ACSIJ Advances in Computer Science: an International Journal, Vol. 4, Issue 1, No.13 , January 2015
ISSN : 2322-5157
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

Representing an Effective Approach to Understand the Dynamic Frequent Pattern of Web Visitor

Farid Soleymani Sabzchi [1], Mehdi Afzali [2]

[1]Department of Computer Engineering, Zanjan Branch, Islamic Azad University

Zanjan, Iran
farid.soleymani@yahoo.com

[2]Department of Computer Engineering, Zanjan Branch, Islamic Azad University

Zanjan, Iran
Mehdi.afzali.2@gmail.com

Abstract

Developing word of the Web, increasing the content information and requirements of user's Web site has been changed. Therefore, Web needs a dynamic and an accurate algorithm to recognize user's requirements to suggest new patterns. There are many algorithms proposed for discovering frequent patterns. Mining frequent patterns is one of the fundamental and essential operations in many data mining application such as discovering association rules. In various applications, database frequently changes by inserting, deleting and modifying transaction. The proposed algorithm has the potential to apply these four factors to modify database in the path tree by incremental mining. This algorithm has been compared to similar algorithms such as CATS-tree, AFPIM, CAN-tree and CP-tree. Suggested algorithm has lower time complexity and higher speed in compare with other algorithms. To describe this algorithm, an illustrative example is presented. Obtained results show that extracted patterns by this method will specify degree of user's interest to the pages more accurately and also the steps of sorting branches after applying any change in the tree has the lowest executive order in compare with other algorithms.

Keywords: *Incremental Mining, Path Tree, Frequent Patterns, association rules.*

1. Introduction

Today, needs of Web users are changing. Therefore, the transactions of database change. There are states that might change database which are explained at four conditions [1-6].

1. The new requests send to Web server, therefore information if users is added to log file and database as new rows. In the other word, some new transaction data is added to the old transaction database.

2. Maybe one or several rows have been deleted from transaction database. In the other word, old transaction data is deleted from the transaction database.

3. By changing users' interest to pages, perhaps the pages that have been met become less popular. It causes adding new nodes to the path tree. Therefore, new frequent items will add to the path tree.

4. By changing users' interest to pages, perhaps the pages that have been met become lower view. It causes deleting nodes from path tree. In other words, some frequent items will delete from path tree.

Cheung and Zaiane proposed FELINE algorithm with CATS-tree [7], elsewhere KOH and Shieh suggested AFPIM algorithm [8]. In the CAN-tree all items are arranged in canonical order so it usually yields poor compaction in tree compare with the size of FP tree [9] .The Can-tree does not require adjustment, merging and splitting of tree nodes during maintenance.
In CP-tree it construct compact tree data structure with one scan of database and provides the same mining performance as the FP-growth technique by efficient tree restructuring process[10]. According to this approach, items frequencies in previous transactions are used to insert each transaction into proposed tree. Then, this tree is restructured using Branch-Sorting method. Applying presented tree will have led to increase the efficiency of Branch-Sorting restructuring method.

2. ALGORITHM OF SORTING BRANCH

To apply the above states on path tree, branches of tree must be sorted. The most common algorithms for sorting branches are Path adjusting method and Branch sorting method. These methods are described in the following.

2.1 Path Adjusting Method

The path adjusting method was proposed in [11] where the authors used this technique to reorder the structure of current constructed FP-tree due to updating the DB. In this method, the paths in a prefix-tree are adjusted by recursively swapping the adjacent nodes in the path unless the path has completely achieved the new sort order. Thus, it uses the bubble sort technique to swap between the two nodes. When both I-lists are in opposite order, the swapping between two nodes in a path must consider the bubble sort running cost of O (n^2). It means that items are ordered in reverse direction from each other. The worst case swapping complexity of this method which observed is O (mn^2) where m is the total number of paths and n in the average length of all transactions.

