International Education Studies
www.ccsenet.org/journal.html
Vol. 1, No. 4
November 2008

Binary Coded Web Access Pattern Tree in Education Domain

C. Gomathi
P.G. Department of Computer Science
Kongu Arts and Science College
Erode-638-107, Tamil Nadu, India
E-mail: kc.gomathi@gmail.com

M. Moorthi
Department of Computer Science
Kongu Arts and Science College
Erode-638-107, Tamil Nadu, India
E-mail: moorthi_bm_ka@yahoo.com

K. Duraiswamy
K.S. R. College of Technology
Tiruchengode- 637-209, Tamil Nadu, India
E-mail: kduraiswamy@yahoo.co.in

Abstract

Web Access Pattern (WAP), which is the sequence of accesses pursued by users frequently, is a kind of interesting and useful knowledge in practice. Sequential Pattern mining is the process of applying data mining techniques to a sequential database for the purposes of discovering the correlation relationships that exist among an ordered list of events. WAP tree mining is a sequential pattern mining technique for web log access sequences, which first stores the original web access sequence database on a prefix tree. WAP-tree algorithm then, mines the frequent sequences from the WAP-tree by recursively re-constructing intermediate trees.

In this paper, we propose efficient sequential pattern techniques called **BC-WAP** (Binary Coded WAP). The proposed algorithm uses Kongu Arts and Science College web logs for sequential pattern mining. It eliminates recursively reconstructing intermediate WAP trees during the mining by assigning the **binary codes to each node in the WAP tree**. The results of the experiments show the efficiency of the improved algorithm.

Keywords: WAP tree, Data mining, Sequential pattern mining, Web log files

1. Introduction

Data Mining is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. With the wide spread use of databases and the explosive growth in their sizes, organization is faced with the problem of information overload. The problem of effectively utilizing these massive volumes of data is becoming a major problem for all enterprises. Traditionally, we have been using data for querying a reliable databases repository via some well-circumscribed application for canned report generating utility. Data mining attempts to source out patterns and trends in the data and infers rules from these patterns. With these rules the user will be able to support, review and examine decisions in some related business or scientific area. This opens up the possibility of a new way of interacting with databases and data warehouses. Sequential mining is the process of applying data mining techniques to a sequential database for the purposes of discovering the correlation relationships that exist among an ordered list of events which is the objective of this paper. The application of sequential pattern mining are in areas like Medical treatment, science & engineering processes, telephone calling patterns. Sequential pattern mining Web usage mining for automatic discovery of user access patterns from web servers.

It is used by education domain, this means detecting the behavior of the students, which pages seen many times and which is to improve etc.

The rest of this paper is organized as follows. In Section 2, we introduce sequential access pattern mining techniques and its related works. The proposed BC-WAP mine algorithm is then presented in Section 3. The experiment results are shown in Section 4. Finally, the conclusions of the paper are given in Section 5.

2. Sequential Access Pattern Mining

As an important branch of data mining, sequential pattern mining, which finds high-frequency patterns with respect to time or other patterns, was first introduced by (Agrawal R., and Srikant R.(1994)) as follows: given a sequence database where each sequence is a list of transactions ordered by transaction time and each transaction consists of a set of items, find all sequential patterns with a user specified minimum support, where the support is the number of data sequences that contain the pattern. Since the access patterns from web log take on obvious time sequence characteristic, it is natural to apply the technology of sequential pattern mining to web mining. According to the downward closure property of frequent sequences, to some extent, maximal frequent sequences have already included all frequent sequences. The space to store maximum frequent sequences is much lower than to store complete set, and web mining applications partly only depend on maximum frequent sequences rather than the complete set of frequent sequences so that mining maximum frequent access sequences is of essential practicability.

Sequential access pattern mining techniques are mainly based on two approaches: Apriori-based mining algorithms and WAP tree based mining algorithms.

