AN EFFICIENT ALGORITHM TO ACCESS WEB LOG PATTERN USING WAP TREE-MINE

Karthikeswaran[1] D., Dinakar[2] S.

[1]Jayaram College of Engineering and Technology, Trichy.
[1]E-mail: karthids@gmail.com.
[2]Jayaram College of Engineering and Technology, Trichy.
[2]E-mail: s.dinakar27@gmail.com.

Abstract

World Wide Web is a huge data repository and is growing with the explosive rate of about 1 million pages a day, web log records each access of the web page and number of entries in the web logs is increasing rapidly. These web logs, when mined properly can provide useful information for decision-making. Sequential pattern mining discovers frequent user access patterns from web logs. Since Apriori-like sequential pattern mining techniques requires expensive multiple scans of database. But, recently a novel data structure, known as Web Access Pattern Tree (or WAP-tree), was developed. This proposed method an efficient WAP-tree mining algorithm, known as DLT-mine (Doubly Linked Tree algorithm). Proposed recursive algorithm uses this doubly Linked tree to efficiently find all access patterns that satisfy user specified criteria. This mining algorithm is faster than the other Apriori-based mining algorithms.

Keywords: Web mining; Pattern discovery.

I. INTRODUCTION

Web usage mining is used to identify user behavior on a particular website. It performs mining on web usage data or web logs [10]. Web usage mining looks at the log of Web access. Web server records each access of the web page in web logs. Number of entries in the web logs is increasing rapidly as the access to the web site is increasing. These web logs, when mined properly can provide useful information for decision-making. Most of the web logs contain information about fields: The mined knowledge can then be used in many practical applications, such as improving the design of web sites, analyzing user behaviors for personalized services, and developing adaptive web sites according to different usage scenarios. IP Address, User Name, Time Stamp, Access Request, Result Status, Byte Transferred, Referrer URL and User Agent. There are many efforts towards mining various patterns from Web logs [3].

Web access patterns mined from Web logs can be used for purposes like: Improving design of web sites, used to gather business intelligence to improve sales and advertisement, analyzing system performance, building adaptive Web sites [3].

Mining frequent access patterns (called sequential access pattern mining) in a sequence database was firstly introduced by Agrawal and Srikant [1] which is based on AprioriAll algorithm. After its introduction lots of work was done to mine sequential pattern efficiently AprioriAll algorithm sequence database is scanned many times to mine sequential access pattern.

In the first scan, it finds all frequent 1-event and forms a set of 1-event frequent sequences. In the following scans, it generates candidate sequences from the set of frequent sequences and checks their supports.

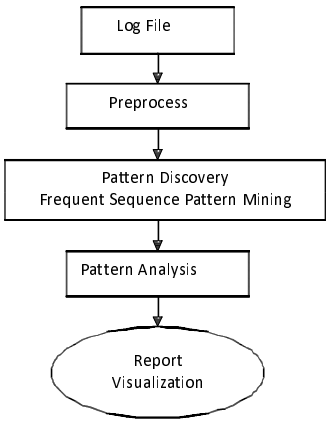


Fig 1. Web log mining structure

The problem with AprioriAll is that it does not perform well if the length of the access sequences and

transactions are large, which is the basic need of Web log mining.

II. PROBLEM STATEMENT

Web log consist of many types of information including the information about the user and the access done by the user. We can extract the unnecessary data and only keep the required data in the preprocessing phase of the log mining. If each access is regarded as event we can say that after preprocessing web log is a sequence of events from one user or session in timestamp ascending order.

Let E be a set of events. Then $S = e1e2 \ldots ek\, ek+1\, en$ for $(1 \le i \le n)$ is an access sequence and n is the length of the access sequence called n-sequence. Remember in any access sequence repetition is allowed i.e. $ei =! ej$ for $i =! j$.

Access sequence $S' = e1'\, e2'\, e3' \ldots ek'$ is called a subsequence of s equence S and S is called the super-sequence of sequence S' denoted as $S' \subseteq S$, if and only if there exist $1 \le i1 < i2 < K \ldots\ldots ik \le n$, such that $ej = eij$ for $(1 \le j \le k)$. S' is called proper subsequence of sequence S that is $S' \subset S$, if and only if S' is a subsequence of S and $S' +! S$. Subsequenc $Ss = ek+1\, ek+2 \ldots en$ of S is a super sequence of a sequence $P = ek+1\, ek+2 \ldots em$ where $m \le n$ then $Sp = e1\, e2 \ldots\ldots ek$ is called the prefix of S with respect to sequence P and Ss is called the suffix of Sp. Let Web access sequence database WAS is represented as a set $S1, S2L, Sm$ where each $Si\ (1 \le i \le m)$ are access sequences.