2.2 Branch Sorting Method

The Branch sorting method is unlike the Path adjusting method [10]. first obtains the Isort by rearranging the items in I in a frequency-descending order and then performs the restructuring operation on T. It is an array-based technique that performs the branch-by-branch restructuring process from the root of T. Each sub-tree under each child of root can be treated as a branch. Therefore, a tree T contains many branches as some of children that are under the root. Each branch may consist of several paths. By restructuring a branch in BSM, each path in the branch is sorted according to the new sort order by removing it from the tree. This sorting is done in a temporary array and again inserts the removed branch into the tree. However, while processing a path, if it is found to already be in sorted order, the path is skipped and no sorting operation is performed. Finally, the restructuring mechanism will complete when all the branches are processed which produces the final Tsort. On the other hand, the BSM uses a merge sort approach to sort the nodes of each path. Therefore, the degree of disorder does not have a large effect on the performance during sorting, since irrespective of data distribution the complexity of merge sort is always O ($n\log_2 n$), where n is the total number of items in the list. Moreover, the number of intermediate nodes to be sorted is also not an influencing parameter in BSM. One of the important features of BSM is handling the branches that have sorted path(s) which can reduce the number of sorting operations and/or the size of data to be sorted. The cost of sorting all paths in the tree is O ($mn\log_2 n$), where n is the average length of transactions and m is the number of paths in the tree.

3. RELATED WORK

In [12] authors planned the AFPIM algorithm for incremental mining. Similar to FP-tree, it only keeps frequent items. In this algorithm, a threshold called PreMinsup is considered which its values are set less than the Minsup. Since items are structured based on the number of events, the insertion, deletion or modification of transactions may affect the frequency and order of the items. More particularly, items in the tree are adjusted when the order of them is changed. The AFPIM algorithm swaps such items by applying bubble sort algorithm that involves vast calculation.

In [13] authors presented a new tree structure called Can-Tree. CAN-tree algorithm is used for incremental mining and requires only one database scan. According to this algorithm, items are structured on the basis of a canonical standard (e.g. alphabetical) which can be determined via the user. So any change in frequency which is caused by incremental updates (such as insert, delete, or modify transactions) will not affect the order of items in the CAN-tree. Therefore, new transactions are inserted into the tree without exchange any node of the. The CAN-Tree can be easily maintained when database transactions are inserted, deleted, and/or modified. For example, the CAN-Tree does not require adjustment, merging and/or splitting of tree nodes during upholding. No rescan of the entire updated database or reconstruction of a new tree is required for incremental updating.

In [14] a tree structure called CATS is planned to mine frequent patterns in an incremental method. The proposed tree structure improves FP-tree on data compression and allows extracting frequent patterns without the need to produce a set of candidate items. According to this method, the primary transaction in database is added to the tree roots. And for later transactions, the item within the transaction is compared with the items in the tree for recognizing similar items. Generally, if there is any item in nodes of the tree and the transaction, then transaction is merged with the node that has the maximum frequency level. Then, the rest of the transaction is added to the merged nodes.

In [15] a novel tree structure called CP-tree is put forward. CP-tree is a dynamic tree which can be used to interactive as well as incremental mining. In this technique, all the transactions are inserted into the tree in agreement with a predefined item order. The item order of a CP-tree is maintained through a list, called I-list. After inserting some of the transactions, if the item order of the I-list differs from the present frequency-descending item order to a predefined degree, the CP-tree will be restructured through a method called the branch sorting. Then, the item order is updated with the present list.

4. PROPOSED ALGORITHM

In this part, we present a new tree data structure which is an extension of CP-tree. We have seen that CP-Tree contains all the items (frequent and also non-frequent) in the tree at the mining time. We introduce a new tree which contains all the items. In the proposed algorithm, frequent item sets from database with only one database scan like CP-tree with CP-mine algorithm. In FP growth algorithm when mining a long frequent item set on great database, the algorithm is considerably outperforms the Apriori algorithm. When we keep only small part of candidate item sets becomes frequent item sets then generating FP tree which is very costly .In CP-mine it take out frequent item set from our proposed tree structure by using pruning tree and marked value technique. Like a CP-tree, our new proposed tree structure contains two phases.

1. Insertion Phase
2. Restructure Phase.

In first phase it scans all transaction(s) inserted in to tree according to current item order of I-list and inform the frequency count of personal items in I-list. Next phase is restructuring I-list according to frequency descending order of items and restructures a tree with nodes according to new I-list, but our proposed data structure similar to FP tree, contains only frequent items in restructure phase. In this phase it sorts items using Branch Sorting method. It is an array-based technique that performs the branch-by-branch restructuring process from the root. In BSM it sorts each path in the branch according to the new sort order with removing path from the tree it sorting in to temporary array and once more inserting it in to the tree. However, while processing a path, if it establish path which is already sorted order then that path will skip and go to the next path. At last restructuring method finishes when all the branches are processed which produces the final sort. Here these two phases are dynamically executed in alternate approach, starting with insertion phase the first part of database and finishes with the restructuring phase at the end of database.