2.1 Apriori-based Mining Algorithms

The AprioriAll (Pei J., Han J., Mortazavi-asl B., and Zhu H.(2000)) algorithm proposed a three-step approach for mining sequential patterns. It first finds all frequent itemsets. Then, it transforms the database such that each original transaction is replaced by the set of all frequent itemsets contained in the transaction. And finally, it finds the sequential patterns. However, this algorithm does not scale well due to the costly transformation step. In (Hafidh Ba-Omar, Iiias Petrounias and Fahad Anwar(ICALT 2007)), a generalized sequential pattern mining algorithm known as GSP mining algorithm was proposed. Similar to the AprioriAll algorithm, GSP scans the database several times. In the first scan, it finds all frequent items and forms a set of frequent sequences of length one. In subsequent scans, it generates candidate sequences from a set of frequent sequences obtained from the pervious scan and checks their supports. The process terminates when no candidate is found to be frequent. So this algorithm requires multiple scans of database. So now we discuss WAP tree based mining algorithm.

2.2 WAP Mining Algorithm

The WAP-tree is a very effective compressed data structure designed for storing the data obtained from web logs. To construct a WAP-tree, we need two scans of the web access sequence database: (1) Scan database once, find all frequent individual events; (2) Scan database again, construct the WAP-tree over the sub-sequences with only frequent individual events of each sequence, which are also called frequent subsequences, by merging their common prefixes,. At the same time, all nodes that contain the same frequent event are linked into an event queue and the Header Table with all frequent events is created for this WAP-tree with the head of each event queue registered in it. Then, all the nodes labeled with the same event can be visited by following the related event queue, starting from the Header Table.

The FS-tree (Pei J., Han J., Mortazavi-asl B., and Zhu H.(2000)) extends the WAP-tree structure for incremental and interactive mining. The corresponding mining algorithm FS-mine (Frequent Sequence mining) is used to analyze the FS-tree to discover frequent sequences.

J Han puts forward a web access pattern tree structure (WAP-tree) and an algorithm for mining frequent access path based on WAP-tree (WAP-mine) in (Zhou B.Y., Hui,S.C. and A.C.M. Fong(2004)). This algorithm and not producing candidate frequent patterns. Consequently, WAP-mine algorithm is an order of magnitude faster than Apriori algorithm (B Zhou B.Y., Hui,S.C. and A.C.M. Fong(2004)) put forward by Agrawal at earlier stage. Nevertheless, WAP-mine needs to produce a mass of conditional WAP-tree, which influences the efficiency of WAP-mine in a certain degree.

In recent years, some classical algorithms applied to mine maximum patterns include MaxMiner, DethProject, MAFIA and GenMax etc.

3. Prototype – BC-WAP

The proposed approach is based on WAP-tree, but avoids recursively re-constructing intermediate WAP-trees during mining of the original WAP tree for frequent patterns. The modified WAP algorithm is able to quickly determine the suffix of any frequent pattern prefix under consideration by comparing the assigned binary position codes of nodes of the tree. A tree is a data structure accessed starting at its root node and each node of a tree is either a leaf or an interior node. A leaf is an item with no child. An interior node has one or more child nodes and is called the parent of its child nodes. All children of the same node are siblings. Like WAP-tree mining, every frequent sequence in the database can be represented on a branch of a tree. Thus, from the root to any node in the tree defines a frequent sequence. For any node labeled e in the WAP-tree, all nodes in the path from root of the tree to this node (itself excluded) form a prefix

sequence of e. The count of this node e is called the count of the prefix sequence. Any node in the prefix sequence of e is an ancestor of e. On the other hand, the nodes from e (itself excluded) to leaves form the suffix sequences of e.

Given a WAP-tree with some nodes, the binary code of each node can simply be 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. In general, for the nth leftmost child, the position code is obtained by appending the binary number for 2n-1 to the parent's code. 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 α) + 1).

The tree data structure, similar to WAP-tree, is used to store access sequences in the database, and the corresponding counts of frequent events compactly, so that the tedious support counting is avoided during mining. A Binary code is assigned to each node in proposed WAP-tree. These codes are used during mining for identifying the position of the nodes in the tree. The header table is constructed by linking the nodes in sequential events fashion. Here the linking is used to keep track of nodes with the same label for traversing prefix sequences. This mining algorithm is prefix sequence search rather than suffix search.

