ISSN 2319-2720

Volume 1, No.1, July – August 2012
International Journal of Computing, Communications and Networking
Available Online at http://warse.org/pdfs/ijccn08112012.pdf

Analysis and Implementation of FP & Q-FP tree with minimum CPU utilization in association rule mining

Yasha Sharma
M.Tech (Software System), SATI, Vidisha (M.P)
yashasharma78@gmail.com
Dr. R.C. Jain
Dept. Of Computer Application, SATI, Vidisha (M.P)

ABSTRACT

Association rule mining, one of the most important and well researched techniques of data mining, was first introduced in. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. However, no method has been shown to be able to handle data streams, as no method is scalable enough to manage the high rate which stream data arrive at. More recently, they have received attention from the data mining community and methods have been defined to automatically extract and maintain gradual rules from numerical databases. In this paper, we thus propose an original approach to mine data streams for Association rule mining. Our method is based on Q-Based and FP growth in order to speed up the process. Q-based are used to store already-known for order to maintain the knowledge over time and provide a fast way to discard non relevant data while FP growth.

1. INTRODUCTION OF FP GROWTH

The problem of mining association rules from a data stream has been addressed by many authors but there are several issues (as highlighted in previous sections) that remain to be addressed. In the following section existing literature based on the problems in data stream mining that is addressed. The work in this domain can be effectively classified into three different domains namely, *exact methods for Frequent Item set Mining*, *Approximate Methods* and *Memory Management* techniques adopted for data stream mining [1].

2. BACKGROUND OF STUDY

Frequent-pattern mining plays an essential role in mining associations [1] *if any length k pattern is not frequent in the database, its length* (k + 1) *super-pattern can never be frequent.* The essential idea is to iteratively generate the set of candidate patterns of length (k+1) from the set of frequent-patterns of length k (for $k \geq 1$), and check their corresponding occurrence frequencies in the database.

The *Apriori* heuristic achieves good performance gained by (possibly significantly) reducing the size of candidate sets. However, in situations with a large number of frequent patterns, long patterns, or quite low minimum support thresholds, an *Apriori*-like algorithm may suffer from the following two nontrivial costs: – It is costly to handle a huge number of candidate sets. For example, if there are 104 frequent 1-itemsets, the *Apriori* algorithm will need to generate more than 107 length-2 candidates and accumulate and test their occurrence frequencies. Moreover, to discover a frequent pattern of size 100, such as {$a1. \ . \ . \ a100$}, it must generate $2100 - 2 \approx 1030$ candidates in total.

This is the inherent cost of candidate generation, no matter what implementation technique is applied. It is tedious to repeatedly scan the database and check a large set of candidates by pattern matching, which is especially true for mining long patterns. Can one develop a method that may avoid candidate generation-and-test and utilize some novel data structures to reduce the cost in frequent-pattern mining? This is the motivation of this study [5].

In this work, we develop and integrate the following three techniques in order to solve this problem. First, a novel, compact data structure, called frequent-pattern tree, or FP-tree in short, is constructed, which is extended prefix-tree structure storing crucial, quantitative information about frequent patterns. To ensure that the tree structure is compact and informative, only frequent length-1 items will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of node sharing than less frequently occurring ones.

Subsequent frequent-pattern mining will only need to work on the FP-tree instead of the whole data set. Second, an FP-tree-based pattern-fragment growth mining method is developed, which starts from a frequent length-1 pattern (as an initial

suffix pattern), examines only its conditional-pattern base (a "sub-database" which consists of the set of frequent items co-occurring with the suffix pattern), constructs its (conditional) FP-tree, and performs mining recursively with such a tree[5][6]. The pattern growth is achieved via concatenation of the suffix pattern with the new ones generated from a conditional FP-tree.

Concept Hierarchy:

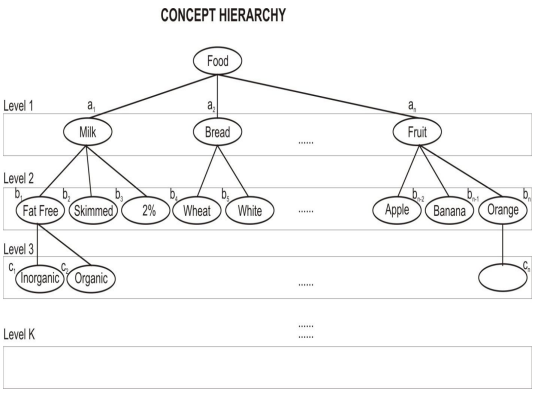


Figure 1: Concept hierarchy

3. DATA PREPARATION

The data of association rule mining is used for finding the frequent itemset and closed frequent itemset. The click stream is a sequence of mouse click made by every user. The transactions are generated by eliminating noisy, and very short or very long access sequences. The dataset from www. fimi.com.

