A Top-Down Algorithm for Mining Maximal Traversal Paths in Web Log Sessions

M. Thilagu
VLB Janakiammal College of Engineering and Technology
India

R. Nadarajan
PSG College of Technology
India

R. Jeevitha
VLB Janakiammal College of Engineering and Technology
India

ABSTRACT
Mining of frequent traversal paths in web logs is an application of sequence mining and useful with many applications that include web recommendation, caching, pre-fetching etc. Most of the existing algorithms follow a bottom-up approach to mine sequence patterns in a database. In this paper, a fast top-down algorithm is presented to discover maximal traversal paths which are contiguous sequences in web log session sequences. The algorithm avoids candidate sequence generation and searches only maximal potential patterns in the minimized search space during mining process. Experimental results show that the proposed algorithm can perform better than an existing approach.

Keywords
Sequence Database, Contiguous Sequence, Maximal Potential Pattern, Maximal Traversal Path.

1. INTRODUCTION
The expansion of the World Wide Web (WWW) has resulted in a large amount of data and retrieving the relevant information or hidden information from them is a challenging task. Web mining is the application of data mining techniques to extract knowledge from Web data, i.e. Web Content, Web Structure and Web Usage data. Web Mining can be broadly divided into three distinct categories, according to the kinds of data to be mined and they are 1)Web Content Mining is the process of extracting useful information from the contents of Web documents 2) Web Structure Mining is the process of discovering the structure of hyperlinks in the Web and 3) Web usage mining is the task of discovering the activities of the users while they are browsing and navigating through the Web.

Web Usage Mining, also known as web log mining, aims to discover interesting and frequent user access patterns from web browsing data that are stored in web server logs, proxy server logs or browser logs. Web usage mining consists of three phases, namely preprocessing, pattern discovery, and pattern analysis. Preprocessing converts the original usage data into data abstractions necessary for pattern discovery. The main pre-processing tasks include data cleaning, user identification, session identification and transaction identification. Pattern discovery adopts the various data mining techniques for discovering access patterns from the preprocessed usage data. The main techniques include statistical analysis, association rule mining, clustering, classification and sequential pattern mining. Once the access patterns have been identified, they require to be analyzed to determine how that information can be used. Pattern analysis filters out uninteresting patterns from the set found during the pattern discovery phase. Web access patterns mined from web logs can be used for many practical applications include personalization, system improvement through pre-fetching, catching, site modification, e-business intelligence etc[Nakagawa M. and Mobasher B. (2003)].Discovering access pattern is the problem of finding association rules in the web logs.

Mining frequent access patterns (called sequential access pattern mining) in a sequence database was firstly introduced by Srikant and Agrawal [Agrawal R. and Srikant R. (1995)]. Among the sequential pattern algorithms, the representative ones are GSP, FreeSpan, SPADE and PrefixSpan [Agrawal R. and Srikant R (1996), Han J. et al. (2001), Zaki M. (2001), Pei J. et al (2004)] and they first detect short patterns and then gradually grow pattern length, which may be inefficient when long sequences exist. These algorithms have been studied on customer purchase behaviour sequences, however variants of these could be applied on web log sequences since a web access pattern is like a sequential pattern. The apriori based web access pattern techniques full-scan (FS) and selective-scan (SS) proposed by [Chen et al. (1998)] are used to mine traversal paths with maximal forward references and an algorithm was also developed for mining frequent traversal sequences with backward references [Show-Jane Yen (2003)]. Pei et al. (2000) proposed an algorithm, called WAP mine, for finding access patterns from Web logs. WAP-mine constructs a WAP-tree and mines the sequences without candidate generation. However, the size of the structure for the WAP-tree is very large for long sequences and also needs to create a large number of links for linking to the next items. A Contiguous Item Sequential Patterns (CISP) using a compact representation called UpDown Tree was proposed to mine contiguous frequent patterns in web logs. The algorithm generates prefix and suffix trees for each item by identifying the sequences containing that item and item based patterns are mined [Chen J. (2008)].