The algorithm scans the access sequence database first time to obtain the support of all events in the event set, *E*. All events that have a support greater than or equal to the minimum support are frequent. Each node in a modified tree registers three pieces of information: node label, node count and node code, denoted as label: count: position. The root of the tree is a special virtual node with an empty label and count 0. Every other node is labeled by an event in the event set *E*. Then it scans the database a second time to obtain the frequent sequences in each transaction. The non-frequent events in each sequence are deleted from the sequence. This algorithm also builds a prefix tree data structure by inserting the frequent sequence of each transaction in the tree the same way the WAP-tree algorithm would insert them.

Once the frequent sequence of the last database transaction is inserted in the tree, the tree is traversed to build the frequent header node linkages. All the nodes in the tree with the same label are linked by shared-label linkages into a queue. Then, the algorithm recursively mines the tree using prefix conditional sequence search to find all web frequent access patterns. Starting with an event, *ei* on the header list, it finds the next prefix frequent event to be appended to an already computed *m*-sequence frequent subsequence, which confirms an *en* node in the root set of *ei* , frequent only if the count of all current suffix trees of *en* is frequent. It continues the search for each next prefix event along the path, using subsequent suffix trees of some *en* (a frequent 1-event in the header table), until there are no more suffix trees to search. To mine the tree, the algorithm starts with an empty list of already discovered frequent patterns and the list of frequent events in the head linkage table. Then, for each event, *ei*, in the head table, it follows its linkage to first mine 1-sequences, which are recursively extended until the *m*-sequences are discovered. The algorithm finds the next tree node, *en*; to be appended to the last discovered sequence, by counting the support of *en* in the current suffix tree of *ei* (header linkage event). Note that *ei* and *en* could be the same events. The mining process would start with an *ei* event and given the tree, it first mines the first event in the frequent pattern by obtaining the sum of the counts of the first *en* nodes in the suffix subtrees of the Root. This event is confirmed frequent if this count is greater than or equal to minimum support. To find frequent 2-sequences that start with this event, the next suffix trees of *ei* are mined in turn to possibly obtain frequent 2- sequences respectively if support thresholds are met. Frequent 3-sequences are computed using frequent 2-sequences and the appropriate suffix subtrees. All frequent events in the header list are searched for, in each round of mining in each suffix tree set. Once the mining of the suffix subtrees near the leaves of the tree are completed, it recursively backtracks to the suffix trees towards the root of the tree until the mining of all suffix trees of all patterns starting with all elements in the header link table are completed.

3.1 The BC-WAP Algorithm

Input : Access sequence database D(i), min support MS ($0 < MS \leq 1$)

Output : frequent sequential patterns in D(i).

Variables : Cn stores total number of events in suffix trees, A stores whether a node is ancestor in queue.

Method:

Step 1 : Construct a initial WAP tree.

Step 2 : Assign code to each node

(i) Root has null position code

(ii) Left child = 1

Step 3 : Repeat step-2 in order to find pattern

4. Experimental Result

The proposed algorithm is implemented in VB.NET and all experiments were found on Intel Pentium running on Microsoft Window XP profession. The web server log file dated on Nov 2007 from KASC (Kongu Arts and Science College) web server after preprocessed (Gomathi.C., Moorthi M., Duraiswamy K. (2008),) has been selected for our experiments. This KASC log file size is 100KB. The proposed method was applied on this web log files to prepare sequence pattern. The proposed mining algorithm has significant advantages when compared with the original WAP-mine algorithm. First, it avoids the costly construction of the initial WAP-trees. Second, the position code assigned is more efficient than the method based on the conditional WAP-trees. These special features of the proposed algorithm help improve the efficiency of the mining process significantly. The experiments showed that the proposed methodology needs less time to find frequent sequence and needs only minimum storage area. The complete sequence pattern with minimum support 31% is shown in Figure 1.

5. Conclusion

In this paper, BC-WAP mine tool developed using VB.NET for sequential access pattern from KASC web log files. The proposed system eliminates the need to store numerous intermediate WAP trees during mining. Since only the original tree is stored, it cuts off huge memory access costs. This new system assigns binary position code to each node in the WAP tree.