3.1 Introduction

Association Analysis is the discovery of association rules attribute-value conditions that occur frequently together in a given data set. Association analysis is widely used for market basket or transaction data analysis. Association Rule mining techniques can be used to discover unknown or hidden correlation between items found in the database of transactions. An association rule [1, 3, 4, and 7] is a rule, which implies certain association relationships among a set of objects (such as 'occurs together' or 'one implies to other') in a database. Discovery of association rules can help in business decision making, planning marketing strategies etc.Apriori was proposed by Agrawal and Srikant in 1994. It is also called the level-wise algorithm. It is the most popular and influent algorithm to find all the frequent sets. The mining of multilevel association is involving items at different level of abstraction. For many applications, it is difficult to find strong association among data items at low or primitive level of abstraction due to the sparsity of data in multilevel dimension. Strong associations discovered at higher levels may represent common sense knowledge. For example, instead of discovering 70% customers of a supermarket that buy milk may also buy bread. It is also interesting to know that 60% customer of a super market buys white bread if they buy skimmed milk. The association relationship in the second statement is expressed at lower level but it conveys more specific and concrete information than that in the first one. To describe multilevel association rule mining, there is a requirement to find frequent items at multiple level of abstraction and find efficient method for generating association rules. The first requirement can be fulfilled by providing concept taxonomies from the primitive level concepts to higher level. There are possible to way to explore efficient discovery of multiple level association rules. One way is to apply the existing single level association rule mining method to mine Q based association rules. If we apply same minimum support and minimum confidence thresholds (as single level) to the Q levels, it may lead to some undesirable results. For example, if we apply Apriori algorithm [1] to find data items at different level of abstraction under the same minimum support and minimum confidence thresholds. It may lead to generation of some uninteresting associations at higher or intermediate levels.

1. Large support is more likely to exist at high concept level such as bread and butter rather than at low concept levels, such as a particular.

3.2 Fundamental of Q- based FP Tree

The previous chapters have described the fundamental background behind closed item set mining, work objectives, overall architecture, and experimental design. This chapter will focus on the experimental findings. Both Q-based and FP growth were tested on synthetic datasets and compared against predefined performance metrics such as Accuracy, computational performance, and Memory consumption. Supposed our Database is given in this format.

4. PROPOSED ALGORITHM

Algorithm 1: **FP tree**

Step1: Start for finding the frequent item set
Step2: Arrange them according to the base ascending or descending order
Step3: Put minimum support 4 in the data base: figure ('Name',' Elements with minimum value is >4')
Step4:Suitable('Data',strcat(str,'=',num2str(strc)),'Units','pixels','Position',[20,0,390,160]);
Step5:hp=ipanel ('Title',' Elements with minimum value is >4','TitlePosition','centertop','FontSize', 20);
Step6: FPTree; time (1) =toc tic
Step 7: Calculate the cpu time for FP tree.

```
     ylabel ('CPU Time in second'); set
(gca,'XTickLabel',{'FP
Tree','Q_based_FP_Tree'});
```

Descriptions of Algorithm1:

Step1.Data set: We apply data set
Step2. First we apply FP tree in this Dataset and then find the frequent itemset at min <4%
Step3. Element with minimum support Value<4

Step4.Form a FP-tree.

Example of Algorithm 1:

Data set:

	1	2	3	
1	A	G	D	C
2	B	C	H	E
3	B	D	E	A
4	C	E	F	A
5	A	B	N	O
6	A	C	Q	R
7	A	C	H	I
8	L	E	F	K

Table1: Find the frequent itemset at min <4%

	1
1	A=11
2	B=10
3	C=10
4	D= 9
5	E= 8
6	F= 7
7	G= 5
8	H= 3
9	I= 3
10	J= 2
11	K= 3

Table 1.1 Element with minimum support Value<4%

	1
1	A=11
2	B=10
3	C=10
4	D= 9
5	E= 8
6	F= 7
7	G= 5

FPGrowth.

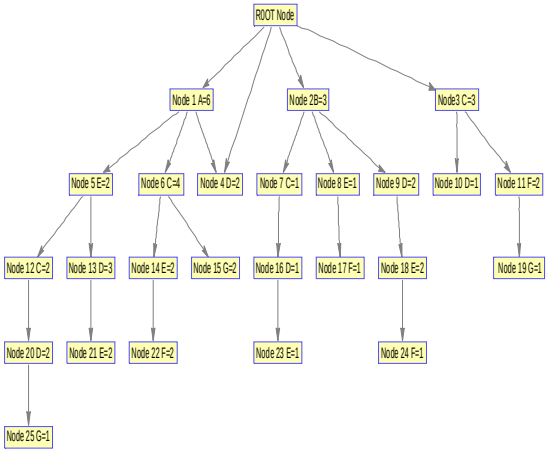


Figure 2: FP tree.