One of the interesting problems in web mining is mining of web navigation patterns or traversal paths that are hidden in web log sessions. Such discovered knowledge is useful in analyzing how the web pages are accessed or what are seeking for by the users. A user web access sequence is a sequence of web pages ordered by increasing traversal-time. In web log sequences, contiguous sequence patterns may appear as subsequences or subsets and mining of these sequences is important since link between web

pages is important and to be maintained. Mining of all subsequences would require a lot of time and tedious in sequence mining and hence finding only maximal frequent patterns out of them are potential and less time consuming. With this motive, a top-down algorithm has been proposed to mine maximal traversal paths which are frequent contiguous sequences in web log sessions. The rest of the paper is organized as follows: Section 2 defines the problem statement. Section 3 describes the proposed algorithm to mine maximal traversal paths with an example and Section 4 exhibits the performance evaluation with experimental results. Finally, conclusion is presented with summary of our study.

2. PROBLEM STATEMENT

Given a web log database and the min-sup threshold, the problem defined here is to mine all maximal traversal paths or contiguous sequence patterns in the database. The constraint in the problem is that sequences are with single occurrence of events without repetition, since the input used in this problem is a web log database with sessions having first occurrence of pages. That is**,** S = <e_1,e_2, …,e_n>is a n-sequence of traversal path, where e_i is an event and $D = \{T_1, T_2, \ldots, T_n\}$ is a weblog database, where T_i represents a web access sequence in a session. We give out the definitions of some basic terms below.

Definition 1 (Contiguous Sequence Pattern) : A sequence $a = <a_1,a_2,\ldots,a_n>$ is a **contiguous subsequence** of another sequence $b = <b_1, b_2, \ldots, b_m>$, $m \geq n$, if there exists an integer i, $1 \leq i \leq m-n+1$ and $a_1 \subseteq b_i$, $a_2 \subseteq b_{i+1},\ldots,a_n \subseteq b_{i+n-1}$. A contiguous sequence pattern is a frequent pattern in which the items appearing must be adjacent with respect to the ordering of items as defined in the pattern and its support is greater than or equal to min-sup.

Definition 2 (Maximal Contiguous Sequence Pattern): A contiguous sequence pattern is said to be maximal if it is not a subsequence of any other frequent pattern.

3. A TOP-DOWN APPROACH TO MINE MAXIMAL TRAVERSAL PATHS

This section describes the proposed algorithm for mining maximal frequent sequences or patterns in web log sessions. The algorithm applies a top-down approach to discover the maximal traversal paths which are contiguous sequence patterns in web logs. The proposed algorithm mines maximal patterns fast in two phases. That is, it first generates the maximal potential patterns by merging frequent contiguous patterns detected in the divided search space. It then mines only those generated patterns in the minimized search space, to discover frequent patterns among them. The merits of the proposed algorithm are identified as follows: 1) No level-wise candidate generation 2) Reduced number of database scans by minimizing the search space. We explain the algorithm with an illustration as follows.

Consider a web log sequence database with session sequences as shown in Table 1 and min-sup as 2. In this case, the maximal frequent contiguous sequence prefixed with 'P1' satisfying the given min-sup is (P1P2P5P6).

Table 1: Sequence Database

Session-id	Page Sequence
1	P1 P2 P5 P6 P4
2	P2 P6 P5 P4
3	P3 P1 P2 P7 P5 P6 P4
4	P3 P1 P2 P6 P7 P4
5	P3 P1 P2 P5 P6 P7

For the given database, the algorithm works as follows: By scanning the database twice, the algorithm first discovers frequent 1-itemset and frequent contiguous 2-itemset. For each frequent contiguous 2-itemset as a prefix, the database is scanned to find all frequent contiguous patterns with 2-itemset from the prefix based sequences and a vertical table is created with frequent 1-itemset and their transaction-ids for the divided search space. From the above table, frequent contiguous patterns with 2-itemset found are {(P2P5),(P5P6),(P6P7)} from the sequences prefixed with {(P1P2)}. Then, these patterns are merged to generate a maximal potential pattern (P1P2P5P6P7) using the vertical table. At the time of pattern generation, a prefix pattern search table is created with first two columns namely prefix pattern and transaction-id list as shown in Table 2 whereas the third column (minimized search space) is a derived one. During mining process, the minimized search space is derived by performing the difference operation on the search space (ie. transaction id-lists) of prefix patterns MPP_k and MPP_{k-1} of a maximal potential pattern MPP. Hence, the generated patterns are searched in the minimized search space in order to improve the mining process by reducing the number of scans.

