

Top Down Approach to find Maximal Frequent Item Sets using Subset Creation

Jnanamurthy HK, Vishesh HV, Vishruth Jain, Preetham Kumar,
Radhika M. Pai

Department of Information and Communication Technology
Manipal Institute of Technology, Manipal University, Manipal-576104, India
jnanamurthy.hk@gmail.com

ABSTRACT

Association rule has been an area of active research in the field of knowledge discovery. Data mining researchers had improved upon the quality of association rule mining for business development by incorporating influential factors like value (utility), quantity of items sold (weight) and more for the mining of association patterns. In this paper, we propose an efficient approach to find maximal frequent item set first. Most of the algorithms in literature used to find minimal frequent item first, then with the help of minimal frequent item sets derive the maximal frequent item sets. These methods consume more time to find maximal frequent item sets. To overcome this problem, we propose a novel approach to find maximal frequent item set directly using the concepts of subsets. The proposed method is found to be efficient in finding maximal frequent item sets.

KEYWORDS

Data Mining (DM), Frequent Item Set (FIS), Association Rules (AR), Apriori Algorithm (AA), Maximal Frequent Item First (MFIF).

1. INTRODUCTION

With the popularization of computer and development of Database Technology, more and more data are stored in large databases. Obviously, it is impossible to find useful information without using efficient methods. Data Mining (DM)[1] techniques have emerged as a reflection of this request. Association rules mining, an important research direction aims to find out the dependence among multiple domains based on a given degree of support and credibility.

Association rules mining process is divided into two steps. The first step is to find the frequent item-sets whose support degree is larger than the initial support degree from the transaction database; the second step is to generate the rules of value from the frequent item-sets, and the acquisition of frequent item-sets is the key step during mining association rules procedure. In 1993, R. Agrawal first promoted an association rule mining algorithm named Apriori Algorithm[2]. This algorithm's basic idea is to identify all the frequent sets whose support is greater than minimum support. Frequent item generates strong association rule, which must satisfy minimum support and minimum confidence. An Apriori idea is a brief description of the core algorithm is that has two key steps: the connecting step and the pruning step [3].

- **Connecting step:** In order to identify the $L(k)$ (a frequent k set), a candidate k -items ($C(k)$) can be generated by $L(k)-1$ and its connections, which elements of $L(k)-1$ can be connected.
- **Pruning step:** $C(k)$ is a superset of $L(k)$ whose members may or may not be frequent, but all the frequent sets are included in $C(k)$. If scanning database, each count of a candidate in $C(k)$ can be determined, also $L(k)$ (frequent candidates whose count is not less than the minimum support count). However, $C(k)$ may be large, its calculation amount also be lots. For compression of $C(k)$ the Apriori may be used: any non-frequent $(k-1)$ items can not be subsets of frequent k -items. Therefore, if $(k-1)$ items of a candidate k -items is not in $L(k)$, then the candidate cannot be frequent, which can be deleted from $C(k)$.

Subsequent researchers have given a lot of improvement for the AA. However, all of these improved algorithms have the following problems in varying degrees. The first problem is that algorithms need more time complexity to produce the candidate frequent item-sets. And the second is that algorithms have to scan the transaction database many times to do the pattern-matching for candidate frequent item-sets. These two issues are both the hotspots and difficulties during current research on mining association rules. In our paper, we promote a faster and more efficient algorithm based on the classical AA.

2. BASIC CONCEPTS

Data Mining is a method that extracts some kind of information knowledge which cannot be discovered easily, but contains certain regularity from the massive primary data [4]. Let I be a set of items and D a database of transactions. Every transaction is a set of distinct items (item set) from I . An item set with k items is referred to as a k -item set. The *support of an item set X* , denoted as $\sigma(X)$, is the total number of transactions in which that item set occurs as a subset. A second formal definition for the support of an item set X is given by Agrawal. An item set X has a support of s if $s\%$ of transactions in D contains X as a subset. This second formal definition is somewhat more rigorous, as it emphasizes that the maximum support of an item set cannot exceed the total number of transactions in D . An item set is called *frequent* if its support is greater than a user-defined *minimum support* value. A frequent k -item set X is *maximal* if no other k' -item set (where $k < k'$) contains X as a subset.

An *association rule* is an expression $X \Rightarrow Y$, where X and Y are disjoint item sets. An important note is that an association rule must not be considered not only as an implication, but rather as a coexistence of the two item sets. The *support* of an association rule is given by the support of the $X \cup Y$ item set. The *confidence* of an association rule is the conditional Probability that a transaction contains Y , given that it contains X . The confidence is computed using the formula $c(X \Rightarrow Y) = \sigma(X \cup Y) / \sigma(X)$. *Minimum confidence* of a rule is a user defined value. An association rule is *strong* if it has a support greater than minimum support value and confidence greater than the minimum confidence value.

