Basic Framework of CATSIM Tree for Efficient Frequent Pattern Mining

SANJAY PATEL [1]
SANJAY GARG [2]

[1] Sankalchand Patel College of Engineering
Visnagar - 384315.
(sanjaypatel.ce@gmail.com)
[2] Computer Engineering Department, A.D. Patel Institute of Technology
New V. V. Nagar - 388121.
(gargsv@gmail.com)

Abstract. Finding frequent patterns from databases have been the most time consuming process in association rule mining. Several effective data structures, such as two-dimensional arrays, graphs, trees and tries have been proposed to collect candidate itemsets and frequent itemsets. It seems that the tree structure is most extractive to storing itemsets. The outstanding tree has been proposed so far is called FP-tree which is a prefix tree structure. Some advancement with this tree structure is called CATS tree. CATS Tree extends an idea of FP-Tree to improve storage compression and allow frequent pattern mining without generation of candidate itemsets. It allows the mining with a single pass over the database. In this work, CATSIM Tree is presented for which an attempt has been made to modify present CATS Tree in order to make it efficient for incremental mining.

Keywords: Data Minging, Frequent Pattern Mining, CATS Tree, FP-Tree, Incremental Mining.

(Received September 13, 2008 / Accepted January 19, 2009)

1 Introduction

Due to modern storage technologies advancement, it is possible to store a large amount of data cheaply, in both financial and physical sense. So, for companies it is feasible to record all kinds of data from customer's personal information to purchasing transactions. This leads to accumulation of huge amount of data.

Huge amount of data does not equate to that much amount of information and most of the data require large amount of processing before useful information can be extracted. The process of extracting hidden patterns from large datasets is called knowledge discovery.

One crucial phase of the knowledge process is data mining. Data mining is to find valid, novel, potentially useful and ultimately understandable patterns in data [2].

In general, there are many kinds of patterns that can be discovered from data. For example, association rules can be mined from market basket analysis, classification rules can be found for accurate classifiers, clusters and outliers can be identified for customer relation management [6].

Frequent pattern mining plays an important role in many data mining tasks, such as mining association rules, correlations, causality, sequential patterns, multi dimensional patterns, partial periodicity and emerging patterns. Information extracted from huge transactional datasets is generally represented in the form of association rules. Frequent itemsets are itemsets that have support greater than a minimum user defined support. Before association rules can be constructed, the frequencies of the underlying frequent itemsets have to be found.

The first and published data-mining algorithm is Apriori. It is based on the downward closure property of itemset that if an itemset of length k is not frequent, none of its superset patterns can be frequent. Before each data scan, candidate frequent item sets are verified whether they are frequent or not during the next data scan. There are many variations proposed to improve the efficiency of the Apriori algorithm, they are as 1) Hash-based techniques, 2) Transaction reduction, 3) Partitioning the data to find candidate itemsets etc [6].

The second class of algorithms is based on Frequent Pattern growth (FP-growth) which generates frequent patterns without candidate generation. FP-growth which adopts a divide and conquers strategy as follows: compress the database representing frequent items into a frequent pattern tree, or FP-tree, but retain the itemset association information, and then divide such a compressed database into a set of conditional databases, each associated with one frequent item, and mine each such database separately. There are some problems of the FP-growth algorithm like it requires two data scan to find frequent patterns [7].

The extension to the FP-growth algorithm is k-nown as CATS (Compressed Arranged Transaction Sequences) Tree, which solves the problem of FP-growth by reducing the number of data scan to one.

The contribution of this work is the development of a simple, but yet powerful, novel tree structure for maintaining frequent patterns found in the updated database. The CATS tree is for interactive mining, not for incremental mining. Attempt has been made to make the CATS Tree efficient for incremental mining.

This paper is organized as follows. Section II discusses about previous work related with the frequent pattern mining. Section III discusses the CAT-SIM Tree method. Section IV shows the experimental results, and Conclusion and future scope is discussed in section V.

2 PREVIOUS WORK

Given a user specified support threshold minsup, X is called frequent itemset if sup(X) , is greater than minsup. From frequent itemsets association rules can be derived. This section provides discussion of some existing algorithms for frequent itemset mining.

2.1 Apriori based Algorithms

The very first well known algorithm for frequent pattern mining is Apriori [6]. It works on the principle of candidate generation and test. It suffers from the following problems [7].

1. To handle a huge number of candidate sets is costly. For, example if there are 105 frequent 1-itemsets, the Apriori algorithm will need to generate more than 109 length-2 candidates and test their occurrence frequencies.

2. To frequently scan the database and check a large set of candidates by pattern matching is complex task.

2.2 Pattern Growth Methods

The FP-growth is pattern growth approach for frequent pattern mining. It removes the candidate generation and test approach. This data structure can be designed on the following observations.

1. Because only frequent items will play a role in the frequent pattern mining, it is required to perform one scan of transaction database to identify the set of frequent items.

2. If the set of frequent items of each transaction can be stored in some data structure, it may be

possible to avoid frequent scanning the original transaction database.

3. If multiple transactions share a set of frequent items, it may be possible to merge the shared sets with the number of occurrences registered as count.