Then the support of access sequence S in WAS is defined as

$$Sup\ (s) = \frac{|\{Si \mid S \subseteq Si\}|}{m}$$

A sequence S is said a ζ-pattern or simply (Web) access pattern of WAS, if $Sup\ (s) \ge \zeta$. Here it is important to remember that events can be repeated in an access sequence or pattern, and any pattern can get support at most once from one access sequence.

The problem of mining access pattern is: Given Web access sequence database WAS and a support threshold ζ, mine the complete set of ζ-pattern of WAS.

III. EFFICIENT WEB LOG MINING USING DLT-MINE

The central theme of our algorithm is as follows: Scan the WAS twice. In the first scan, determine the set of frequent events. An event is called a frequent event if and only if it appears in at least (ζ. |WAS|). Where |WAS| denotes the number of access sequences in WAS and ζ denote the support threshold. In the second scan, build a doubly linked tree After creating a doubly linked tree we recursively mine it using conditional search to find all ζ-pattern.

The following observations are helpful in the construction of the doubly linked tree.

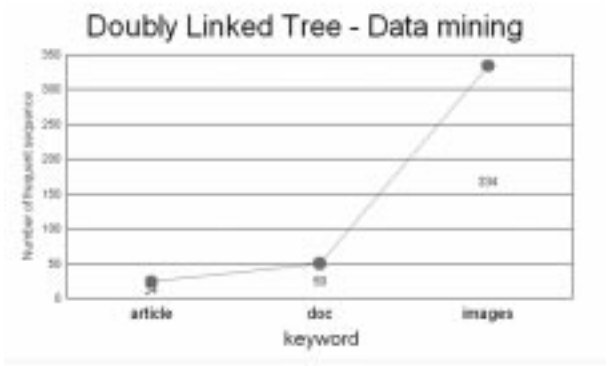
1. Apriori property that if a sequence G is not a ζ-pattern of sequence database, any super-sequence of G cannot be a ζ-pattern of sequence database is used. That means, if an event e is not in the set of frequent 1-sequences, there is no need to include e in the construction of a doubly linked tree.
2. We create a single branch for the shared prefix P in the tree. It helps in saving space and support counting of any subsequence of the prefix P.

Above observation suggest that doubly linked tree should be defined to contain following information:

- Each node must contain event (we call it label) and its count except the root node which have empty label and coun $t = 0$. The count specifies the number of occurrences of the corresponding prefix ended with that event in the WAS.
- To manage the linkage and backward traversal we need two additional pointers except the pointers tree normally has. First, all the nodes in the tree with the same label are linked by a queue called event-node queue. To maintain the front of a queue for each frequent event in the tree one header table is maintained. Second, for backward traversal from any intermediate node to the root we add a pointer to the parent at each node.

The tree construction process is as follows: First of all filter out the non frequent events from each access sequence in WAS and then insert the resulting frequent subsequence into tree started from the root. Considering the first event, denoted as e, increment the count of child node with label e by 1 if there exists one; otherwise create a child labeled by e and set the count to 1. Then, recursively insert the rest of the frequent subsequence to the sub tree rooted at that child labeled e. The complete algorithm for doubly linked tree creation is given below:

mwap-Data mining



mWAP-Data mining

Fig. Frequent Access Sequence

Algorithm 1 (Doubly Linked Tree Construction)

Input: A Web access sequence database WAS and a set of all possible events E.

Output: A doubly linked tree T.

Method:

Scan 1:

1. For each access sequence S of the WAS

1.1. For each event in E

1.1.1. For each event of an access sequence of WAS. If selected event of access sequence is equal to selected event of E then

(a) event count = event count + 1

(b) continue with the next event in *E*.