Table 2: Search Space of Pattern (P1P2P5P6P7)

Prefix Pattern	Transaction id-list	Minimized Search Space ($T_k - T_{k-1}$)
P1P2	{1,3,4,5}	{4}
P1P2P5	{1,3,5}	-
P1P2P5P6	{1,3,5}	{1}
P1P2P5P6P7	{3,5}	{3,5}

During the mining process, the initial search space of the maximal potential pattern (P1P2P5P6P7) ie MPP_k is found as transaction-id list {3,5} using Table 2. Then, these transactions are scanned to check whether MPP_k exists and find the support count of its suffix sequence. Since its support count < min-sup, the pattern becomes infrequent. Then, the support count of prefix pattern MPP_{k-1} (P1P2P5P6) is checked and it also does not satisfy the min-sup in the searchced space. At the same time, the minimized search space of MPP_{k-1} is found by performing difference operation on the transaction id-lists of MPP_k and MPP_{k-1} that is {1}. During next iteration, the derived search space {1} is scanned to find the support count of suffix sequence of MPP_k (ie.P1P2P5P6) of MPP. Since the transaction-id {1} has MPP_k as subset, its support count is incremented. Now the searched pattern satisfies the min-sup. Hence, the pattern (P1P2P5P6) is a maximal frequent pattern and the mining process gets terminated at this point. Similarly, other patterns are mined in the minimized search space at each level and the above process is recursively done until

the pattern found to be frequent or the length of the pattern k becomes 2, since we start from frequent 2-itemset as prefix. The search space is pruned by removing infrequent prefix patterns after mining a maximal potential pattern. The frequent patterns are retained and used to mine maximal potential patterns with common prefixes generated in subsequence processes.

Algorithm

Input : Sequence Database D, min-sup

Output : Maximal Frequent Patterns

Initially, read the database D and find all the frequent 1-itemset and frequent contiguous 2-itemset. Repeat the steps in Phase-I and Phase-II for each frequent contiguous 2-itemset α, to mine all maximal frequent patterns in the database. During Phase-I, pruning process is done to check whether the generated pattern already exists or not.

Phase-I : Generating Prefix Based Maximal Potential Patterns

// MPP - Maximal Potential Pattern

Procedure MPPGen()

Step 1: Scan the sequences prefixed with α twice to find frequent 1-itemset and frequent contiguous 2-itemset FC_s

Step 2: Create a vertical table with frequent 1-itemset & their transaction id-list found in the divided search space

Step 3: Repeat

Generate a maximal potential pattern MPP_i by

merging patterns in FC_s ($p_i \bowtie p_j$, where $i \neq j$)

satisfying the min-sup using their search space in

the vertical table

Call MPPMine()

Until no pattern can be generated

Endproc

Phase-II : Mining of Maximal Frequent Patterns

Procedure MPPMine()

Step 1: Initialize k = Length of MPP, MPP_k = MPP, $T_s = T_k$ (Transaction id-list of MPP_k)

// Prefix Pattern Search Table is used to find the search space of MPP

// k indicates the length of the pattern

Step 2: Repeat

Call PPMine with MPP_k, T_s

If support count of MPP_k >=min-sup

MPP_k is frequent

Return

Else

If support count of MPP_{k-1} >=min-sup

MPP_{k-1} is frequent

Return

Endif

Endif

$MPP_k = MPP_{k-1}$

k=k-1

Until k=2

Endproc

Procedure PPMine()