Although FP-growth is more efficient than Apriori in many cases, it may still face problems in some cases as below [7]:

1. Space requirement is increased substantially. If the database is huge and sparse, the FP-tree will be large and the space requirement for recursion is a challenge.

2. Database contains all the cases. Real data sets can be sparse or dense in different applications.

3. More scalability is needed in large applications. Many existing methods are efficient when the data set is not very large.

2.3 CATS Tree and FELINE Algorithm

CATS Tree extends the idea of FP-Tree to improve the storage compression and allow frequent pattern mining without generation of candidate itemsets. The differences of FP-tree and CATS Tree are shown in the table 1.

The algorithm for the CATS Tree builder is briefly discussed in [10].The FELINE algorithm takes the CATS Tree as an input and mines the frequent items according to the algorithm given in [2]. However the CATS Tree and FELINE algorithm suffers from the following problems [10]:

1. It is expensive to build the tree without removing the items with frequency less than minimum support.

2. Swapping will take more times than the normal FP-Tree nodes.

CATS Tree	**FP-Tree**
Contains all items in every transaction	Contains only frequent items
Single Scan data mining	Two scan data mining
Items within a transaction do not need to be sorted	Items within a transaction are sorted
Sub-trees are locally optimized to improve compression	Sub-trees are not locally optimized

Table 1: Difference between the CATS Tree and FP-tree [10]

TID	Original Transaction
1	A,F,C,D,G,I,M,P
2	A,B,C,F,L,M,O
3	B,A,H,J,O
4	B,C,K,S,P
5	A,F,C,E,L,P,M,N

Table 2: Transaction Database

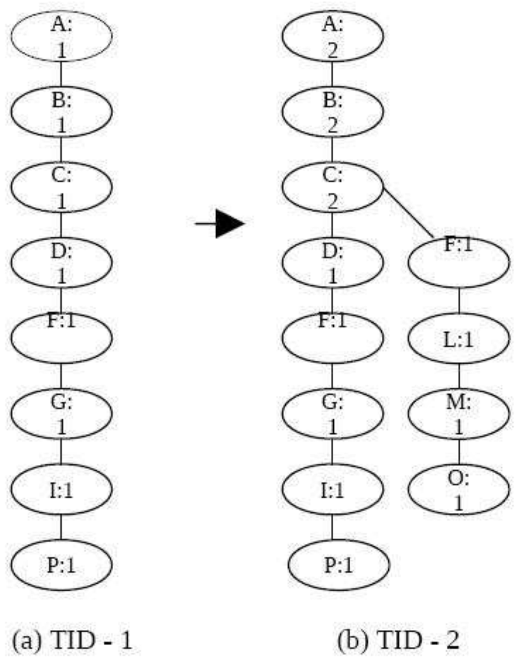
3 CATSIM TREE

CATS Tree is an extension of the FP-Tree, it is structurally similar, except branches in CATS tree are longer than those of FP-Tree. This is because CATS Tree contains all items in each transaction rather than just the frequent items. The CATS Tree satisfies the following properties.

1. The compactness of CATS Tree measures how many transactions are compressed at a node. The compactness of CATS Tree is the highest at the root and the compactness decreases as a node is further away from the root.

2. No item of the same kind, i.e., nodes containing the same item label, could appear on the lower right hand side of that level item.

To illustrate, how the CATSIM tree is working we will take the following database as an example. First all the elements of the transaction will be sorted according to the alphabetical order, and then the tree will be formed according to that order as shown in the figure 1.