2. For each event in E if event qualify the threshold add that event in the set of frequent event **FE**.

Scan 2:

1. Create a root node for *T*

2. For each access sequence *S* in the access sequence database WAS do

(a) Extract frequent subsequence *S′* from *S* by removing all events appearing in *S* but not in FE. Let $S' = s1\,s2 \ldots sn$, where $si\,(1 \le i \le n)$ are events in *S′*. Let current node is a pointer that is currently pointing to the root of *T*.

(b) For $i = 1$ to *n* do, if current node has a child labeled *si*, increase the count of *si* by 1 and make current node point to *si*, else create a new child node with label = *si*, count = 1, parent pointer = current node and make current node point to the new node, and insert it into the *si*-queue.

3. Return (T);

After the execution of this algorithm we get doubly linked tree. This contains all the information in very condensed form. Now we do not need WAS database to mine the access pattern. The length of the tree is one plus the maximum length of the frequent subsequences in the database. The width of the tree that is the number of distinct leaf nodes as well as paths in a doubly linked tree cannot be more than the number of distinct frequent subsequences in the WAS database. Access sequences with same prefix will share some upper part of path from root and due to this scheme size of the tree is much smaller than the size of WAS database.

Maintaining some additional links provides some interesting properties which helps in mining frequent access sequences.

1. For any frequent event *ei*, all the frequent subsequences contain *ei* can be visited by following the *ei* -queue, starting from the record for *ei* in the header table of doubly linked tree.

2. For any node labeled *ei* in a doubly linked tree, all nodes in the path from root of the tree to this node (excluded) form a prefix sequence of *ei*. The count of this node labeled *ei* is called the count of the prefix sequence.

3. A path from root may have more than one node labeled *ei*, thus a prefix sequence say G of *ei* if it contain another prefix sequence say H of *ei* then G is called the super-prefix sequence and H is called the sub-prefix sequence. The problem is that super-prefix sequence contributes in the counting of sub-prefix sequence. This problem is resolved using unsubsumed count**.** A prefix sequence of *ei* without any super-prefix sequences, unsubsumed count is the count of *ei.* For a prefix sequence of *ei* with some super-prefix sequences, the unsubsumed count of it is the count of that sequence minus unsubsumed counts of all its super-prefix sequences.

4. It is very difficult to traverse from root to the node pointed by the *ei* -queue because it requires several traversal hits to get required prefix. Parent pointer allows backward traversal from any intermediate node pointed by *ei* -queue to the root and efficiently extract the prefix sequences.

With the above information we can apply conditional search to mine all Web access patterns using doubly linked tree. Conditional search means, instead of searching all Web access patterns at a time, it turns to search Web access patterns with same suffix. This suffix is then used as the condition to narrow the search space. As the suffix becomes longer, the remaining search space becomes smaller potentially.

The algorithm to mine all ζ-patterns is as follows:

Algorithm 2 (Mining all ζ-patterns in doubly linked tree)

Input: a Doubly linked tree ***T*** and support threshold ζ.

Output: the complete set of ζ-patterns.

Method:

1. If doubly linked tree T has only one branch, return all the unique combinations of nodes in that branch

2. Initialize Web access pattern set WAP=¥õ . Every event in T itself is a Web access pattern, insert them into WAP

3. For each event *ei* in *T*,

(a) Construct a conditional sequence base of *ei*, i.e.PS(*ei*), by following the *ei*-queue, count

conditional frequent events at the same time.

(b) If the set of conditional frequent events is not empty, build a conditional doubly linked tree for *ei* over PS(*ei*) using algorithm 1. Recursively mine the conditional doubly linked tree

(c) For each Web access pattern returned from mining the conditional doubly linked tree, concatenate *ei* to it and insert it into WAP.

4. Return WAP.

IV. RESULT AND ANALYSIS

Since these web logs are in different format we did some preprocessing work to convert this web log into the Web Access Pattern Dataset (WASD) format. We filter out the web logs according to our need. The proposed algorithm is implemented in Microsoft Visual Studio 2005 .NET and all experiments were found on Intel Pentium running on Microsoft Window XP profession. The web server log file size 101 KB.

