count + 1$
(8) $C.count = \sum C.(Session{:}T_i) + Session{:}P_j.T_0$
(9) $T = C$
(10) Else
(11) Insert (T, P_j)
(12) $P_j.count = 1$; $P_j = P_j.next$}
(13)end if
(14)end if

Given a database of session *DB*, which consists of traversal sequences of users, as Table1 shows. The notation Sid represents the identifier of a session. We first convert the database *DB* with timestamp constraint into *TDB* with dwell time constraint. *TDB* is shown in Table2. According to Algorithm1, we are able to use FTP-Tree to store and compress *TDB*. Let the time of loading every page in traversal sequence be T_3=0.5min, the two thresholds of dwell time be λ_1=1.5min, λ_2=10min (Decision-makers are free to determine the values of λ_1 and λ_2 according to his purpose for mining MFTP). Thus we adopt equation (2) to compute the dwell time (T_0) of every page. Through scanning *TDB*, we can insert the accessed pages of traversal sequences into FTP-Tree as its nodes. But the inserted pages must be constrained by λ_1 and λ_2, that is, the pages in FTP-Tree must cater for the two thresholds. For example, though page P_8 has appeared in s_2, its dwell time T_0 (0.5min) is less than λ_1 (1.5min), therefore P_8 can't link to P_3. Fig. 2 shows the FTP-Tree of *TDB*.

Table 1: *DB* with timestamp constraint

Sid	Traversal Sequence	Timestamp
S_1	P_1 P_2 P_3 P_4 P_2 P_5	0,7,16,21,24.5,27,30
S_2	P_1 P_2 P_3 P_4 P_3 P_2 P_5 P_7 P_5 P_3 P_8	0,3,5,9,12,15,17,21,24,26.5 ,29,30
S_3	P_1 P_2 P_1 P_3 P_9	0,5.5,13,17.5,26,30
S_4	P_3 P_9 P_7 P_9 P_9 P_4	0,4,9.5,12,17.5,23.5,30

Table 2: *TDB* with dwell time constraint

Sid	Traversal Sequence	Dwell time
S_1	P_1 P_2 P_3 P_4 P_2 P_5	6.5,8.5,4.5,3,2,2.5
S_2	P_1 P_2 P_3 P_4 P_3 P_2 P_5 P_7 P_5 P_3 P_8	2.5,1.5,3.5,2.5,2.5,1.5 3.5,2.5,1.5,2, 0.5
S_3	P_1 P_2 P_1 P_3 P_9	5,7,4,8,3.5
S_4	P_3 P_9 P_7 P_9 P_9 P_4	3.5,5,2,5,5.5,6

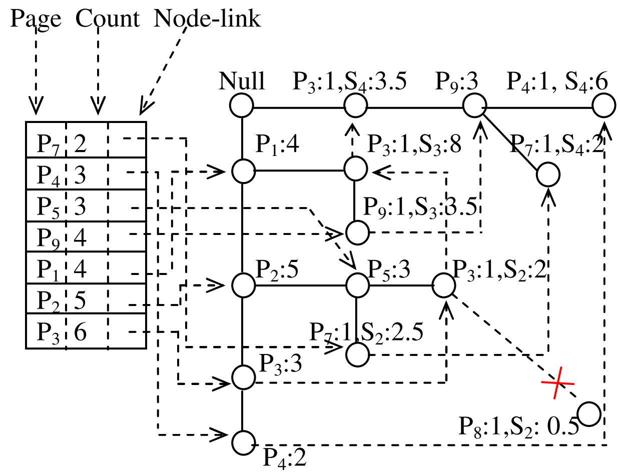

Fig. 2 : FTP-Tree of *TDB*

THE MFTPM ALGORITHM

We apply the following strategies to mine MFTP from FTP-Tree. At first, according to λ_1, λ_2, *30min*, and *Min_sup* specified by user, function MFTPM in Algorithm2 can generate every *1*-FTP, denoted by α_i, as initial suffix, the next constructs its prefix traversal sequence base B, and finally builds longer traversal sequences by every traversal sequence base connecting with its suffix α_i. For example, if β is a traversal sequence base in B, the sequence $\beta \cup \alpha_i$ will be longer traversal sequence with α_i suffix. If the longer sequence $\beta \cup \alpha_i$ can simultaneously satisfy the above parameters, then $\beta \cup \alpha_i$ becomes a new FTP. The function executes the procedure until all *1*-FTP have been done. At last, we discover MFTP in FTP with new judging conditions. The following algorithm shows the steps for MFTP.

