IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 6, Ver. IV (Nov.-Dec. 2016), PP 109-114
www.iosrjournals.org

Mining Frequent Itemsets Along with Rare Itemsets Based on Categorical Multiple Minimum Support

Md. Fazla Elahe[1], Kun Zhang[1]
[1]*(School of Computer Science and Engineering, Nanjing University of Science and Technology, China)*

Abstract: *Mining interesting itemsets is one of the key tasks in data mining. In real world applications both frequent itemsets and rare itemsets might be interesting. Frequent itemsets are the itemsets that occur with a frequency greater than user specified threshold. Sometimes we need to mine frequent itemsets along with rare itemsets. It is computationally expensive and unwise to mine all the itemsets form the database. Researchers have proposed several algorithms to mine interesting itemsets (both frequent and rare). But it is still challenging to mine frequent itemsets with rare itemsets because if we set a low minimum support to mine rare itemsets then a lot of uninteresting itemsets may be mined, if we set a high minimum support then we may not get the interesting rare itemsets. Using multiple minimum support is somehow solve this issue. But setting minimum support to each and every item is somehow tricky. In this paper we propose an algorithm named categorical multiple minimum support apriori (C-MSApropri) that uses two category to mine frequent itemsets along with rare itemsets. Experiment results show that the proposed algorithm is very effective.*
Keywords : *Frequent itemset mining; Multiple minimum support; Rare itemset, C-MSApriori*

I. Introduction

Since R. Agrawal et. al have introduced the idea of frequent itemsets mining [1], lots of problems have come into front. Several approaches have been proposed in the meantime to solve these problem. Most of the frequent itemset mining algorithms like Apriori [2] and Fp-growth [3] use support confidence framework for mining frequent patterns. If we say about frequent pattern there is a question to arise that how frequent is the pattern. The simplest answer is if a pattern satisfies the user specified threshold then we call that pattern as a frequent pattern. The frequent itemsets should satisfy user specified minimum support (minsup).

Example 1: Let's consider a transaction database with 10 transactions (TABLE I). The itemset (Mouse, Flash) occurs 6 times in the dataset. Therefore support S(Flash ∪ Mouse)= (6*100)/10=60%. The itemset (Flash, Mouse) that will be frequent if and only if the support is less than or equal 60%. If minsup is 50% then the itemest (Flash, Mouse) is frequent itemset.

Table I. Transactional Dataset

Transactions	Items
T1	Keyboard, Cable
T2	Mouse, Memory, Flash
T3	Keyboard, Flash
T4	Mouse, Keyboard, Flash
T5	Memory, Flash
T6	Mouse, Memory, Flash
T7	Mouse, Laptop, Flash
T8	Mouse, Keyboard, Flash
T9	Mouse, Cable, Flash
T10	Mouse, Keyboard

Most of the real-world datasets are not uniform. From the above example it is clear that the number of frequent itemsets depends on minimum support. If we set a low minimum support we will get more frequent items and if we set minimum support to a high value we will get less frequent itemsets. If we want to mine rare itemsets we have to set low minimum support. In that case we may get some uninteresting itemsets. If we set high minimum support some rare itemsets may miss to mine. This dilemma is called the rare item problem [4].

Example 2: Let's consider candidate table (table II & III) generated from table I. If we want to mine rare itemset for an example (laptop, flash) then we need to set minimum support to 10%. But in that case we will get an itemset (Mouse, Cable), which is not interesting. If we set minimum support to 30% then we will get all the interesting itemset but we will not get rare (Laptop, Flash) itemset.

Table II. Candidate 1 Itemsets

Item	Support
Mouse	70%
Keyboard	50%
Memory	30%
Cable	20%
Laptop	10%
Flash	80%

Table III. Candidate 2 Itemsets (10% Minsup)

Item	Support
Mouse, Keyboard	30%
Mouse, Memory	20%
Mouse, Cable	10%
Mouse, Laptop	10%
Mouse, Flash	60%
Keyboard, Cable	10%
Keyboard, Flash	30%
Memory, Flash	30%
Cable, Flash	10%
Laptop, Flash	10%

To overcome the problem mentioned above, several research papers have been published. Most of the approach use multiple minimum support. But still they suffer from a few problems. Real life frequent pattern mining algorithms need to adjust minimum support for getting required itemsets. For algorithms like MSApriori if we want to get our required result we need to reassign minimum support to each and every items in the database which might be tricky and hard to determine the exact support for each and every items. Our proposed algorithm uses only three different values for getting frequent itemsets along with the rare itemsets. By using only three minimum support gives user flexibility to adjust minimum support in a quick sensation.The remaining part of the paper is organized as follows: in section II, we will summarize the existing approaches that deal with mining frequent itemsets. In section III, we will discuss our proposed approach. Experimental results on a synthetic dataset are discussed in section IV. In the last section we will discuss conclusion.