5. PROPOSED TREE CONSTRUCTION

In this part, a new tree structure has been introduced .In the proposed method, the first transaction is inserted into the l-list and based on descending order of count value.in the next step sorted items in the l-list are added to the tree branch. This process is repeated until the last transaction of database. If deviation rate of items order in l-list is much more than deviation threshold, tree of interest is restructured. In Branch-Sorting method, all paths that present in tree are evaluated and in case items would not ordered in terms of their descending frequencies, path of interest is removed from tree and having been ordered, it is inserted in tree again. So it is worthy to note that one of the effective factors in efficiency of Branch-Sorting method is ordering degree of associated tree. To evaluate this method, both proposed tree and CP-tree created in previous section are restructured based on Branch-Sorting restructuring method. In this approach items which have the same frequency are ordered alphabetically.

TABLE: 1 The transactional database

TID	Items
1	a,b,c
2	d,e
3	a,b,d
4	a,d,c,e

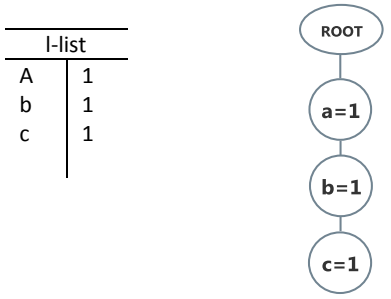

Fig.1. Transaction1 added to the tree

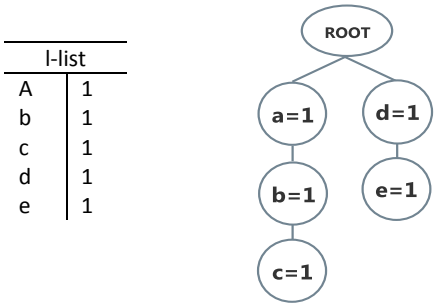

Fig.2. Transaction2 added to the tree

l-list	
A	2
b	2
d	2
c	1
e	1

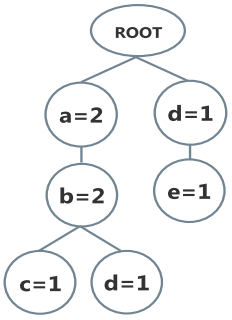

Fig.3. Transaction3 added to the tree

l-list	
A	3
d	3
b	2
c	2
e	2

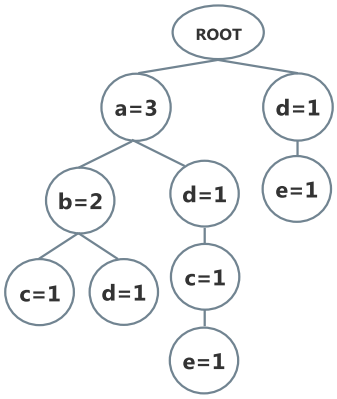

Fig.4. Transaction4 added to the tree

After adding the transactions to the tree, all paths are sorted based on the last status of L-list by using the BMS algorithm.

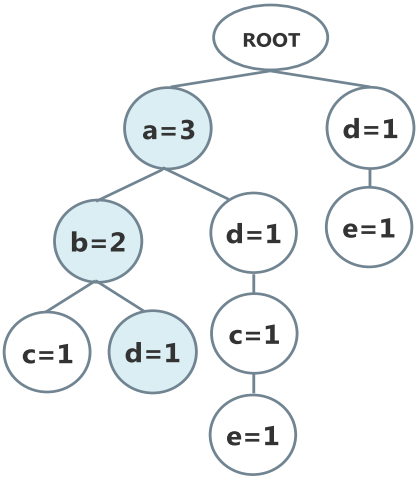

Fig.5. Extracted unordered paths

As can be seen in Fig.5, the path (a,b,c) is unsorted. After applying the BMS algorithm the final path tree is shown in Fig.6.

