Web Access Pattern Mining – A Survey

A. Rajimol[1] and G. Raju[2]

[1] School of Computer Science, Mahatma Gandhi University, Kottayam, Kerala, India
[2] Department of IT, Kannur University , Kannur , Kerala, India
kurupgraju@rediffmail.com

Abstract. This article provides a survey of different Web Access Pattern Tree (WAP-tree) based methods for Web Access Pattern Mining. Web Access Pattern Mining mines complete set of patterns that satisfy the given support threshold from a given Web Access Sequence Database. A brief discussion of basic theory and terminologies related to web access pattern mining are Presented. A comparison of the different methods is also given.

Keywords: Web Access Pattern Mining, Pre-order Linked Web Access Pattern Mining, First-Occurrence Linked Pattern Tree mining, First-Occurrence Forest Mining, Conditional Sequence Mining.

1 Introduction

Web mining is the extraction of interesting and useful knowledge and implicit information from activity related to the WWW. The web data is typically unlabelled, distributed, heterogeneous, semi-structured, time varying and high dimensional [2].

Web usage mining, also known as Web log mining, discover interesting and frequent user access patterns from the web browsing details stored in server web logs, proxy server logs or browser logs [1]. Web log mining has become very critical for effective web site management, creating adaptive Web sites, business and support services, personalization and so on [2].

Sequential pattern mining is an important data mining tool used for web log mining. Sequential pattern mining that discover frequent pattern in a Web Access Sequence Data Base (WASD) was first introduced by Agarwal and Srikant in [3]: Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user-specified minimum support threshold, sequential pattern mining is to find all frequent subsequences. A web log is a sequence of pairs: user-id and access information. For the purpose of study of sequential pattern mining, preprocessing [6] is applied to the original log file and WASD is generated.

Let E be a set of events. A Web Access Sequence (WAS), $S = e_1e_2 .....e_n$ for ($1 \leq i \leq n$), is a sequence of events where events can be repeated; n is the length of access sequence. $S' = e_1'e_2'.....e_n'$ is subsequence of Access Sequence (AS) $S = e_1e_2 .....e_n$ and S is a super sequence of S', if and only if $1 \leq i_1 \leq i_2 \leq .. \leq i_n \leq n$ such that $E'_j = Ej$ for $1 \leq j \leq n$ [7]. In $S = e_1e_2 ....e_ke_{k+1}....e_n$, if subsequence $S_{suffix} = e_{k+1} ...e_n$ is a super sequence of pattern $P = e'_1e'_2 ...e'_l$, and $e_{k+1} = e'_1$, the subsequence of S, S prefix $= e_1e_2...e_k$, is called the prefix of S with respect to pattern P [1]. Support of a pattern S in WASD is the number of sequences S_i in WASD, which contain the subsequence P, divided by the number of transactions in the database. An access sequence S is said to be an access pattern of WASD, if support of S in WAS, *sup*WAS(S), is greater than or equal to the given threshold. Given a WASD and a support threshold ξ, Web Access Pattern Mining mines the complete set of ξ patterns of WAS [7]. In the succeeding sections, we review the WAP-tree structure and some selected WAP-tree based mining algorithms and in the last section we give a comparison of their performance.

2 WAP-Mine Algorithm

Pei et al. in [7] proposed a compressed data structure known as Web Access Pattern Tree (WAP-tree) and WAP-Mine (Web Access Pattern tree Mining) algorithm for mining web access patterns efficiently from web logs. WAP-Mine, the recursive mining algorithm used to generate access patterns from the WAP-tree, is based on the suffix heuristic: if a is a frequent event in the set of prefixes of sequences in WAS with respect to pattern P, then sequence aP is an access pattern of WAS. WAP-tree registers all access sequences and corresponding count very compactly and maintains linkages for traversing prefixes with respect to the same suffix pattern.

2.1 Construction of WAP-Tree and Mining of WAP-Tree

Only frequent 1-sequences are considered for constructing WAP-tree, as they only are useful in generating k-sequences where k>1. Common prefixes are shared in the WAP-tree to save space. WAP-tree registers two pieces of information, label and count. The root of the tree is a special virtual node with an empty label and count 0.

The construction of WAP-tree is done as follows: for each access sequence, frequent subsequence is entered into the tree, starting from the root node. While inserting the first event e, if the current node has already a child e, then the count of the child node with event e is incremented by 1, otherwise a new child with label e and count 1 is created. Then recursively insert the rest of the subsequence to the sub tree rooted at the current e node. All nodes with same label e are linked by shared label linkages into a queue, called event-queue. Event queue for e_i is called e_i-queue. Head of each event-queue is registered in a header table H.

