Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 73 (2014) 26469-26472

ISSN: 2229-712X

A Survey on Mining Frequent Patterns in data mining

Rajeev Kumar Gupta and Roshni Dubey
SRIT Jabalpur.

ARTICLE INFO

Article history:
Received: 6 May 2013;
Received in revised form:
16 August 2014;
Accepted: 25 August 2014;

Keywords
Frequent Pattern Mining(FPM),
Association Rule Mining(ARM),
Itemsets,
Transactional Database,
Minimum Support and Confidence.

ABSTRACT
Frequent pattern mining is a process of mining data as a set of itemsets or patterns from a transactional database which support the minimum support threshold. A frequent pattern is a pattern (ie. a set of items, substructures, subsequences etc.) that occurs frequently in a dataset. Association rule mining is a process of mining data as a set of rules from a transactional database which support the minimum support and confidence. The implementation methods uses special data structures to solve the problem of FPM and ARM. This paper presents some of the data structures for FPM with their advantages and disadvantages.

Introduction

Frequent pattern mining and Association rule mining(ARM) plays a very important role in data mining. There are numerous studies about the problems of frequent pattern mining and association rule mining in large databases. These studies are mainly categorized as based on their functionality and based on their performance. The functionality means what kind of data to mine either as a rule or as a pattern which is related to ARM and FPM[5]. The performance means how to compute the frequent patterns and association rules using efficient algorithms.

The algorithms used for frequent pattern mining is divided into 2 categories : Apriori based algorithms and tree-structure based algorithms. The apriori-based algorithms uses a generate-and-test strategy(ie)they finds frequent patterns by constructing candidate items and checking their support counts or frequencies against the transactional database. Examples are: FP(Fast Update)[11], UWEP(Update With Early Pruning)[12].

The tree-structure based algorithms follows a test only approach(ie)there is no need to generate candidate items and tests only the support counts or frequencies. Examples are FP-tree and FP-growth, CAN-tree, CAST-tree, trie structure etc.

The data structures have to be chosen, if they are able to support incremental mining. Generally apriori-based incremental mining algorithms are not easily adoptable with tree-structure based incremental mining algorithms. This document is a template. An electronic copy can be downloaded from the conference website. For questions on paper guidelines, please contact the conference publications committee as indicated on the conference website. Information about final paper submission is available from the conference website.

FP-TREE (FREQUENT PATTERN TREE)

A tree structure in which all items are arranged in descending order of their frequency or support count. After constructing the tree, the frequent items can be mined using FP-growth[1].

Creation of FP-Tree

First Iteration

Consider a transactional database which consists of set of transactions with their transaction id and list of items in the transaction. Then scan the entire database. Collect the count of the items present in the database. Then sort the items in decreasing order based on their frequencies (no. of occurrences).

Second Iteration:

Now, once again scan the transactional database. The FP-tree is constructed as follows. Start with an empty root node. Add the transactions one after another as prefix subtrees of the root node. Repeat this process until all the transactions have been included in the FP-tree. Then construct a header table which consists of the items, counts and their head-of-node links. Consider the transactional database shown in Table 1 with 5 transactions.

TABLE I
EXAMPLE OF TRANSACTIONAL DATABASE

tran. id	items
t1	a,b,d,e
t2	a,c,d
t3	e,f,h,i
t4	a,b
t5	c,e,f

The frequent itemlist for the above database is given in Table 2.

TABLE III
FREQUENT ITEMLIST FOR THE TRANSACTIONAL DATABASE IN TABLE1

Items	Count
A	3
B	2
C	2
D	2
E	3
F	2
H	1
I	1

The items that does not meet the minimum threshold has been eliminated. The frequent itemlist that support the minimum support threshold is given in Table 3.

Tele:
E-mail addresses: errajeevgupta@yahoo.co.in

TABLE IIIII
FREQUENT ITEMLIST FOR THE TRANSACTIONAL DATABASE THAT SUPPORT MINIMUM THRESHOLD

Items	Count
A	3
E	3
B	2
C	2
D	2
F	2

The transactional database according to the frequent item list is given in Table 4.