Algorithm 2: MFTPM
Input: FTP-Tree T, λ_1, λ_2, T_0, *Min_sup*
Output: MFTPS //Maximal Frequent Traversal Pattern Set
Initialization: MFTPS= $\varnothing$
Function MFTPM (T, α, MFTPS) {
(1) If T only contains a single path then
(2) Then {
(3) Generate MFTP $\beta \cup \alpha$
(4) If (($\beta \cup \alpha$) $\geq$ *Min_sup*)) Then {
(5) If (($\beta \cup \alpha$) $\not\subset$ MFTPS) and (($\beta \cup \alpha$) $\neq$ *Sseq*)//*Sseq* is not a subsequence of any other frequent traversal sequence in MFTPS
(6) Then MFTPS= MFTPS $\cup$ ($\beta \cup \alpha$)
(7) Else Discard $\beta \cup \alpha$}
}

(8) Else
(9) For each α_i ($\alpha_i.count \geq Min\_sup$) and ($\lambda_1 \leq \alpha_i.T_0 \leq \lambda_2$) Do {//$\alpha_i$ is a page in the head table
(10) Generate *1*-FTP α_i
(11) $S=S \cup \alpha_i$, $Q=\alpha_i.next$ //Q points to the first location of α_i in the FTP-Tree
(12) While ($Q \neq null$) and ($Q.count \geq Min\_sup$) and ($\lambda_1 \leq Q.T_0 \leq \lambda_2$) Do {//$Q.T_0$ shows the dwell time
(13) Generate α_i's prefix traversal sequence base β, and construct long traversal sequence β_i //$\beta_i=\beta \cup \alpha_i$
(14) If ($\beta_i.count \geq Min\_sup$) and ($\lambda_1 \leq \beta_i.T_0 \leq 30min$) Then {//$\lambda_1$ is minimal dwell time in long sequence β_i. $\lambda_1=Min(\beta_i.T_0)$
(15) $S=S \cup \beta_i$, $Q=Q.next$}// End While
(16) If ($S \geq Min\_sup$) Then {
(17) If ($S \not\subset$ MFTPS) and ($S \neq Sseq$)
(18) Then MFTPS= MFTPS $\cup$ S
(19) Else Discard S}
(20) Exit for loop body}

ANALYSIS AND PERFORMANCE EVALUATION

The analysis of MFTPM algorithm is similar to FP-growth[8]. First, given an FTP-Tree T, λ_1, λ_2, and parameter T_0, we mine the MFTP from T with traversal strategy. If T only contains a single path of FTP-Tree in which each node only has a single child, then we can directly get MFTP $\beta \cup \alpha$. Utilizing (4)(5)(6)(7) lines judge whether $\beta \cup \alpha$ merge into MFTPS or not. When T is multipath FTP-Tree, each α_i of catering for (9) line should generate *1*-FTP and construct prefix traversal sequence β for each α_i. The long FTP $\beta \cup \alpha_i$ can be formed with α_i suffix and β prefix in steps (12)(13)(14) and (15). The next steps (16)(17) and (18) of MFTPM are to determine the set of all maximal frequent traversal patterns from the FTP-Tree constructed so far. Let's examine the efficiency of the algorithm by mining *TDB* on condition that three parameters are fulfilled Min_sup=0.5, λ_1=1.5min, λ_2=10min, the result of MFTPS={$P_1\,P_2\,P_3\,P_4$, $P_1\,P_2\,P_5$, P_9}.