A simple web access sequence database obtained after preprocessing, with the set of access events $E = \{a, b, c, d, e, f\}$ is shown in Table 1 [1] and WAP-tree with linkage header for the WASD is given in Fig. 1.

Table 1. A database of web access sequences

User ID	web access sequence	frequent sub- sequence
100	abdac	abac
200	eaebcac	abcac
300	babfaec	babac
400	afbacfc	abacc

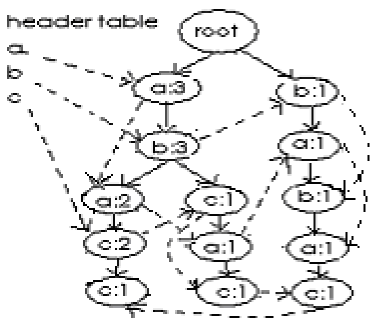

Fig. 1. WAP-tree with linkage (*dotted line*) for the frequent sub-sequences in Table 1

All pattern information related to a frequent event e_i can be accessed by following all the branches in WAP-tree linked by e_i-queue only once. All nodes in the path from root of the tree to node e_i (excluded) form a prefix sequence of e_i and the count of this node is the count of the prefix sequence. Let G and H be two prefix sequences of e_i and G is also formed by the sub-path from root that H is formed by, H is called a super-prefix sequence of G, and G is a sub-prefix sequence of H. For a prefix sequence of e_i without any super-prefix sequences, the un-subsumed count is the count of it. For a prefix sequence of e_i with some super-prefix sequences, the un-subsumed count is the count of that sequence minus un-subsumed counts of all its super-prefix sequences. Access patterns with same suffix are now used for searching all web access patterns.

Mining is done as follows: For each event e_i in the header list, conditional sequence base is found. The conditional sequence list of a suffix event is got by following the header link of the event and reading the path from the root to each node (excluding the node). The count of the prefix sequence is same as the count of the node. For each prefix sequence inserted into the conditional sequence base with count c, all its sub-prefix sequences are inserted with $-c$ as count, to get the un-subsumed count. Then the set of conditional frequent events is found. If it is not empty, recursive mining is done on the conditional WAP-tree of each frequent event. When there is only one branch in the conditional WAP-tree, all unique combinations of nodes in that branch are generated as patterns.

3 PLWAP-Mine Algorithm

Y. Lu, and C. I. Ezeife propose Pre-Order Linked WAP-Tree Mining (PLWAP) and a method of assigning binary position code to tree nodes in [8]. PLWAP algorithm based on WAP-tree avoids recursive re-construction of intermediate WAP-trees by using binary position codes to the nodes of the tree to determine the suffix trees of any frequent pattern prefix under consideration by comparison.

The position code is assigned to the nodes on the binary tree equivalent of the tree using the Huffman coding idea. Given a WAP-tree with some nodes, the position code of each node is assigned following the rule that the root has null position code, and the leftmost child of the root has a code of 1, but the code of any other node is derived by appending 1 to the position code of its parent, if this node is the leftmost child, or appending 10 to the position code of the parent if this node is the second leftmost child, the third leftmost child has 100 appended, etc. A node α is an ancestor of another node β if and only if the position code of α with "1" appended to its end, equals the first x number of bits in the position code of β, where x is the number of bits in the position code of $\alpha + 1$.

Building of PLWAP-tree is done as follows: Root node is created with position code NULL and count 0. Tree and header table is generated from access sequences as in the creation of WAP-tree [7]. The Constructed tree is traversed in pre-order and each node e_i is attached to the corresponding e_i-queue and head of each queue is registered in corresponding header table entry. Major difference in the construction of a PLWAP-tree and a WAP-tree is in the generation of binary code for nodes along with tree creation and in the generation of the pre-order linkages.

PLWAP-Mine starts by finding the frequent 1-sequence $\{a, b, c\}$. Then for every frequent event and the suffix trees of current conditional PLWAP-tree being mined, it finds the first occurrence of this frequent event in every suffix tree being mined by following the pre-order linkage of this event, and adds the support count of all first occurrences of this frequent event. If the count is greater than the minimum support threshold, then this event is concatenated to the previous list of frequent sequence, F. Now, the suffix trees of these first occurrence events in the previously mined conditional suffix PLWAP-trees are used for mining the next event.