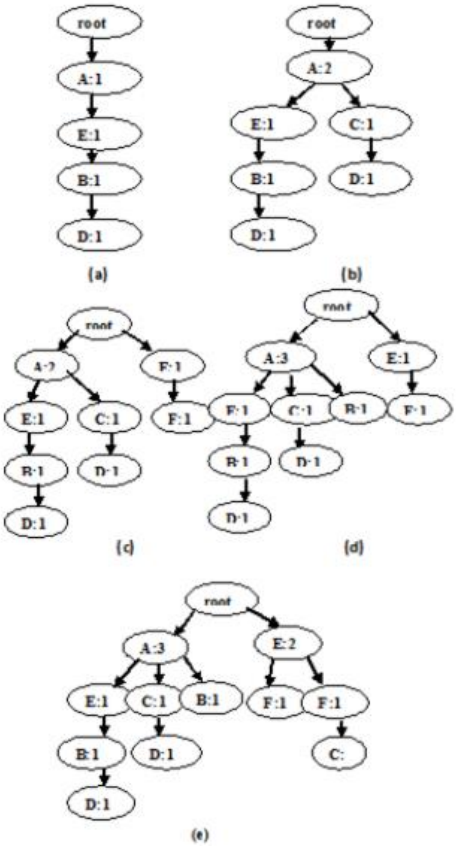

Fig. 1 Steps in Creating the FP-Tree.

TABLE IVV
SORTED AND ELIMINATED TRANSACTIONS OF THE DATABASE IN TABLE 1

Tran. ID	Items
T1	A,E,B,D
T2	A,C,D
T3	E,F
T4	A,B
T5	E,C,F

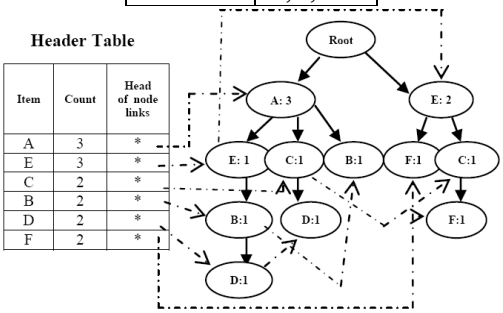

Fig. 2 FP-Tree with Header Table

Finding Frequent Patterns from FP-Tree

After the construction of FP-tree, the frequent patterns can be mined using a iterative approach FP-growth. This approach looks up the header table shown above and selects the items that supports the minimum support. It removes the infrequent items from the prefix-path of a existing node and the remaining items are considered as the frequent itemsets of the specified item.

Consider the item D.Its prefix paths are {((A,E,B):1),((A,C):1)}. After removing the infrequent items,(A:2).So the frequent itemset for D is A.

Advantages and Disadvantages

This method is advantageous because, it doesn't generate any candidate items. It is disadvantageous because, it suffers from the issues of spacial and temporal locality issues.

CAN-TREE (CANONICAL TREE)

A tree structure that arranges or orders the nodes of a tree in some canonical order. It follows a tree-based incremental mining approach. Like FP-tree approach, there is no need to rescan the transactional database when it is updated. Because of following the canonical order, frequency changes(if any) due to incremental updates like insertion, deletion and modification of the transactions will not affect the ordering of the nodes in CAN tree[3]. After constructing the CAN tree, we can mine the frequent patterns from the tree.

Creation of can tree consider the following database

TABLE VV
EXAMPLE OF TRANSACTIONAL DATABASE

TID		items	
DB	Original database	T1 T2 T3	{a,c,d,g} {b,c,d,e} {b}
DB1	1st group of insertions	T5	{a,e,f}
DB2	2nd group of insertions	T6 T7	{b,c} {a}

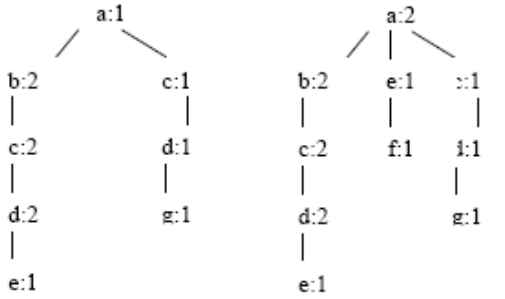

Fig 3. Initial CAN-tree

Fig 4. CAN-tree after 1st group of insertions.