II. Related Work

Since the introduction of frequent itemsets mining [1] it has become an interesting research area and studied widely [5-15]. Finding interesting itemsets is the main focus of these algorithms. Hipp, J. et al. published a nice survey article on association rule mining algorithms [16]. In this section we will discuss some related approaches.

A. Apriori algorithm

Apriori algorithm [2] is the most widely studied algorithm in the area of frequent itemset mining. Apriori uses single minimum support for all the items to extract frequent itemsets. It employs interactive level wise search for generating frequent itemset. The support of frequent itemsets should be greater than or equal to user specified minimum support (minsup). An itemset having K number of itemsets is called a K itemset candidate and k-frequent itemsets refers to subset of candidate k-itemsets having support greater than or equal to user defined minsup which is represented by Lk. Apriori algorithm starts from K=1 till no more frequent itemsets are found. Apriori algorithm follows following steps:

1. C1 is generated from the dataset.
2. Items that satisfy minsup from C1 is used to generate L1.
3. Ck is generated from LK-1 by joining LK-1 with itself.
4. LK is generated from CK that satisfies minimum support & respect downward closure property (every subset of a frequent itemset is frequent).

B. MSApriori (Multiple minimum support Apriori)

Real world datasets are not uniform. Items in the database may have different importance. That's why it is necessary to use multiple minimum support to overcome rare item problem. MSApriori [5] is an extension of Apriori algorithm, uses multiple minimum supports for the items in dataset. The main philosophy of this algorithm is to assign minimum item support (MIS) to every item and generate frequent itemsets based on their MIS values. MIS for a data item i expressed as MIS(i) calculated from the following formula.

$$MIS(i) = \begin{cases} M(i) & M(i) > LS \\ LS & Otherwise \end{cases}$$

$M(i) = \beta f(i)$

Where f(i) is the actual frequency of the item i in the dataset. MSapriori takes two parameters from the user to calculate MIS of an item. One is least support (LS) and other is β. Parameter β takes a value from 0 to 1 to control the MIS values. There is a problem to satisfy downward closure property if we want to use multiple minimum support. According to downward closure property, every subset of a frequent itemset is frequent. If we use this property for pruning then frequent itemset may be discarded as we have multiple minimum support. To overcome this problem B. Liu, W. Hsu, and Y. Ma proposed sorting itemsets according to their MIS values [5] and such itemsets satisfy the sorted closure property.

C. Relative support Apriori algorithm (RSAA)

Another approach for generating both frequent and rare pattern are relative support Apriori algorithm (RSAA) [6]. RSAA uses three user specified supports called 1st support, 2nd support and relative support (RSUP). If the support of an item is greater than or equal to 1st support then it is called frequent item. If the support of an item is less than 1st support but greater than or equal to 2nd support then it is called rare item. Itemsets having only frequent items have to satisfy only 1st support to be frequent on the other hand, itemsets containing rare items have to satisfy 2nd support and its relative support (RSUP) should satisfy minimum relative support (mRSUP) specified by the user. Relative support is a value between 0 and 1 and is determined by the following formula:

RSUP(i1, i2, i3,…,ik) = max(sup(i1,i2,i3,..,ik) / sup(i1), sup(i1,i2,i3,..,ik) / sup(i2), ……., sup(i1,i2,i3,..,ik) / sup(ik)) Where {i1, i2, i3....., ik} is an itemset and support of item i is expressed as sup(i).

III. Proposed Approach

Our proposed algorithm takes three user parameter for mining frequent itemsets along with rare itemsets.

Frequent minimum support: Frequent minimum support is a parameter that our algorithm takes from the user. It is a threshold value. Itemsets having support greater than this value are called frequent itemsets. It is denoted as FMS.

Support difference: Support difference is denoted by SD and it's a value that makes the difference between frequent itemsets and rare itemsets. It's a value that is always less than frequent minimum support but greater than rare minimum support.