4 CS-Mine

In [9] B.Y. Zhou et al. propose a method CS-Mine (Conditional Sequence Mining) based on WAP-tree structure [7]. CS-Mine involves Constructing Initial Conditional Sequence Base (CSB), Constructing Event Queues (EQ) for CSB, Single Sequence Testing for CSB, Constructing Sub-CSB, and Recursive Mining for Sub-CSB.

The conditional sequence base of event e_i based on suffix sequence S_{suffix}, is the set of all long prefix sequences of e_i in sequences of a database. If Ssuffix is empty, the conditional sequence base is the frequent sub-sequence set of the given database. Otherwise, it is the conditional sequence base CSB(S_{suffix}).

CS-mine starts by constructing the initial CSB by mining WAP-tree. For each frequent event e_i in the given WAP-tree, all the prefix sequences of all the nodes in the e_i-queue form the Initial CSB of e_i, Init-CSB(e_i). The count of related suffix node in the e_i-queue is the count of each prefix sequence. To avoid duplicate counting, for each prefix sequence of e_i with count n, all of its sub-prefix sequences are also inserted into Init-CSB(e_i) with minus count, -n. To construct event queues for each CSB(S_c) (S_c=e_i for Init-CSB), conditional frequent events, events in CSB(S_c) that satisfy minimum support are found first. Header table is created with all the conditional frequent events. Then, for each conditional frequent event e_i, a linked-list structure called e_i–queue, connecting the last item labelled ei in the sequences of CSB (S_c), is created. The head pointer of each event queue is recorded in the Header Table. Finally, all non-frequent events in sequences in CSB (S_c) are discarded.

CSB(e_i+S_{suffix}) is called the sub-conditional sequence base of CSB(S_{suffix}), if e_i is not null. The Construct-SubCSB algorithm is used for constructing CSB(e_i+S_c) based on CSB(S_c) for each event e_i in the Header Table of CSB(S_c). Single Sequence Test checks whether all sequences in CSB(S_c) can be combined into a single sequence. If so, the mining of CSB(S_c) will be stopped. Otherwise, Sub-CSB for each event in the Header Table based on CSB(S_c) is found and recursive mining is done.

5 FLWAP-Mine

Peiyi Tang et al. introduce First-occurrence Linked WAP-tree and present a pattern mining algorithm FLWAP-Mine based on that in [10]. They also present the theory of conditional searching on which their pattern-growth mining algorithms are based.

Given a web access sequence S and a symbol a, from the set of symbols $\sum$ such that a is in S, the a-prefix of S is the prefix of S from the first symbol to the first occurrence of a inclusive. The a-projection of S is what is left after the a-prefix is deleted. Given the database D and a symbol a in $\sum$, the a-projection database D_a of D is the multi-set of a-projections of the web access sequences in D that support a. D_a is used to grow frequent patterns by using symbols from $\sum$. It is sufficient to use the sub-trees rooted at the children of the first-occurrences of a, to represent projection database D_a, ignoring possible empty sequences. Since first-occurrences play a central role in finding the support of a symbol and its projection database, all pattern-growth mining algorithms based on aggregate trees [10] try to find them efficiently.

Given a symbol a, from $\sum$ and database D, the support of pattern p in the a-projection database D_a of D is equal to the support of pattern $a.p$ in the original database D. And support of empty pattern is $|D_a|$. That is,

$$\text{Sup}_D(a) = |D_a| . \tag{1}$$

Let $F(D, \eta)$ be the set of frequent patterns in D with respect to η and $F'(a.D\ \eta)$ be the set of non-empty frequent patterns in D that start with symbol a, then $F'(a.D ,\eta)$ is equal to a. $F (Da ,\eta)$. Thus

$$\begin{aligned} F(D,\eta) &= \emptyset && \text{if } |D| < \eta \quad \text{and} \\ F(D,\eta) &= \{\varepsilon\} \cup \text{U } a \in \textstyle\sum a\, F(Da, \eta) && \text{if } |D| \geq \eta \end{aligned} \tag{2}$$

The equations (1) and (2) are the base of the FLWAP-Mining algorithm.