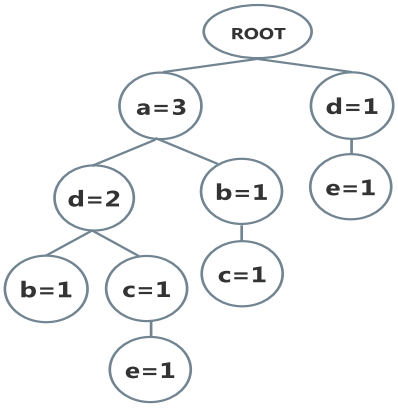

Fig. 6. The final tree

6. EXPERIMENT RESULTS

All algorithms for building the path tree and discovering frequent patterns are being measured based on time complexity, speed, time of building tree and ability to apply FP-growth algorithm, according to the following table.

TABLE: 2 The comparison criteria

Comparison criteria	**AFPIM**	**Can-tree**	**CATS-tree**	**CP-tree**	**Proposed Algorithm**
Total Access Database	2	1	1	1	**1**
The resulting tree	Sorted	Unsorted	Sorted	Sorted	**Sorted**
Time complexity of sorting one branch	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n\log_2 n)$	**$O(n\log_2 n)$**
Ability to apply the FP-growth algorithm	Yes	No	No	Yes	**Yes**
Number of frequencies of branches sorting	-	-	-	$\frac{m(m+1)}{2}$	**M**
Time complexity of sorting total path tree	$O(mn^2)$	$O(mn^2)$	$O(m^2n^2)$	$O(m^2n\log_2 n)$	**$O(mn\log_2 n)$**

AFPIM algorithm has two accesses to database that is inappropriate in terms of speed and time. This algorithm cannot support all the dynamic modes. For example, if a new node is added to path tree as a frequent item, the tree must be rebuilt.

The CATS-tree algorithm to insert a node in the tree has to identify a common node in the tree. This algorithm uses a

bubble sort algorithm to sort branches which its executive order is o (n^2). The large numbers of repeats of the operations such as merge and split can lead the algorithm to be not optimal.

The CAN-tree algorithm to insert a node in the tree has a special order such as alphabetical order that in the worst case, all branches may be irregular.

In the CP-tree algorithm after insert each branch in the tree, branches previously ordered based on the latest status of the list. This algorithm for sorting m branches needs $\frac{m(m+1)}{2}$ sorting operations. Time complexity of sorting algorithm is of order o ($m^2 n\log_2 n$).

The proposed algorithm for sorting m branches with n nodes has a time complexity o ($mn\log_2 n$). Results show this algorithm has higher efficiency compared to similar algorithms.

In the part, the performance of our proposed structure evaluated by comparing with it CP tree structure. All experiments are performed on a 3.0 GHz Pentium 3 GB memory running on a window 7 home basic. Here we perform experiments on two kind of dataset synthetic (Mushroom and T10I4D100K) and real world dataset. Here we take both databases from Item set Mining Dataset Repository

TABLE: 3 Time require for execute mushroom database

Support value	CP-tree	Proposed algorithm
0.2	23000	2406
0.25	23782	2015
0.3	23047	1719
0.35	23360	1594

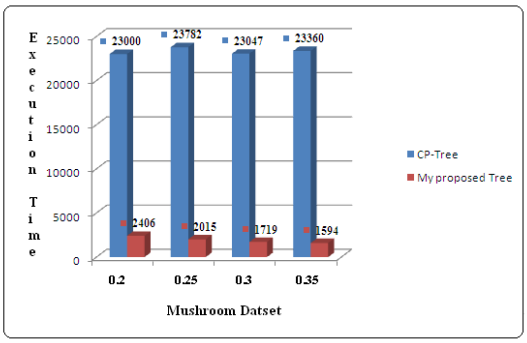

Fig.7.Compare runtime (CP-tree & proposed tree) on the Mushroom database

The time of discovering frequent patterns with different support values CP-tree algorithm and method is shown in table3.

Performing suggested algorithm on mushroom database with different support values is faster and more efficient compare to CP-tree algorithm as seen in figure 7.

The time of obtaining frequent patterns by execute time is shown in figure 7.

TABLE: 4 Time require for execute T10I4D100K database

Support value	CP-tree	Proposed algorithm
0.015	386188	188094
0.02	391407	206172
0.025	381000	218969
0.03	380579	199125

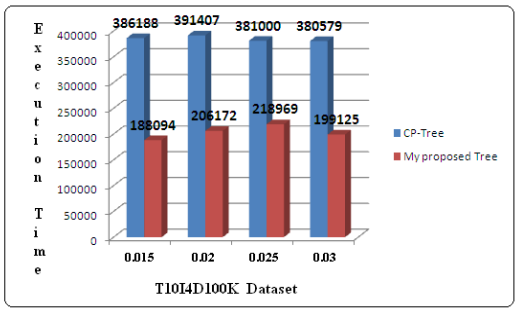

Fig.8.Compare runtime (CP-tree & proposed tree) on the T10I4D100K database

The result of studying the time of obtaining frequent patterns of database T10I4D100K by four different support values is that suggested algorithm is faster than CP-tree algorithm.