Step 1: Scan the sequences in T_s and increment the support count

of suffix sequence of MPP_k if found

// Derive the minimized search space

Step 2: If $T_k \# T_{k-1}$

$T_s = T_k - T_{k-1}$

Endif

Endproc

4. PERFORMANCE EVALUATION

In this section, we report our experimental results on the performance of our work in comparison with PrefixSpan, an efficient SP mining algorithm among several algorithms. The performance parameter used here to evaluate the algorithms is the execution time. As proposed in [[Pei J. et al (2004)], PrefixSpan algorithm scans the database at least once for each sequential pattern. It requires 'n' number of scans for a pattern with length n and repeats the scanning process in the projected database. Our algorithm overcomes this problem by generating maximal potential patterns to avoid level by level candidate generation and scanning only those generated sequences. Hence, it requires only five scans for a sequential pattern.

The machine used for performance evaluation was an IBM compatible personal computer with 2GHz Pentium microprocessor, 1GB RAM and the Windows XP operating system. The dataset used for performance evaluation is the CTI Dataset containing the preprocessed and filtered sessionized data for the main DePaul CTI Web server (http://www.cs.depaul.edu). The CTI dataset is a small dataset with 682 items which are scattered in the dataset. The original (unfiltered) data contained a total of 20950 sessions from 5446 users. Here, the file used for our experiment is the cti.tra file which contains the filtered sessionized data in transaction format. Each line in this file corresponds to the sequence of pages visited during one session. The performance evaluation of the algorithms is evaluated based on their time complexity and it is shown as a comparative study

on the execution time of both the algorithms for the min-sup=0.01 in Table 3. Figure 1 shows the performance analysis of these two algorithms on CTI dataset and a graph is drawn by using the execution time details of both the algorithms given in Table 3. That is, the analysis is performed by varying the data sizes with the support count as 0.01. From the performance analysis report, it is found that the proposed algorithm performs better than PrefixSpan.

Table 3: A Comparative Study On Execution Time

Data Size	Proposed Algorithm (Minutes)	PrefixSpan Algorithm (Minutes)
110K	0.9	1.6
120K	1.3	3.4
130K	1.8	5.8
140K	2.4	7.3

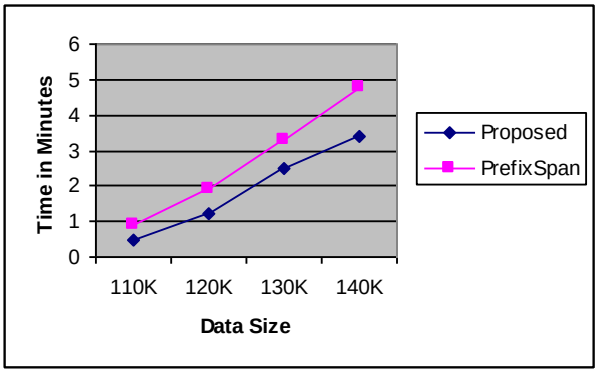

Figure 1: Performance analysis for min-sup=0.01 varying data sizes

Before mining is performed on cti.tra file, preprocessing is done to translate pageviews into page ids using cti.cod file in order to make the mining process easier. From the experimental results, it is found that there are seventeen contiguous sequence patterns in cti dataset with a total datasize of 320KB, satisfying the min-sup=10%. A list of maximal contiguous sequence patterns discovered are represented in terms of page ids with their support count in Table 4 and for some sample page ids, their corresponding page views are given in Table 5.