Rare minimum support: Rare minimum support is a parameter that our algorithm takes from the user. It is a threshold value. Itemsets having support less than support difference but greater than rare minimum support are called rare itemsets. The value of rare minimum support should be less than support difference. It is denoted as RMS.

A. Problem definition

For a given dataset D, along with a frequent minimum support (FMS), support difference (SD) and rare minimum support (RMS) our goal is to find all the frequent itemsets and rare itemsets.

B. Basic idea

Our proposed algorithm adopts both Apriori and MSApriori. We call our algorithm C-MSAPriori (categorical multiple minimum support Apriori). Like Apriori and MSApriori, our algorithm is based on level-wise search and multiple pass-over the data is required to generate all the large itemsets. In this approach we have used three different thresholds to prune the uninteresting itemsets. First one is frequent minimum support (FMS). If the support of an item is greater than or equal to FMS then we call that item as a frequent item. Second one is support difference (SD). It is used as a gap between frequent itemsets and rare itemsets. The last parameter called rare minimum support (RMS). If the support of an item is less than the SD but greater than or equal to RMS then we call it as a rare item. The main advantage of this algorithm is that there will be no uninteresting itemsets while we mine some rare itemsets. At the same time we just need to tune just three parameters to get our required result while the traditional frequent itemsets mining algorithms need to tune the minimum support for each and every items in the database.

C. Algorithm: C-MSApriori

Let's consider a transaction database D with t transaction. Lk denotes the set of large K itemsets and CK is candidate set generated from Lk-1 where K>2. Each itemset A is of the following form <A[1], A[2],……., A[k]>, which consists of the items a1, a2,….,ak. The algorithm categorical multiple minimum support Apriori (C-MSApriori) is given bellow

```
1.    generate candidate 1 itemset $C_1$;  /* get all the item name from the dataset and store it to $C_1$*/
2.    for each item in $C_1$ do
3.    if sup(a) $\geq$ FMS then
4.    MIS(a) = FMS
```

```
5.    insert a into $C_F$;
6.    if sup(a) < SD then
7.    MIS(a) = RMS;
8.    insert a into $C_R$;
9.    else
10.   MIS(a) = FMS
11.   end
12.   $L_1$=$C_F$+$C_R$;
13.   $L_1$ = sort ($L_1$, MIS); /* sort $L_1$ according to their MIS value */
14.   SMS = sort ($C_1$, MIS); /* sort $C_1$ according to their MIS value */
15.   F = F-gen (SMS, D);
16.   for (k=2; $L_{k-1} \neq Ø$; k++) do
17.   if k=2 then $C_2$ = level 2-candidate gen(F);
18.   else $C_k$=$C_k$ candidate gen($L_{k-1}$);
19.   for each transaction $t \in D$ do
20.   $C_t$ = subset($C_k$, t);
21.   for each candidate $A \in C_t$ do
22.   A.count++;
23.   end
24.   end
25.   $L_k = \{ A \in C_k \mid A.count \geq MIS(A[1])\}$;
26.   end
27.   Answer $L_k$=$U_kL_k$
```

Line 1 gets all the unique item from the dataset and store it to C1. From line 2 to line 11 it assign MIS value to all items. Large 1-itemsets (L1) is obtained by adding frequent items (CF) and selected rare items (C_R). Level 2-candidate generation is different from any other level. We will discuss about it later. If k>2 then C_k is generated using C_k candidate gen function. Function subset (C_k, t) does pretty much same as [1] except it requires the items in each transaction t to be sorted according to their MIS values in ascending order so that it achieves sorted downward closure property.

Example 3: Let's continue with example 2. If we assign 60% to FMS, 20% to SD and 10% to RMS then we will get 2 frequent item and 1 rare item (table II). That is how we can mine rare interesting item laptop along with frequent interesting items flash and mouse without getting uninteresting items keyboard, memory and cable.

F-gen is a function used for counting actual support and items having support greater than their MIS and items having support greater than the MIS of previous item(in the same order) are used to store in F. The algorithm of F-gen is given bellow.