The focus of the framework was utilizing web usage mining with learning styles for pedagogically effective and technologically possible personalized e-learning courses. Results suggest that dimensions of learning styles, i.e. preferences to learning material, can be modeled using suitable attributes and can be detected using data mining techniques. We are currently investigating using the output of the mining tool into personalized learning scenarios, in which the learners are assisted by the system based on the patterns and the preferred learning styles. We plan to compare our work with others (Zhou B.Y., Hui S.C. and A.C.M. Fong(2004)), which has used other techniques, such as the Bayesian model or genetic algorithms.

References

Agrawal R. & Srikant R. (1994) Fast Algorithms for Mining Association. *In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, Santiago, Chile, pp. 487-499.

Gomathi.C., Moorthi M. & Duraiswamy K. (2008), Preprocessing of Web Log Files in Web Usage Mining. *The ICFAI journal of Information Technology,* Vol. 4, No. 1, pp. 55-66.

Hafidh Ba-Omar, IIias Petrounias & Fahad Anwar. (ICALT 2007). A framework for using web –usage mining to personalize E-learning, *seventh IEEE international conference on Advanced Technologies*(ICALT 2007).

Pei J., Han J., Mortazavi-asl B. & Zhu H. (2000). Mining Access Patterns Efficiently from Web Logs. *In Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD),* Kyoto, Japan, pp. 396- 407.

Zhou B.Y., Hui, S.C. & A.C.M. Fong. (2004). CS-mine: An Efficient WAP-tree Mining for Web Access Patterns. *In Proceedings of the 6th Asia Pacific Web Conference (APWeb'04), Hangzhou, China, Lecture Notes in Computer Science 3007*, Springer, pp. 523-532.

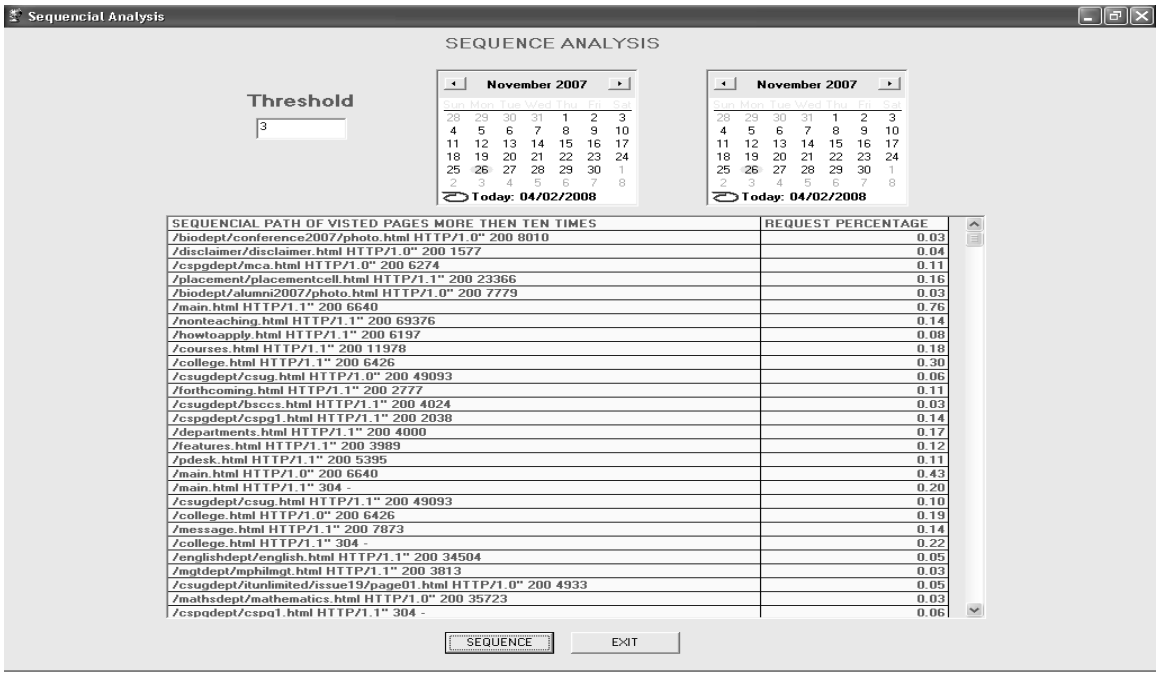

Figure 1. Sequence Pattern with minimum support 31%