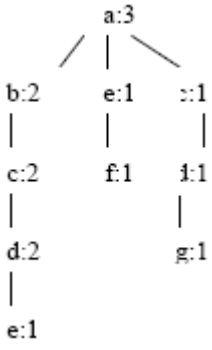

Fig 5. CAN-tree after 2nd group of insertions.

Finding Frequent Patterns from CAN-Tree

After constructing the CAN-tree, we have to mine the frequent patterns by traversing the tree in a upward direction. This can be done similar to FP-growth by constructing a header table and finding only the frequent items.

Advantages and Disadvantages

This method is advantageous because it supports incremental updates without any major changes in the tree.

COFI-TREE

Cofi is much faster than FP-Growth and requires significantly less memory. The idea of COFI is to build projections from the FP-tree each corresponding to sub-transactions of items co-occurring with a given frequent item. These trees are built and efficiently mined one at a time making the footprint in memory significantly small.

The COFI algorithm generates candidates using a top-down approach, where its performance shows to be severely affected while mining databases that has potentially long candidate patterns that turns to be not frequent, as COFI needs to generate candidate sub-patterns for all its candidates patterns and also build upon the COFI approach to find the set of frequent patterns but after avoiding generating useless candidates.

Creation of COFI-Tree

The basic idea of our new algorithm is simple and is and is based on the notion of maximal frequent patterns. A frequent item set X is said to be maximal if there is no frequent item set X' such that X€ X'.

Frequent maximal patterns are a relatively small subset of all frequent item sets. In other words, each maximal frequent item set is a superset of some frequent item sets. Let us assume that we have an Oracle that knows all the maximal frequent item sets in a transactional database. Deriving all frequent item sets becomes trivial. All there is to do is counting them, and there is no need to generate candidates that are doomed infrequent. The oracle obviously does not exist, but we propose a pseudo-oracle that discovers this maximal pattern using the COFI-trees[4] and we derive all item sets from them. Consider the following database,

TABLE IV

EXAMPLE OF TRANSACTIONAL DATABASE

d	e				
a	b	d	f	e	h
a	g	d	e	c	b
a	g	d	f	e	c
a	g	e	b	f	
a	d	h	e	f	c
a	g	d	b	l	
a	b	c	f		
a	d	b	c	g	
a	f	b	c	e	
a	b	c	h		

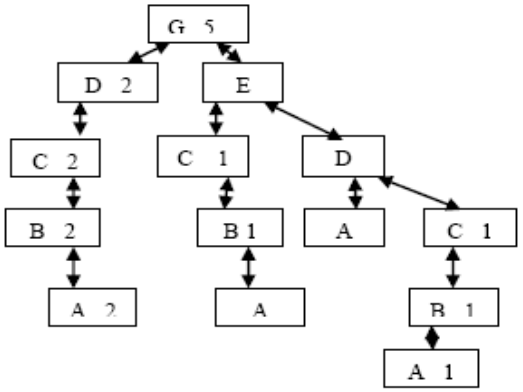

Fig 6. Creation of COFI Tree

Finding Frequent Patterns from COFI-Tree

After constructing the COFI tree, the frequent patterns in the tree can be found by identifying the frequent path bases. Then find the local maximal patterns of different sizes. Then frequent patterns can be found from the identified local maximal patterns.

Advantages and Disadvantages

This method is advantageous because the size of the generated candidate item list is minimized. It is disadvantageous because maximum effort is required for minimization.

ATS-TREE (COMPRESSED AND ARRANGED TRANSACTION SEQUENCES TREE)

This tree extends the idea of FPTree to improve storage compression and allow frequent pattern mining without generation of candidate itemsets. CATS algorithms enable frequent pattern mining with different supports without rebuilding the tree structure. The algorithms allow mining with a single pass over the database as well as efficient insertion or deletion of transactions at any time.

Cats Tree And Cats Tree algorithms. Once CATS Tree is built, it can be used for multiple frequent pattern mining with different supports. CATS Tree and CATS Tree algorithms allow single pass frequent pattern mining and transaction stream mining. In addition, transactions can be added to or removed from the tree at any time.