F-gen algorithm:

```
1.    for each transaction $t \in D$ do
2.    for each item $a \in t$ do
3.    a.count++;
4.    end
5.    end
6.    for each item a in SMS in the same order do
7.    if sup(a) ≥ MIS(a) then
8.    insert a into F;
9.    end
10.   for each item $a_2$ in SMS that is after $a_1$ in the same order do
11.   if sup($a_2$) ≥ MIS($a_1$) then
12.   insert $a_2$ into F;
13.   end
```

Level 2 candidate generation takes F as the only argument to generate superset of the set of all large 2 itemsets. We have to use frontier set F rather than L1 because F may contain items that satisfy minimum support threshold at the same time L1 may not contain those items. This occurs because we are using multiple minimum support and sorted closure property for pruning. If we consider candidate 2-itemset <a1,a2> generated from L1 then a1 and a2 both have to be listed in L1 that means both have to be frequent. But if we consider that a1 is not frequent but a2 is frequent and <a1, a2> is frequent then we can not generate <a1, a2> from L1. So if we pass L1 as an argument then our algorithm may fail to generate frequent itemsets in C_2. The algorithm is given bellow:

Level 2 candidate generation:

```
1.  for each item a ∈ F in the same order do
2.  if sup(a_1) ≥ MIS(a_1) then
3.  for each item a_2 ∈ F that is after a_1 do
4.  if sup(a_2) ≥ MIS(a_1) then
5.  insert < a_1, a_2 > into C_2;
6.  end
7.  end
```

Candidate generation function performs two tasks: joining and pruning. It joins L_{k-1} with L_{k-1} (k>2) in a similar way like apriori-gen. After joining there may still have some itemsets that are impossible to be large. Pruning task is used to remove those itemsets. The algorithm is given bellow.

C_k candidate generation:

```
1.  C_k = apriori-gen(L_{k-1})   /*Joining of L_{k-1} with L_{k-1} follows same procedure like apriori-gen*/
2.  for each itemset A ∈ C_k do
3.  for each (k-1) subset s of A do
4.  if(a_1 ∈ s) or (MIS(a_2))=MIS(a_1) then
5.  if (s ∈ L_{k-1}) then
6.  delete A from C_k;
7.  end
8.  end
```

After joining of every L_{k-1} with L_{k-1} itemset according to the function apriori-gen it checks each itemset A in Ck to see whether it can be deleted by finding its (k-1) subsets in L_{k-1}. For each (k-1) subset s in itemset A, if s is not in Lk-1 then we can delete A from Ck.

IV. Evaluation

A. Experimental details

We have used a synthetic dataset to evaluate the performance of our proposed approach. The synthetic dataset is T10I4D100K containing 870 items and 100K transactions. The details of the dataset is given in the table IV.It is hard to compare the performance between the existing algorithm and our proposed algorithm because existing algorithms takes two parameter while our proposed algorithm takes three parameters. we develop our own tool for evaluating performance of C-MSApriori. For simplicity we fixed the value of SD to 200 and RMS to 175 for C-MSApriori. With these constraints we get C-MSApriori in both time domain and candidate domain.

Table IV. Dataset Description

Name of the dataset	T10I4D100K
Dataset type	Synthetic
Number of items	870
Number of transaction	100000
Minimum item frequency	0.001%
Maximum item frequency	7.8%
Average item frequency	0.11%

B. Performance result

In the performance graphs (fig. 1, fig. 2), X axis is used for minimum support and Y axis is used for time (fig. 2), candidate (fig. 1). From the graphs it is clear that the number of generated candidate and required time for execution depends on the value of minimum support which is called frequent minimum support (FMS) for the algorithm C-MSApriori. But the problem occurs when we want to mine some rare itemsets using MSApriori. To do so we have to set the value of LS and β to a small value. Low value of LS and β force MSApriori to mine some uninteresting patterns along with interesting rare pattern. On the other hand our proposed algorithm C-MSApriori does not generate any uninteresting itemsets while it mines some rare itemsets along with frequent itemsets.