Algorithm 2: Q_based_FP_Tree

Step1: Start for finding the Frequent item set
Step2.Arrange them in Table format no need to arrange in ascending or descending order because it is Q based technique. So the element come first they serve first
Step3: Put minimum support 4 in the data
Base: figure ('Name',' Elements with minimum value is >4')
Step4:Suitable('Data',strcat(str,'=',num2str(strc)),'Units','pixels' ,'Position',[20,0,390,160]);
Step 5: hp =ipanel ('Title',' Elements with minimum value is >4','TitlePosition','centertop','FontSize', 20);
Step6: QFPTree; time (1) =toc tic; basedFPTree; time (2) =toc; Figure (); bar (time,'g');
Step 7: Calculate the cpu time for FP tree.

```
ylabel('CPU Time in second');
set(gca,'XTickLabel',{'FP
Tree','Q_based_FP_Tree'});
```

Descriptions of Algorithm 2:
Step1: Find the Frequent item set.
Step2.Arrange them in Table format , the element come first will serve first.
Step3: Put minimum support 4 in the data
Step 4: Form a Q Based tree.

Example of Algorithm 2:

Data set:

	1	2	3	
1	A	G	D	C
2	B	C	H	E
3	B	D	E	A
4	C	E	F	A
5	A	B	N	O
6	A	C	Q	R
7	A	C	H	I
8	L	E	F	K

Element with minimum support Value<4%

	1
1	A=11
2	B=10
3	C=10
4	D= 9
5	E= 8
6	F= 7
7	G= 5

Q-based FP-Tree

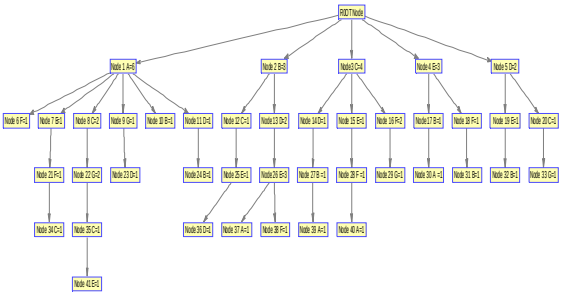


Figure 3 :Q based tree

4.2 Findings from Experiment 1

This experiment was mainly designed for comparing Data structure using Q FP-Tree and FP-Tree with respect to performance. We first varied the minimum support threshold while keeping the delta parameter constant. We recorded the accuracy, performance and memory consumption for Data structure and then repeated the procedure for FP tree. For this experiment, we have used dense datasets generated using the IBM data generator (IBM). The Recall and Precision were calculated by comparing Data structure using FP Tree and FP tree results against the Apriority implementation process is repeated at time Ts3 with tuple T3 checking that

5 SOFTWARE EVALUATIONS

We have taken the 200 data set and make the frequent itemset for Q-based_FP-Tree approach. Code is implemented in Mat lab.

5.1 Explanation

The discrenibility matrix corresponding to the sample database shown in Table 1 with

Itemset={1,2,3,4,5….},C={A,B,C,D,E}APPLY 4%
SUPPORT than L={A,B,C,D,E,F,G}={11,10,10,9,8,7,5}

5.2 Dataset Selection

The datasets used in this paper is used by various Data mining experts in their research. The elements in the datasets which are represented in the numeric format, easy to evaluate the processes, which involved in those mining concepts. These datasets consists of frequent itemset in each record level. In record level they are separated by special identification. The elements are separated by space. The original values of Mushroom and Connect datasets observations represented by its index values using mining concepts. The frequent items and its associative datasets are easy to calculate and represented as a flat file (or) text file. The dataset which are used for the evaluation contains following characteristics.

Table 1.3

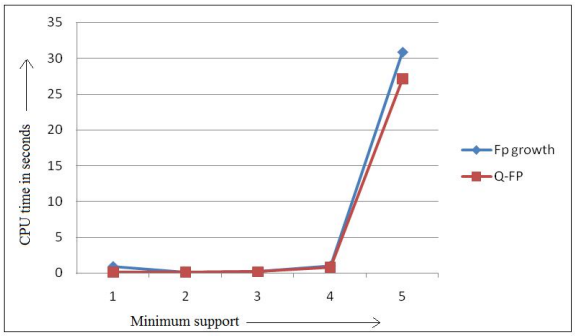
DATA	# Item	Transaction
Mushroom	120	23
Connect	30	43

5.3. Result and Comparison graph

Under large minimum supports, FP-Growth runs faster than FP-Graph while running slower under large minimum supports. Fig 2 and 3 show what minimum support used in experiments. Both algorithms adopts a divide and conquer approach to decompose the mining problem into a set of smaller problems and uses the frequent pattern (FP-tree) tree and (QFP) data structure to achieve a condensed representation of the database transactions. Under large minimum supports, resulting tree and graph in relatively small size so with this condition FP-Q does not take advantages of small memory space and also page fault for both algorithm is almost equal. But as minimum supports decrease resulting data structure size rapidly increase, it require more memory space , at this point advantage of FP-Q come in existence with less page fault FP-Q considerable work well with high dense database along with small minimum supports, it shown in Fig. 2 and 3 . Response time FP Growth tree good but total run time for large database, FP-Q good because it gives less page fault. FP Growth Tree uses tree for arranging the items before mining, where more than one node can contain single item. This causes repetition of same item and needs more space to store many copies of same item.