Table 4: Maximal Traversal Paths

Traversal Path/ Contiguous Sequence Pattern	Support Count
387,415,556	673
387,415,557	520
387,59,71	1292
387,59,72,71	256
387,59,661	145
387,54,358	794
387,316,286	233
54,358,357	155
54,358,324	244
54,358,326,328	160
393,54,358	148
24,45,327	130
45,327,328	183
316,286,309	170
316,286,266	193
0,349,340	153
20,349,340	169

Table 5: Sample Page Views with IDs

Page id	Pageview
387	/news/default.asp
59	/courses/
72	/courses/syllabisearch.asp
71	/courses/syllabilist.asp
54	/authenticate/login.asp?Section=mycti&title=mycti &url ahead=studentprofile/studentprofile
358	/cti/studentprofile/studentprofile.asp?Section=mycti
324	/cti/core/core.asp

5. CONCLUSION

In this paper, a fast top-down method has been explored for mining maximal traversal paths which are contiguous sequence patterns in a web log database. To enable fast mining, maximal potential patterns are generated and mining is performed only for those patterns with minimized search space. The proposed algorithm could improve the mining performance significantly in order to mine maximal patterns by avoiding level-wise candidate generation and reducing number of scans in the database. Experimental results and evaluations performed on real data demonstrate that the proposed algorithm outperforms an existing algorithm.

REFERENCES

[1] Agrawal R. and Srikant R. (1995) Mining Sequential Patterns. In Proceedings ICDE'95, 3-14.

[2] Antunes C. and Oliveira A. L. (2003) Generalization of Pattern-

[3] Growth Methods for Sequential Pattern Mining with Gap Constraints. In Int'l Conf Machine Learning and Data Mining, 239-251.

[4] Antunes C. and Oliveira A. L. (2004) Sequential pattern mining algorithms: Trade-offs between speed and memory. In 2nd Workshop on Mining Graphs, Trees and Seq, Italy.

[5] Ayres J., Gehrke J., Yu T., and Flannick J. (2002) Sequential PAttern Mining using a Bitmap Representation. In SIGKDD 429-435.

[6] Chen, M.S., Park, J.S. & Yu, P.S. (1998). "Efficient Data Mining for Path Traversal Patterns." In IEEE Transactions on Knowledge and Data Engineering , 209-220.

[7] Chen J. (2008) Contiguous Item Sequential Pattern Mining Using UpDown Tree, Intelligent Data Analysis – An International Journal, Vol. 12, No. 1, pp. 25-49.

[8] Chen J, Cook T. (2007) Using d-gap Patterns for Index Compression. In WWW, 1209-1210.

[9] Dhany Saputra, Dayang R. A. Rambli, Oi Mean Foong, (2008) Mining Sequential Patterns Using I-PrefixSpan, International Journal of Computer Science and Engineering 2;2.

[10] Han J., Pei J.,Mortazavi-Asl B., Chen Q., Dayal U.,and Hsu M.-C. (2001) FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining. In Proc. ACM SIGKDD 355-359.

[11] Lin M. and Lee S. (2005) Fast Discovery of Sequential Patterns through Memory Indexing and Database Partitioning. J. Info. Sci. and Eng., 21, 109-128.

[12] Nakagawa M. and Mobasher B. (2003) A Hybrid Web Personalization Model Based on Site Connectivity. In WEBKDD 59-70.

[13] Pei, J., Han, J., Mortazavi-asi, B. and Zhu, H. (2000). Mining Access Patterns Efficiently from Web Logs. In Proceedings of 6th Pacific Area Conference on Knowledge Discovery and Data Mining (PAKDD), 396-407.

[14] Pei J., Han J., Mortazavi-Asl B., Wang J., Pinto H., Chen Q., Dayal U. and Hsu M. C. (2004) Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. IEEE TKDE, vol. 16, 1424-1440.

[15] Show-Jane Yen "An Efficient Approach for Analyzing User Behaviors in a Web-Based Training Environment" Journal of Distance Education Technologies, 1(4), 55-71, Oct-Dec 2003.

[16] Srikant R., Agrawal R., (1996) Mining Sequential Patterns: Generalizations and Performance Improvements. In Int'l Conf Extd. DB. Tech. 3-17.

[17] Zaki M. (2001) SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Lrng., 40, 31-60.

[18] Zhang Z. and Kitsuregawa M. (2005) "LAPIN-SPAM: An Improved Algorithm for Mining Sequential Pattern," Proc. of Int'l Special Workshop on Databases For Next Generation Researchers, pp. 8-11