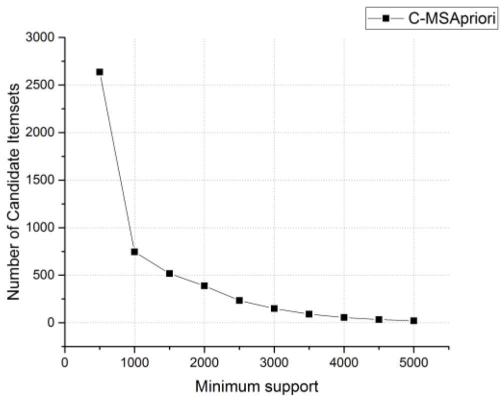

Fig. 1. Candidate itemsets C-MSApriori

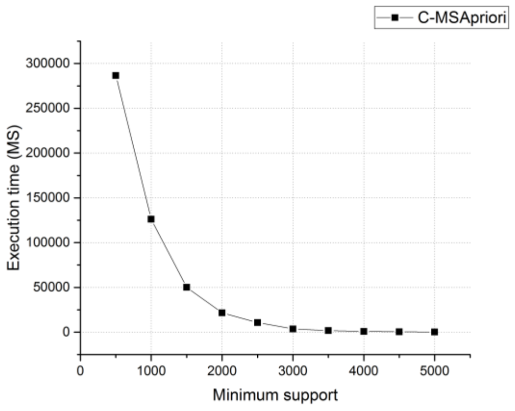

Fig. 2. Execution time C-MSApriori

V. Conclusion

In this paper we have investigated the problem of using item specific minimum support. To solve this problem we have proposed an algorithm called categorical minimum support Apriori (C-MSApriori). Our proposed algorithm is capable of mining both frequent and rare patterns effectively without mining any uninteresting pattern. We have evaluated the performance of our proposed algorithm by conducting an experiment on synthetic dataset. The results show that our proposed algorithm has come out from the rare item problem and gives user more flexible and powerful model to specify minimum support for rare item. Thus our proposed algorithm enables us to mine rare pattern without producing any uninteresting pattern.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. pages 206–216. In ACM SIGMOD International Conference on Management of Data, 1993.
[2] Agrawal, R., and Srikanth, R. “Fast algorithms for mining association rules.” VLDB, 1994.
[3] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Min. Knowl. Discov., 8(1):53–87, 2004.
[4] H. Mannila, Database methods for data mining, in: ACM SIGKDD Conf on Knowledge Discovery and Data Mining (KDD 1998) Tutorial, 1998.
[5] B. Liu, W. Hsu, and Y. Ma, “Mining association rules with multiple minimum supports,” Proceedings of 1999 ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining 1999, pp. 337-341.
[6] Yun, H., Ha, D., Hwang, B., and Ryu K. H. “Mining association rules on significant rare data using relative support.”, The Journal of Systems and Software 67, 2003, pp. 181-191.
[7] Tseng, M. C., & Lin, W. Y. (2007). Efficient mining of generalized association rules with non-uniform minimum support. Data and Knowledge Engineering, 62(1), 41–64.
[8] Hoque, F. a, Debnath, M., Easmin, N., & Rashed, K. (2011). Frequent Pattern Mining for Multiple Minimum Supports with Support Tuning and Tree Maintenance on Incremental Database. Journal of Information Technology, 3(2), 79–90.
[9] Kiran, R. U., & Reddy, P. K. (2011). Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. Proceedings of the 14th International Conference on Extending Database Technology - EDBT/ICDT ’11, 11.
[10] Uday Kiran, R., Krishna Reddy, P.: An Improved Multiple Minimum Support Based Ap- proach toMine Rare Association Rules. In: IEEE Symposiumon Computational Intelligence and DataMining, pp. 340–347 (2009)
[11] M. Hahsler. A model-based frequency constraint for mining associations from transaction data. Data Min. Knowl. Discov., 13(2):137–166, 2006.
[12] Y.-H. Hu and Y.-L. Chen. Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism. Decis. Support Syst., 42(1):1–24, 2006.
[13] C. S. K. Selvi and A. Tamilarasi. Mining association rules with dynamic and collective support thresholds. International Journal on Open Problems Computational Mathematics, 2(3):427–438, 2009.
[14] G. M. Weiss. Mining with rarity: a unifying framework. SIGKDD Explor. Newsl., 6(1):7–19, 2004.
[15] L. Zhou and S. Yau. Association rule and quantitative association rule mining among infrequent items. In International Workshop on Multimedia Data Mining, 2007.
[16] Hipp, J., Guntzer, U., Nakhaeizadeh, G.: Algorithms for Association Rule Mining - A Gen- eral Survey and Comparision. ACM Special Interest Group on Knowledge Discovery and Data Mining 2(1), 58–64 (2000)
[17] (SPMF) Fournier-Viger, P., Gomariz, Gueniche, T., A., Soltani, A., Wu., C., Tseng, V. S. (2014). SPMF: a Java Open-Source Pattern Mining Library. Journal of Machine Learning Research (JMLR)