As the results shows performance of the doubly linked tree mining:

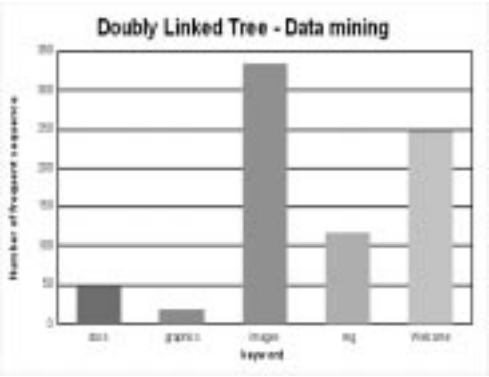


Figure 2. Frequent Access Sequences.

V. CONCLUSION

In this paper, DLT- mine developed using VB.NET for sequential access pattern from JCET web log files. Doubly Linked Tree mining performance is much better than Apriori Base Algorithms. Efficient web usage mining could benefit from relating usage of the web page to the content of web page. Some other area of interest may be implementing Doubly Linked mine algorithm to the distributed environment.

VI. SCREENSHOTS

Fig. Raw Web File

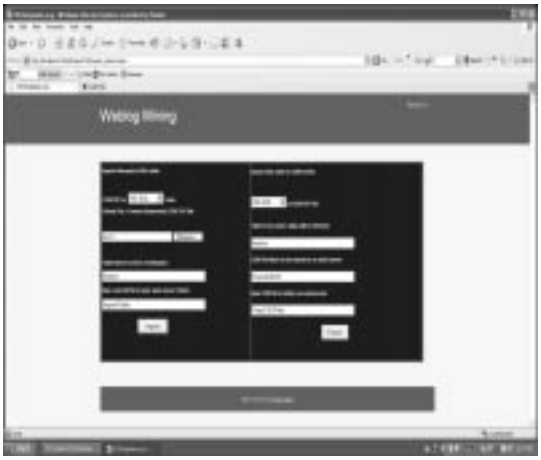


Fig. Import Web Log File

Fig. Export Web Log File

Fig. DLT Mine

REFERENCES

[1] 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, Santiago, Chile, September 1994.

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

[3] Cooley.R, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing pattern. Knowledge and Information Systems, 1(1), 1999.

[4] Ezeife, C. and Lu, Y. Mining web log sequential patterns with position coded preorder linked wap-tree. International Journal of Data Mining and Knowledge Discovery (DMKD) Kluwer Publishers, p.5–38, 2005.

[5] Gomathi.C., Moorthi M. & Duraiswamy K. (2008), Web Access Pattern Algorithms in Education Domain. Computer and information science, November 2008, vol 1. no.4.

[6] 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.

[7] 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).

[8] Han, J., Pei, J., Mortazavi-Asl, B. and Pinto, H. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In Proceedings of the 001International Conference on Data Engineering (ICDE 01), p.214–224, 2001.

[9] Jiawei Han, Micheline Kamber, Data Mining: Concepts and Techniques , Mrogan kaufmann Publication,2002.

[10] Kosala R, and H. Blockeel. (2000). Web Mining Research: A Survey. In ACM SIGKDD Explorations, Vol.2, pp. 1-15.

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

[12] Pujari, A. : Data Mining Techniques ,Universities Press, India, February 2001.

[13] Raymond Kosala, Hendrik Blockeel, Web Mining Research: A Survey, ACM SIGKDD, July 2000.

[14] Srikant.R and R. Agrawal. Mining quantitative association rules in large relational tables. In Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data, pages 1-12, Montreal, Canada, June 1996

[15] Srivastava.J, R.Cooley, M. Deshpande, and P.-N. Tan. (2000). Web Usage Mining:Discover and Applications of Usage Patterns from Web Data. In ACM SIGKDD Explorations, Vol. 1, No. 2, pp.12-23.

[16] Suneetha K. R., Dr. R. Krishnamoorthi, Identifying User Behavior by Analyzing Web Server Access Log File, IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.4, April 2009.

[17] Vasumathi.D, A .Govardhan, BC-WASPT : Web Acess Sequential Pattern Tree Mining, International Journal of Computer Science and Network Security, VOL.9 No.6, June 2009.

[18] Yu Hirate and Hayato Yamana," Generalized Sequential Pattern Mining with Item Intervals", 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2006), Singapore, April 9-12, 2006, journal of computers, vol. 1, no. 3, june 2006.

[19] Zaki, M. SPADE: An efficient algorithm for mining frequent sequences. Machine Learning, p.31–60, 2001.