5.1 Mining with FLWAP-Tree

A node is a first-occurrence, if none of its ancestors has the same label. The count of first-occurrence of a node with label *a*, is the number of sequences in *D* that share the common *a*-prefix represented by the path from the root node to this. The sum of the counts of all the first-occurrences of a symbol, is the number of sequences in *D* that contain at least one occurrence of *a*, i.e. the support of *a* in *D*, Sup_D (*a*).

To build FLWAP-tree, sequences are entered into the tree as in the base WAP-tree. Then the first-Occurrences of each symbol are linked to form the First-occurrence Linked WAP-tree (FLWAP-tree). The first-occurrences can be found by the pre-order traversal of a portion of the base WAP-tree. Once the FLWAP-tree is constructed, it is mined to generate all patterns using the FLWAP-mine algorithm. The algorithm works as follows. For each frequent event a, follow the first-occurrence link to find its first-occurrences. If the sum of the counts of first-occurrences is greater than absolute threshold, *a* is concatenated to the previous pattern and added to the set of patterns. Now the set of the sub-trees rooted at the children of the first-occurrences is found, FLWAP-tree for Da is built and recursively mined.

Difference of FLWAP-Mine algorithm from PLWAP-Mine is in finding first-occurrences and in building intermediate tree for the projection database. To find the first-occurrences of a symbol, PLWAP attach a position code to each node and link all the nodes of the same label in a pre-order traversal of the tree whereas in FLWAP this is done by traversing a portion in pre-order and the algorithm is available in [11].

6 FOF-Mine

In [11], the authors present a modification to FLWAP mining method in [10]. Forest of First-Occurrence sub-trees (FOF structure) are the basic data structure for representing projection database instead of the concept of linked trees. Given a symbol *a*, each sub tree rooted at a first-occurrence of *a*, is the first-occurrence sub-tree of *a*. A list of pointers to the first-occurrences of *a* in the aggregate tree is the forest of first-occurrence sub-trees of a symbol.

In First-Occurrence Forest-Mine (FOF-Mine), the aggregate tree is extended to make the root node represent the empty symbol ε. The count of the root node is the total number of sequences in the database. The count of first-occurrences of a node with label *a*, is the number of sequences in that share the common a-prefix. The sub-trees rooted at the children of first-occurrences of symbol *a*, represent all the non-empty *a*-projections of the sequences sharing this common *a*-prefix.The sum of the counts of the root nodes of the first-occurrence sub-trees of *a* gives Sup_D (*a*). As all the nodes already exist in the original aggregate tree of the database to be mined, the memory cost of the forest of first-occurrences sub-trees is only the list of pointers.

Recursive FOF-Mine algorithm is used for mining FOF structure. The initial FOF structure based on aggregate tree is passed on to the mining function. The sub-trees rooted at the children of the first occurrence nodes represent the current database to be mined. The FOF structure for the projection database is established by repeatedly calling a function Find-First-Occurrences. In FOF-Mine, for each frequent event *a*, the projection database of the current database is build. If the sum of the counts of the root nodes of all the sub-trees is greater than the absolute threshold, *a* is appended to the previous pattern and added to the set of pattern. Now, the new database is recursively mined to generate all patterns starting with the current event.

7 Comparison of the Methods

WAP-Mine introduced in [7] is quite different from the apriori methods like Apriori, Apriori All, Apriori Some [4], GSP (Generalized Sequential Pattern mining) [5], PSP (Prefix Tree for Sequential Patterns) that are based on generate and test methods. WAP-Mine uses a very compact WAP-tree structure to store access sequences. Sharing of common prefixes by branches makes support counting and mining easier. Moreover, the conditional searching improves the efficiency. Experimental studies proved WAP-Mine to better than GSP. Both GSP and WAP-Mine exhibit linear scalability, but WAP-Mine outperforms GSP [7].

Recursive reconstruction of intermediate WAP-trees during mining in WAP-Mine, in order to compute frequent prefix subsequences of every suffix sequences is very time consuming. PLWAP-Mine [8] modify base WAP-tree to Pre Order Linked WAP-tree that link similar nodes in a pre-order fashion. PLWAP algorithm identifies the suffix trees or forest of any frequent pattern prefix under consideration by comparing the binary codes of nodes. Experiments show that PLWAP outperforms both GSP and WAP-Mine when the number of frequent patterns increases and the minimum support threshold is low. Performance of PLWAP degrades when the length of sequence is more than 20 because of the increase in the size of position codes as the depth of tree increases [12].