7. Conclusions

Discovering behavioral patterns of Web users is one of the most important issues in Web mining. To understand the behavior of Web users, accurate and dynamic algorithms are required in order to extract the information and interest of Web users and have ability to apply changing on tree with any changes to the database. There are several dynamic algorithms to discover frequent pattern in order to identify the needs of users and adopt their structure with the changes of users' interests. The assessment concluded that the proposed algorithm can reduce the runtime and show more flexibility by changes in the database.

References

[1] R. Agrawal, et al. Mining Associations between Sets of Items in Massive Databases. Proc. Of the ACMSIGMOD 1993 Int'l Conference on Management of Data, Washington D.C., May 1993: 207-216.

[2] N. Pasquier, et al. discovering frequent closed item sets for association rules. ICDT'99, Israel, 1999: 398-416.

[3] S. Brin, et al. Dynamic item set counting and implication rules for market basket data. Proceedings of ACM SIGMOD International Conference on Management of Data. New York: Association for Computing Machinery, 1997:255-264.

[4] D.W. Cheung, et al. A General Incremental Technique for Updating Discovered association Rules. In Proc. 1997 int'l Conf. On Databases Systems for Advanced Applications, Melbourne, Australia, April, 1997.

[5] L. Xiao, et al. A Multidimensional Scaling Based Algorithm for Fast Mining Association Rules. Journal of Software, 1999, 10(7): 749-753.

[6] B. Shi, et al. An improved incremental updating Algorithm for Mining Association Rules. Mini-Micro System, 2000, 12:1327-1329.

[7] W. Cheung and O.R. Za¨ıane. Incremental mining of frequent patterns without candidate generation or support constraint. In Proc. IDEAS2003, pp. 111–116.

[8] J.-L. Koh and S.-F. Shieh. An efficient approach for maintaining association rules based on adjusting FP-tree structures. In Proc. DASFAA 2004, pp. 417–424.

[9] Leung, C. K-S., Khan, Q. I., Li Z., & Hoque, T. "CanTree: A Tree Structure for Efficient Incremental Mining of Frequent Patterns". Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM"05), 2005.

[10] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, and Young-Koo Lee, "CP-Tree: A Tree Structure for Single-Pass Frequent Pattern Mining." In: Springer-Verlag Berlin Heidelberg 2008, pp 1-6.

[11] X. Li, X. Deng, S. Tang, A fast algorithm for maintenance of association rules in incremental databases, in: ADMA, 2006, pp. 56–63.

[12]Lee, C-H., Lin, C-R., & Chen, M.S., "Sliding window filtering: an efficient method for incremental mining on a time-variant database". In ELSEVIER-Information Systems,30(3), 2005, pp. 227-244.

[13]Carson Kai-Sang Leung, Quamrul I. Khan, Zhan Li · Tariqul Hogue "CanTree: a canonical-order tree for incremental frequent-pattern mining." In: Springer-Verlag London Limited 2006

[14]Chen, M.S., Park, J.S. and Yu, P.S. "Data mining for path traversal patterns in a Web environment," in 16th International Conference on Distributed Computing Systems, 1996, 385-392.

[15]Gunaseelan, D. and P. Uma, An Improved Frequent Pattern Algorithm for Mining Association Rules. International Journal of Information and Communication Technology Research, 2012. 2(5): p. 436 - 441.

Farid Soleymani Sabzchi received his M.Sc. degree in computer engineering at Islamic Azad University, Zanjan Branch, Zanjan, Iran in 2014. He works as an ICT expert in Office of education, Ardabil, Iran.
His research interest spans the area of Web Mining issues, especially in the field of web usage mining and pattern discovery. He published some papers in International Journals.

Mehdi Afzali received his Ph.D. degree in IM at Hacet teppe, Ankara, Turkey. He is a lecturer in Department of Computer, Islamic Azad University, Zanjan, Iran.
His research interest spans the area of Web Mining issues, especially in the field of web usage mining, Information systems and digital libraries. He published some papers in International Journals.