Table 1.4

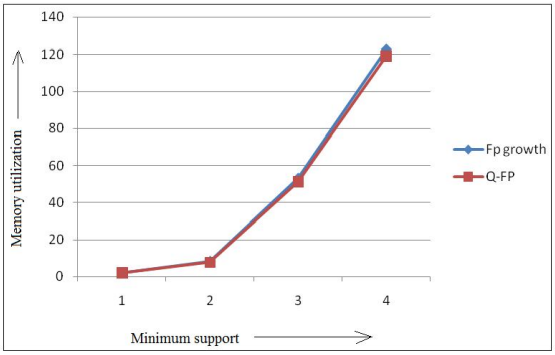
Support	FP Growth	Q-FP
90	0.93	0.45
70	0.109	0.124
30	0.187	0.179
15	1	.89
5	30.89	27.11



Graph 1: CPU utilization using fP growth and Q -FP tree

Table 1.5

Support	FP Growth	Q-FP
90	0.93	0.45
70	0.109	0.124
30	0.187	0.179
15	1	.89
5	30.89	27.11



Graph 2: Memory utilization of FP growth and Q FP tree

6. CONCLUSION

Data stream mining is one of the most intensely investigated and challenging work domains in contemporary work in the data mining discipline as a whole. The peculiarities of data streams render conventional mining schemes inappropriate.

In this dissertation we used novel approach for mining the closed item set from a Data stream. We have implemented Q based-tree to store the closed item set with their support count for this we use Apriori principal to reduce the unnecessary power set creation and prune closed item set with frequent item set. Proposed work develops an incremental frequent item set mining Algorithm based on the Data stream.

The Data Stream can find the lot of data in data set. We compare Q based-tree with FP tree. Our Experiment shows that Q based-Tree not only outperformed FP growth but it provides the short time for pruning the frequent item set.

In this work, we presented an overview of a novel approach for mining the frequent item sets from a data stream. We have implemented an efficient closed prefix Q based-tree to store the intermediate support information of frequent item sets.

REFERENCE

1. R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. **Depth first generation of long patterns**, KDD'00, pp. 108–118, 2000.

2. R. Agrawal, T. Imielinski, and A. N. Swami. **Mining association rules between sets of items in large databases**, ACM SIGMOD'93, pp. 207–216, Washington, D.C.1993.

3. R. Agrawal and R. Srikant. **Fast algorithms for mining association rules**, VLDB'94, pp. 487–499, 1994.

4. R. Agrawal and R. Srikant. **Mining sequential patterns**, ICDE'95, pp. 3–14, 1995.

5. B. Goethals and M. J. Zaki. **Advances in frequent itemset mining implementations: Introduction to fimi03,** Proceedings of the 1st IEEE ICDM Workshop on Frequent Item set Mining Implementations (FIMI'03), Nov 2003.

6. G. Grahne and J. Zhu. **Efficiently using prefix-trees in mining frequent item sets,** 1st IEEE ICDM Workshop on Frequent Item set Mining Implementations (FIMI'03), Nov 2003.

7. J. Han, J. Pei, Y. Yin, and R. Mao. **Mining frequent patterns without candidate generation: A frequent-pattern tree approach**, Data Mining and Knowledge Discovery, 8:53– 87, 2004.

8. M. Kamber, J. Han, and J. Chiang. **Met rule-guided mining of multi-dimensional association rules using data cubes**, Knowledge Discovery and Data Mining, pp. 207–210, 1997.

9. H. Mannila, H. Toivonen, and A. I. Verkamo. **Discovery of frequent episodes in event sequences**, Data Mining and Knowledge Discovery, 1(3):259–289, 1997.

10. Savasere, E. Omiecinski, and S. B. Navathe. **An efficient algorithm for mining association rules in large databases,** VLDB'95, pp. 432–444, 1995.

11. H. Toivonen. **Sampling large databases for association rules,** VLDB'96, pp. 134–145, Sep. 1996.

12. M. Zaki and K. Gouda. **Fast vertical mining using diffsets**,ACM SIGKDD'03, Washington, DC, Aug. 2003.

13. Claudio Lucchese. **Mining frequent closed itemsets out of core**, 2004