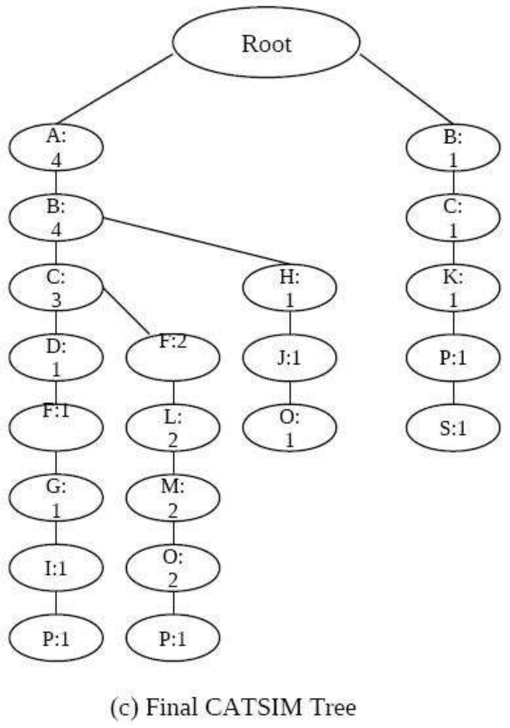


Figure 1: CATSIM Tree construction

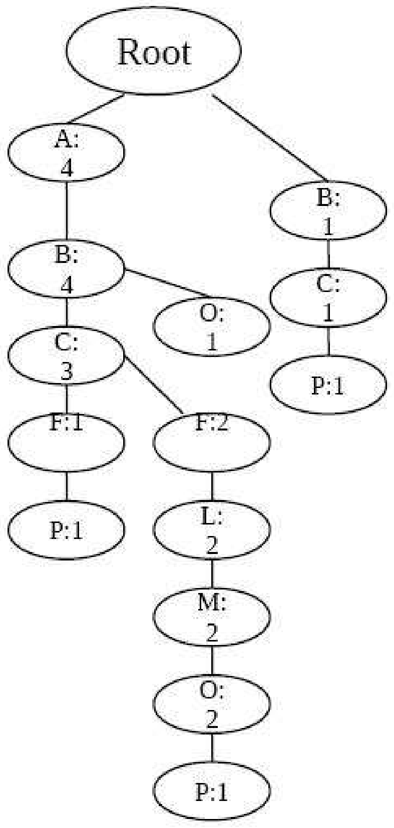


Figure 2: CATSIM Tree for minsup = 3

From the above tree, the frequency of all the elements will be found. Suppose minsup = 2 B : 5 , A : 4 , C : 4 , F : 3 , O : 3 , P : 3 , L : 2 , M: 2 , D : 1 , G : 1 , H: 1 , I: 1 , J: 1 , K : 1 , S : 1

Now the elements whose frequency is less than the minsup will be removed from the tree. So the new tree will be formed as shown in Figure 2. After that the mining of the frequent items will be done according to the FP-growth [10] approach as follows.

1. The Conditional pattern base will be formed according to the ascending order of the items.

2. Conditional FP-tree will be generated according to the same order as in step 1 by removing the items with the frequency less than the minsup from the conditional pattern base.

3. Finally frequent patterns will be generated from the conditional FP-tree.

Parameters	FP - Growth	CATSIM
Total no. of Trans	5577	5577
1-Itemsets	23	27
2-Itemsets	30	35
3-Itemsets	16	18
4-Itemsets	05	06

Table 3: Comparison of results

4 EXPERIMENTAL RESULTS

The implementation of the FP-growth algorithm is already available in Linux in [11]. The implementation of the CATSIM Tree and its algorithm is in Java. The testing of both of the algorithms are performed on Intel Core 2 Duo processor. Here the experimental results are shown in Table 3.

P The result table shows FP-growth algorithm is unable to find all the possible frequent itemsets. Implementation of CATSIM Tree reflects following advantages:

1. The construction of the proposed CATSIM Tree is independent of threshold values. 2. The items are alphabetically ordered in CATSIM Tree, any insertions, deletions and modifications of transactions have no effect on the ordering of items in the tree. So, swapping of tree nodes is not needed. 3. For CATSIM Tree, items are arranged to some order that is unaffected by the item frequency. So searching of the common items during the construction is easy. No extra downward traversals are needed during the mining process.

5 CONCLUSION AND FUTURE SCOPE

CATSIM tree provides efficient updated mining. The tree size will be exponential in the case of huge database, so to use disk based memory rather than main memory may be future work. The CATSIM, its efficient algorithm and the formal theoretic description of the same to support the advantages can also be a further work.

References

[1] Alva Erwin, Raj P.Gopalan, N.R.Achuthan,“*CTU-Mine: An efficient High Utility Itemset Mining Algorithm using Pattern growth approach*," Seventh International conference on Computer and Information Technology,2007.

[2] Cheung W., “*Frequent Pattern mining without candidate generation or support constraint*," Master’s thesis, University of Alberta, 2002, SPRING ’03,doi.ieeecomputersociety.org /10.1109/IDEAS.2003.1214917.

[3] Christian Borgelt, “*An Implementation of the FP-growth Algorithm*," OSDM’05, August 21, 2005, Chicago, Illinois,USA,www.cs.rpi.edu/ zaki/OSDM05/ papers/p1-borgelt.pdf.

[4] http://fimi.cs.helsinki.fi/data/

[5] http://www.almaden.ibm.com/cs/quest// syndata.html#assocSynData

[6] Jiawei Han and Micheline Kamber, “*Data Mining, Concept and Techniques*," 578 pages, books.google.co.in.

[7] Jien Pei, “*Pattern-Growth methods for frequent pattern mining,* " Ph. D. Thesis, Simon Fraser University, 2002, ftp://fas.sfu.ca/pub/cs/theses/2002/ JianPeiPhD.pdf.

[8] Q. I. Khan, T. Hoque and C.K. Leung, “*CanTree : A Tree structure for Efficient Incremental mining of Frequent Patterns,*" Proceeding of the Fifth International Conference on Data Mining (ICDM’05),

[9] Rajnish Dass and Ambuj Mahanti, “*An efficient heuristic search for Real-Time frequent pattern mining*," International Conference on System Sciences - 2006, ieeexplore.ieee.org/iel5/10548/33362/01579371.pdf

[10] William Cheung and Osmar R. Zaiane, “*Incremental Mining of Frequent Patterns without candidate Generation or Support Constraint*," IDEAS’03, doi.ieeecomputersociety.org/10.1109/IDEAS, 2003,1214917.

[11] www.cse.cuhk.edu.hk/ kdd/program.html.