Use of the original PLWAP-tree for the entire mining forces it to go through the nodes that are not in the projection databases to find the first-occurrences though the memory requirement is reduced. The FLWAP-tree algorithm [10] links the first occurrences of each symbol. Since all the non-empty sequences in the a-projection database D_a of database D are included in the sub-trees rooted at the children of the first occurrences of a symbol a, the FLWAP-tree for D_a can be built from these sub-trees. Though the building of new trees for projection databases use more memory, reduction in the sizes of projection databases reduces the processing complexity. That is, FLWAP-tree mining is a trade-off memory for high performance. During experiments FLWAP-tree mining outperforms the PLWAP-tree mining consistently as the average length of sequence increases. The speed up of FLWAP also increases linearly as the average length of sequence increases [10].

The FLWAP-tree algorithm rebuilds every projection database, and thus, uses a lot of memory. Both the PLWAP-tree [8] and FLWAP-tree algorithms [10] are based on the concept of the linked tree. FOF algorithm [11] outperforms both PLWAP and FLWAP. Memory usage is very high for FLWAP as the latter creates intermediate projection trees. Due to the increase in size of the tree nodes in PLWAP, the FOF algorithm outperforms even the PLWAP-tree algorithm in memory usage, even though it does not create additional projection databases [11].

CS-Mine also is based on WAP-tree, but it uses WAP-tree only for generating conditional sequence base. Experimental results have shown that the CS-mine algorithm performs much more efficient than the WAP-mine algorithm, especially when the support threshold becomes small and the number of web access sequences gets larger [9].

8 Conclusion

Though the literature for Web Access Pattern Tree based methods includes a good number of papers, due to the space constraint only a set of important and basic methods of web access pattern mining are selected and presented in this review. Basic terminologies, theory and performance are highlighted. Finally a comparison of the different methods is given. With the exponential growth of WWW, there is imperative need for more efficient Web Access Pattern Mining algorithms.

References

1. Kosala, R., Blockeel, H.: Web Mining Research: A Survey. ACM SIGKDD Explorations 2, 1–15 (2000)
2. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.N.: Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. ACM SIGKDD Explorations 1, 12–23 (2000)
3. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: 20th International Conference on Very Large Databases, Santiago, Chile, pp. 487–499 (1994)
4. Agrawal, R., Srikant, R.: Mining Sequential Patterns. In: 11th International Conference on Data Engineering, Taipei, Taiwan, pp. 3–14 (1995)
5. Srikant, R., Agrawal, R.: Mining Sequential Patterns: Generalizations and Performance improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996)
6. Cooley, R., Mobasher, B., Srivastava, J.: Data Preparation for Mining World Wide Web Browsing Patterns. J. Knowledge and Information Systems 1, 5–32 (1999)
7. Pei, J., Han, J., Mortazavi-asl, B., Zhu, H.: Mining Access Patterns Efficiently from Web Logs. In: Terano, T., Chen, A.L.P. (eds.) PAKDD 2000. LNCS, vol. 1805, pp. 396–407. Springer, Heidelberg (2000)
8. Lu, Y., Ezeife, C.I.: Position Coded Pre-order Linked WAP-Tree for Web Log Sequential Pattern Mining. In: Whang, K.-Y., Jeon, J., Shim, K., Srivastava, J. (eds.) PAKDD 2003. LNCS (LNAI), vol. 2637, pp. 337–349. Springer, Heidelberg (2003)
9. Zhou, B.Y., Hui, S.C., Fong, A.C.M.: CS-Mine: An Efficient WAP-Tree Mining for Web Access Patterns. In: Yu, J.X., Lin, X., Lu, H., Zhang, Y. (eds.) APWeb 2004. LNCS, vol. 3007, pp. 523–532. Springer, Heidelberg (2004)
10. Tang, P., Turkia, M.P., Gallivan, K.A.: Mining web access patterns with first-occurrence linked WAP-trees. In: 16th International Conference on Software Engineering and Data Engineering (SEDE 2007), Las Vegas, USA, pp. 247–252 (2007)
11. Pearson, E.A., Tang, P.: Mining Frequent Sequential Patterns with First-Occurrence Forests. In: 46th ACM Southeastern Conference (ACMSE), Auburn, Alabama, pp. 34–39 (2008)
12. Lu, Y., Ezeife, C.I.: PLWAP sequential Mining: open source code. In: First International Workshop on Open Source Data Mining: Frequent Patterns Mining Implementation, Chicago, Illinois, pp. 26–35 (2005)