Creation of CATS Tree

Consider the transactional database,

TABLE VVII

EXAMPLE OF TRANSACTIONAL DATABASE

		TID	ITEMS
DB	Original database	T1 T2 T3	{a,c,d,g} {b,c,d,e} {b}
DB1	1st group of insertions	T5	{b,e,f}
DB2	2nd group of insertions	T6 T7	{b,c} {b}

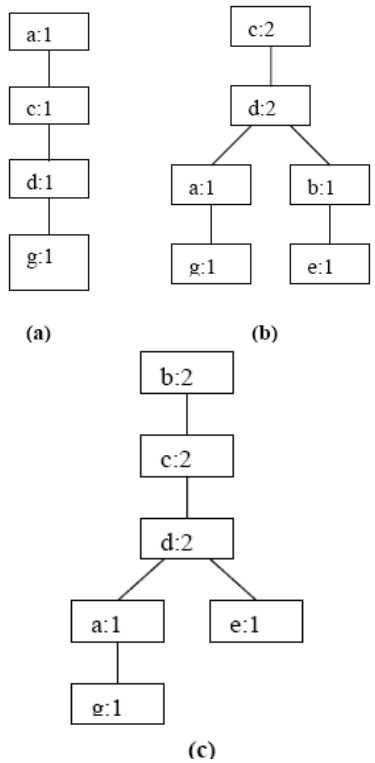

Fig 7. Steps in Creation of CATS tree

Finding Frequent Patterns from CATS-Tree

After constructing the CATS tree, the frequent patterns can be found by following the frequency of an item by considering its both upward and downward paths.

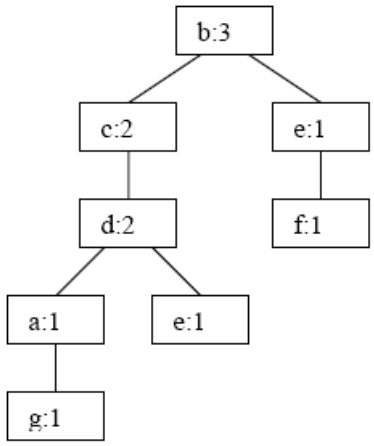

Fig. 8. CATS tree after 1st group of insertions.

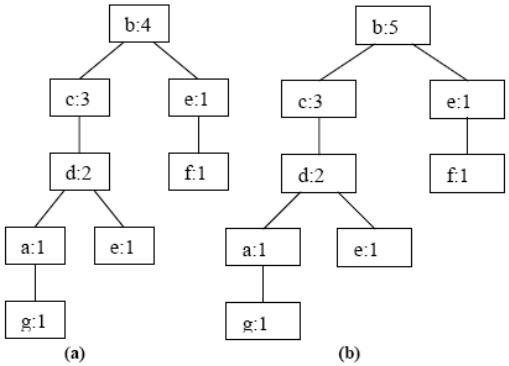

Fig. 9. CATS tree after 2nd group of insertions.

Advantages and Disadvantages

This method is advantageous because it requires only one scan of the database and the trees are ordered according to their local frequency in the paths. It is disadvantageous because lot of computation is required to build the tree.

CONCLUSION

In this paper, we analysed the various data structures that can be used to implement frequent pattern mining in large databases. We have discussed about the structures like CAN-tree, FP-tree, CATS-tree, and COFI with their merits and demeits. Among the discussion on the above structures, CAN-tree can be considered to be optimal because it scans the entire transactional database only once and there is no need for swapping the nodes in the tree. As well as, CAN-tree may be suitable for any incremental updates done in the database.

REFERENCES

S. M. Metev and V. P. Veiko, *Laser Assisted Microtechnology*, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

J. Breckling, Ed., *The Analysis of Directional Time Series: Applications to Wind Speed and Direction*, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," *IEEE Electron Device Lett.*, vol. 20, pp. 569–571, Nov. 1999.

M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in *Proc. ECOC'00*, 2000, paper 11.3.4, p. 109.

R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.

(2002) The IEEE website. [Online]. Available: http://www.ieee.org/

M. Shell. (2002) IEEE tran homepage on CTAN. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/supported/IEEEtran/

FLEXChip Signal Processor (MC68175/D), Motorola, 1996.

"PDCA12-70 data sheet," Opto Speed SA, Mezzovico, Switzerland.

A. Karnik, "Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP," M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1999.. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of TCP Reno congestion avoidance and control," Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.

Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std. 802.11, 1997.