3. THEORETICAL BACKGROUND

Association rules: Association rules are statements of the form $\{X_1, X_2, \dots, X_n\} \rightarrow Y$, meaning that if we find all of X_1, X_2, \dots, X_n in the market basket, then we have a good chance of finding Y . We normally would search only for rules that had confidence above a certain threshold. We may also ask that the confidence be significantly higher than it would be if items were placed at random into baskets.

Frequent item sets: In many (but not all) situations, we only care about association rules or causalities involving sets of items that appear frequently in baskets. For example, we cannot run a good marketing strategy involving items that no one buys anyway. Thus, much data mining starts with the assumption that we only care about sets of items with high support; i.e., they appear together in many baskets. We then find association rules or causalities only involving a high-support set of items i.e., $\{X_1, \dots, X_n, Y\}$ must appear in at least a certain percent of the baskets, called the support threshold.

What is the use of learning association rules?

- With the development of e-commerce and logistics, online shopping plays an increasingly important role in people's life. Some well-known e-commerce site gets lots of benefits from mining association rules. These online shopping sites use mining association rules to get useful information from the huge database, and then set the commodity in a bundle that the customer intends to purchase together. And there are also some shopping sites which use them to set the appropriate cross-selling, where the customer who bought one product will see other related commodities advertised. [5]
- Also we are familiar with Amazon; they use association mining to recommend you the items based on the current item you are browsing/buying.
- Another application is the Search engines where after you type in a word, it searches for frequently associated words that the user types after that particular word. [6]

4. OUR CONTRIBUTIONS

Fig.1 shows activity diagram of MFIF method to find maximal frequent item first. Instead of finding minimal frequent item set first, we developed a new efficient method to find maximal frequent item set first.

Procedure:

Step1: Count the number of items present in each transaction and put in an array $a[]$.

Step2: Find the transactions having maximum items (max) in the array $a[]$.

Step3: If $\text{Count}(\max(a[])) \geq \text{min_sup}$ then transfer those transactions to an another array $\text{arr}[][]$, else find subsets.

Step4: Compare each transaction in $\text{arr}[][]$ with other transactions.

Step5: Take a Counter C and increase the counter if we found similar item sets in $\text{arr}[][]$.

Step6: If $\{C \geq \text{min_sup}\}$ then item set will be the most frequent item set.

Step7: if $C < \text{min_sup}$ then find the subsets of all transactions and store it in an array $\text{sub}[][]$.

Step8: $\text{max} = \text{max} - 1$.

Step9: add the transactions of $\text{sub}[][]$ to $\text{arr}[][]$.

Step10: Repeat from **step3** until frequent item set is found.

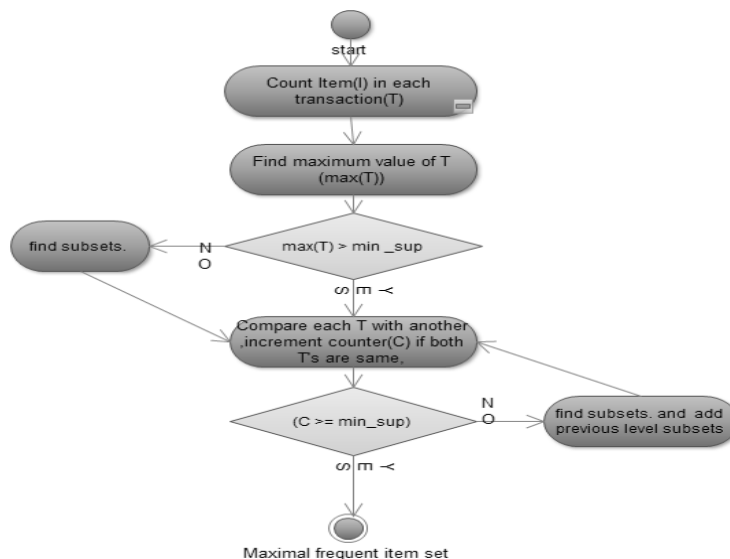


Figure 1: Procedure to find maximal frequent item sets.

5. PROPOSED METHOD

In this section we are presenting the proposed method to find maximal frequent item sets. The working procedure is divided into 2 algorithms. The first is MFIF algorithm and the second is SUBSET FORMATION algorithm.