Table 3: Execution time of MFTPM on BMS-WebVies-1

SR	NB	FTP1	MFTP1	ET1	FTP2	MFTP2	ET2	PT
BMS-WebVies-1	1000	625	23	68s	594	19	33s	51.47%
	2000	1134	45	125s	1002	27	59s	52.80%
	4000	1728	21	170s	1365	14	80s	52.94%
	8000	2861	65	314s	2798	57	143s	54.46%
	16000	3106	54	485s	2983	48	207s	57.32%

Table 4: Execution time of MFTPM on BMS-WebVies-2

SR	NB	FTP1	MFTP1	ET1	FTP2	MFTP2	ET2	PT
BMS-WebVies-2	1000	542	26	75s	518	23	37s	50.67%
	2000	987	32	114s	860	27	53s	53.51%
	4000	1853	29	185s	1675	18	83s	55.14%
	8000	2741	56	301s	2683	42	132s	56.15%
	16000	3219	43	532s	3014	39	220s	58.65%

All the experiments are performed on a 2.4GHz Pentium 4 processor with 512 megabytes main memory, running on Microsoft Windows 2000. In addition, all the programs are written in Microsoft/Visual C++ 6.0. We pursue the experiments on real datasets to evaluate the performance of MFTPM algorithm. The real datasets, BMS-WebVies-1 and BMS-WebVies-2, which contain several months worth of click sequence data from two e-commerce web sites. The two datasets was provided by Blue Martini Software[9], and is available from the KDD Cup 2000 home page. Collecting the same number sequence data of the two datasets, which are divided into the different length sessions, and the average session contains 5-11 pages.

Table 5: Notation for analysis

SR	The source of session
NB	The number of session
FTP1	The number of FTP without λ_1, λ_2
MFTP1	The number of MFTP without λ_1, λ_2
FTP2	The number of FTP with λ_1, λ_2
MFTP2	The number of MFTP with λ_1, λ_2
ET1	Execution time without λ_1, λ_2
ET2	Execution time with λ_1, λ_2
PT	The improved performance in percentage

To test the performance of MFTPM algorithm, two experiments are performed.

The first group experiment: Algorithm2 traverses FTP-Tree from two different aspects. One aspect, the algorithm doesn't involve λ_1, λ_2, the other contains λ_1, λ_2, the results given in Table 3 and Table 4. The two tables list that the numbers of frequent traversal Patterns (FTP1, FTP2) and maximal frequent traversal patterns (MFTP1 MFTP2) have little changed with λ_1, λ_2, but great changes of the execution time have taken place obviously. PT (PT=1- $\frac{ET2}{ET1}$) changed from 51.47% to 57.32% in Table 3, from 50.67% to 58.65% in Table 4. As the size of NB goes up, the performance of MFTPM is clearly predominant. Compared to ET1, ET2 in the two tables shortened over two times at low support (Min_sup=0.5) and appropriate dwell time constraints (λ_1=1.5min, λ_2=10min, T_0=0.5min). For example, in

Table5, when NB size grows 16000, execution time changes from 532s to 220s.

The second group experiment: For measuring the performance of MFTPM furthermore, utilizing the above datasets, we compare our algorithm with SPADE, MSPS and GSP at the different values of *Min_sup*. In Fig. 3 and Fig. 4, we plot total execution time taken by MFTPM algorithm and the others for values of minimum support threshold *Min_sup* ranging from 0.2% to 1.2%. The Figures show how decreasing *Min_sup* leads to an increase in execution time. As seen from the result shown in Fig. 3 and Fig. 4, the run time of MFTPM is distinctly faster (about 2-3 times faster) than SPADE, MSPS and GSP when the support threshold goes down.

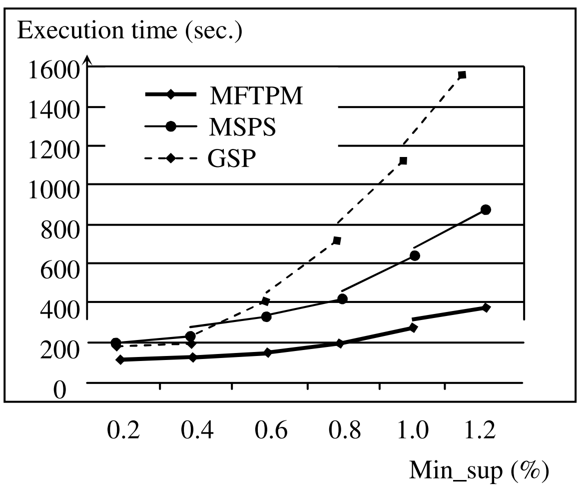

Fig. 3: Execution time of three algorithms on BMS-WebVies-1

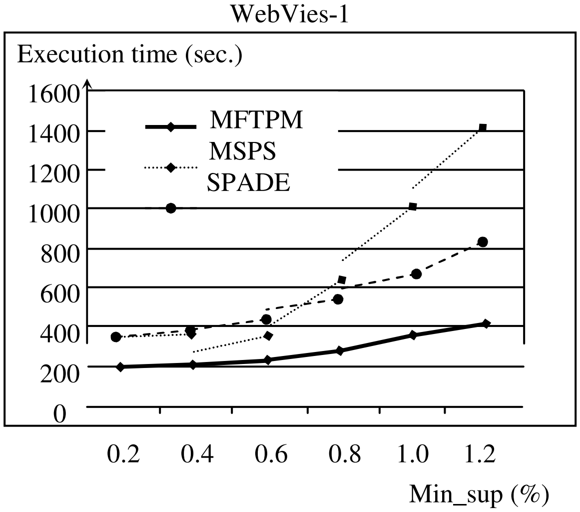

Fig. 4: Execution time of three algorithms on BMS-WebVies-2

CONCLUSION

In this paper we proposed a new algorithm MFTPM that mines the set of all MFTP over the traversal sequences, and an in-memory data structure called FTP-Tree is constructed for storing FTP. In the MFTPM algorithm, a bidirectional dwell time technique is used to constrain every page in session sequences, which efficiently limits the number of the meaningless pages and FTP. Experiments with the decision-makers giving the proper constraint parameters show that MFTPM can effectively reduce the execution time and I/O cost. We also performed extensive experiments comparing with GSP, MSPS and SPADE, and the result showed that the performance of MFTPM outperformed the others. MFTPM algorithm was only examined in the static dataset. The next step, we will undertake further analysis and experiments in dynamic web click stream.

REFERENCES

1. Jiawei Han and Micheline Kamber, 2001. Data Mining: Concepts and Techniques, Morgan Kanfmann, Chinese.
2. Srikant, R. and Agrawal, R., 1996. Mining sequential pattern: Generalizations and performance improvements, In Proceeding of the 5th Conference on Extending Database Technology (EDBT), Avignion, France, pp. 3-17.
3. Zaki, M, J., 2001. SPADE: An efficient algorithm for mining frequent sequences, Machine Learning, Vol 42, No. 1, pp. 31-60.
4. C.Luo and SM Chung, 2004. A Scalable Algorithm for Mining Maximal Frequent Sequences Using Sampling, In Proceeding of the 16th IEEE Int'1 Conf. On Tools with Artificial Intelligence (ICTAI), Florida, USA, pp. 156-165.
5. Xiaolong Zhang, Wenjuan Gong and Yoshihiro Kawamura, 2004. Customer behavior pattern discovering with web mining, In Proceeding of the 6th Asia-Pacific Web Conf (APWeb), Hangzhou, China, pp. 844-853.

6. L.D. Catledge, and J.E. Pitkow, 1998. Characterizing browsing strategies in the World-Wide Web, Computer Networks and ISDN Systems, Vol 27 No. 6, pp. 1065-1073.
7. Long Wang, and Christoph Meinel, 2004. Behavior recovery and complicated pattern definition in web usage mining, In Proceedings of the 2004 IEEE International Conference on Multimedia and Expo (ICME), Taipei, Taiwan, pp. 531-543.
8. J. Han, J. Pei and Y.W. Yin, 2000. Mining frequent patterns without candidate generations, In Proceedings of the ACM. SIGMOD International Conference on Management of Data, Dallas, USA, pp. 1-12.
9. Z. Zheng, J. Shen and L. Mason, 2001. Real world performance of association rule algorithms, In Proceeding of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, California, USA, pp. 401-406.