MFIF ALGORITHM

Precondition:: $i=0$, $max=0$, $Count=0$;

```

a [ ] ← Count ( I ) in each T;
// I represents Items //
for (1 to n transaction)
  if( a[i]>max)
    max=a[i]
  endif
endfor

```

MOVE:

```

if count(max) ≥ min_sup
  move max item set to new_arr[ ] [ ]
endif

```

FIND:

```

for all transaction in new_arr[ ] [ ]
  Compare each I with (I-1) Items;
  If( I=(I-1) )
    Count++;
  endif
endfor
if(Count ≥ min_sup)
  Li ← All Item sets with min_sup;
else

```

```

        max=max-1;
    endif
    Create subsets of all transactions in
    new_arr[ ][ ] and store in sub_arr[ ][ ].
    goto MOVE;
    fin_arr[ ][ ]=new_arr[ ][ ]+sub_arr[ ][ ];
    goto FIND;

```

SUBSET FORMATION ALGORITHM

```

for all transaction in new_arr[i][j]
    temp[k]=j;
    k++;
endfor

```

```

        Initialize l=1+i*4
    for all l less than or equal to k+i*4
        l++;
    endfor
    for all m items
        item[l][m]=new_arr[u][v];
        item[l][temp[w]]=0;
        w++;
        v=0;
    endfor

```

Repeat until all the subsets are formed.

6. EXPERIMENTAL ANALYSIS

Fig.2 and Fig.3 shows the results of MFIF. The proposed method takes less time to find maximal frequent item set. Fig.2 consists of 10 transactions of 20 items as input, in which two transactions have 12 items and the values are similar; it meets 20 percent of minimum support, hence 12 item set results as most maximal frequent item set. The experiment is done till 10000 transactions.

Another example, fig.3 consists of 10 transactions of 20 items as input, here only one transaction has 13 items, and count of 13 item set transactions will be 1, which does not meet the minimum support. So subset formation is done. Subsets will be generated of 12 items from the transaction of 13 item set, later the generated subsets and other transactions which have 12 items will move and combine in one array and compare the subsets .If count value is greater than minimum support, then that set will be the Maximal frequent item set.

```

0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 1
1 1 0 1 0 0 0 0 1 1 0 1 0 1 0 1 0 0 1
0 1 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 0 1 0
0 1 1 0 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 0
1 1 1 0 1 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0
0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 1
0 1 1 0 1 0 1 1 0 0 1 0 1 0 1 0 1 0 0 1
1 1 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1
0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1
1 1 1 0 1 0 0 1 0 1 0 0 1 1 0 1 0 0 0 0
enter the minimum support
2
THE FREQUENT ITEM SET IS:
0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 1
I2 I3 I4 I5 I6 I12 I13 I14 I15 I16 I17 I20
    
```

Fig.2 MFIF Result: 12 item set resulted as maximal frequent item set.

```

1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1
1 1 0 0 0 0 1 1 0 0 1 0 0 1 0 0 1 0 0 0 1
0 1 1 0 0 0 1 1 0 0 1 0 1 0 0 1 0 0 1 0 0 0
1 1 1 0 1 0 1 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0
0 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1
1 1 1 0 0 1 0 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1
0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 0
1 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 0 0
enter the minimum support
2
generated subsets
0111110000011111001
1011110000011111001
1101110000011111001
1110110000011111001
1111010000011111001
1111100000011111001
11111100000011111001
11111100000010111001
1111110000001011001
11111100000011011001
11111100000011101001
11111100000011110001
11111100000011111000
THE COMBINED ARRAY IS:
0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 1
1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1
1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1
1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1
1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1
1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 1
1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1
1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1
1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1
1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1
THE FREQUENT ITEM SET IS:
0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 1
I2 I3 I4 I5 I6 I12 I13 I14 I15 I16 I17 I20
    
```

Fig.3 MFIF Result: 12 item set resulted as maximal frequent item set with subset generation.

Complexity of the Apriori algorithm depends on the number of item sets present in the transaction, i.e. if transaction has ‘n’ items, then we have to consider the items starting from 1 frequent item set till we find out the ‘n’ frequent item sets, so complexity increases as ‘n’ value increases.

MFIF(proposed algorithm) results in less time complexity compared to Apriori; when the item sets are large, it does not depends on the value ‘n’. Complexity increases only at the generation of subsets of each item sets, and yields less time complexity if maximal frequent item set found at the initial stage.

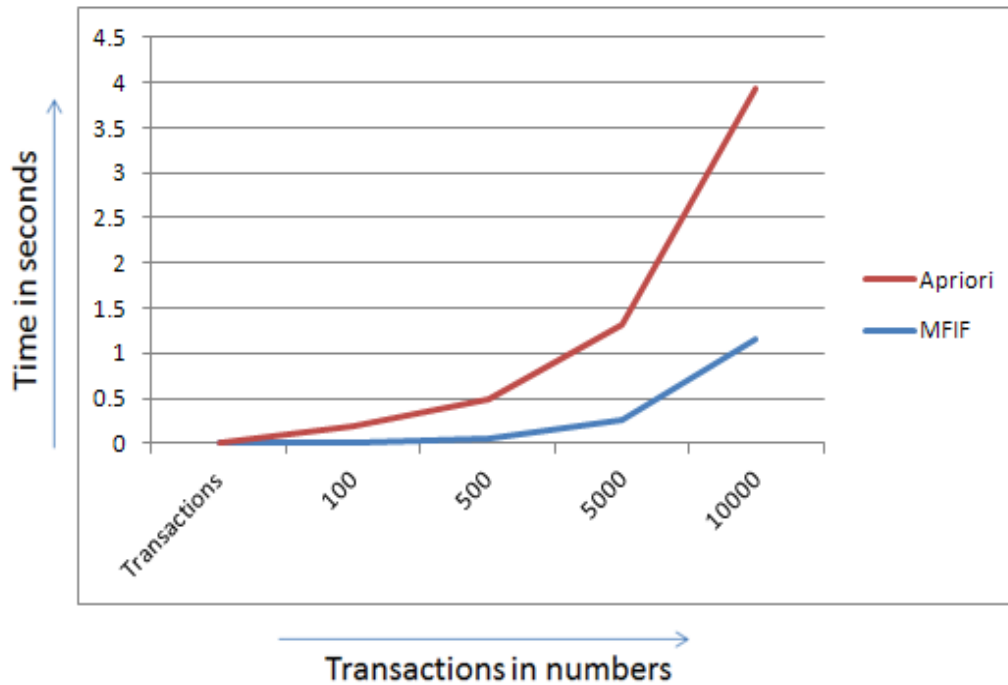


Fig.4. Graphical representation of time Complexity Comparison between Apriori and MFIF for frequent 12 item sets.

Transactions	MFIF (time in seconds)	Apriori (time in seconds)
100	0.016	0.187
500	0.062	0.422
5000	0.266	1.047
10000	1.156	2.781

Table 1: Comparison between Apriori and MFIF

The results are shown in Table 1 and graphical comparison in fig.4. Time taken by MFIF and Apriori for 100, 500, 5000, 10000 transactions is shown in the table. Time complexity of MFIF is less than Apriori. Complexity of Apriori will increase as the number of items in the frequent item set increases. In MFIF complexity does not depend on the number of item sets present. But the time complexity increases only at the time of subset generation.

The results show that, Apriori takes more time because it takes 12 scans to find out 12 element frequent set and MFIF takes 2 scans.

7. ADVANTAGES

- Too much memory space is not required for generation of subsets, because at a time only one level of element subsets are generated; as shown above only 12 element subsets are generated.
- Any element frequent set can be got in a single scan by subset creation method, which will help in applying any search method to traverse and get maximal frequent item set, and it helps in reducing the scans drastically.

8. LIMITATIONS

- For scanning we assume that maximal frequent item set will have at least 50 percent of the total number of items present.
- Extra time is taken by the subset generation algorithm to calculate, but less time is taken in comparing the transactions as its just comparing whole transaction is equal.
- If items in Maximal frequent item set are less, then MFIF algorithm takes more time than Apriori to calculate.

9. CONCLUSION

In data mining, association rule learning is a popular and well researched method for discovering interesting relations between objects in large databases. An efficient way to discover the maximal frequent set can be very important in some kinds of data mining problems. The maximal frequent set provides an effective representation of all the frequent item sets. Discovering maximal frequent item sets implies immediate discovery of all frequent item sets. This paper presents a new algorithm that can efficiently discover the maximal frequent set. The top-down searching strategy is adopted in this algorithm. This approach can be very significant and effective to find maximal frequent item set.

REFERENCES

- [1] Arun K Pujari. Data mining concepts and techniques. Universities Press, 2001.
- [2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. *In Proceedings of the 20th international conference on Very Large Data Bases (VLDB'94)*, pages 478-499. Morgan Kaufmann, September 1994.
- [3] Chengyu and Xiong Ying. Research and improvement of apriori algorithm for association rules. In *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*, pages 1 -4, may 2010.
- [4] Wei Yong-qing, Yang Ren-hua, and Liu Pei-yu. An improved apriori algorithm for association rules of mining. In *IT in Medicine Education, 2009. ITIME '09. IEEE International Symposium on*, volume 1, pages 942 -946, aug. 2009..
- [5] Guo Hongli and Li Juntao. The application of mining association rules in online shopping. In *Computational Intelligence and Design (ISCID), 2011 Fourth International Symposium on*, volume 2, pages 208 -210, oct. 2011.
- [6] Lu Nan, Zhou Chun-Guang, and Cui Lai-Zhong. The application of association rules algorithm on web search engine. *In Computational Intelligence and Security, 2009. CIS '09. International Conference on*, volume 2, pages 102 -108, dec. 2009.