© 2018 IJSRSET | Volume 4 | Issue 2 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
National Conference on Advanced Research Trends in Information and
ʒ Technologies (NCARTICT-2018), Department of IT, L. D. College of Engineering,
Ahmedabad, Gujarat, India. In association with
International Journal of Scientific Research in Science, Engineering and Technology

An Adjacency Matrix Based Apriori Algorithm for Frequent Itemsets Mining

Mahendra N. Patel[1], Suresh B Patel[2], Dr. S. M. Shah[3]
[1]PG Scholar, Computer Engineering Department, Government Engineering College, Modasa, Gujarat, India
[2]PG Scholar, Computer Engineering Department, Government Engineering College, Modasa, Gujarat, India
[3]Computer Engineering Department, Government Engineering College, Modasa, Gujarat, India

ABSTRACT

Finding frequent itemsets is a most researched field in data mining. Currently, the finding of frequent itemsets problem's solution has been proposed by many researchers. The Apriori algorithm is the basic algorithm for frequent itemsets mining. In Apriori algorithm, there are main two issues: scanning the database multiple times and generating a large number of candidate sets. In recent years several improved apriori algorithms have been defined and evaluated to improve efficiency. Our main goal is to define a new optimized algorithm and to compare its performance with the existing algorithms. The main focus of our work is to propose a new optimized algorithm and to compare its performance with the state of the art methods. In proposed work, adjacency matrix will be employed in order to improve the operating efficiency and eliminate the candidate sets. In a proposed system not require the pruning step. Performance of the proposed method will be evaluated on existing datasets. A secondary data set is used to find frequent itemsets with using our proposed algorithm and existing algorithm. The effect of our proposed algorithm is presented.

Keywords: Apriori, Data Mining, Frequent Itemsets Mining (FIM), Adjacency Matrix, FI-generator

I. INTRODUCTION

Recent days have an explosive growth in generating data in all fields of business, science, medicine, defense etc. the same rate of growth in the processing power of evaluating and analyzing the data did not follow this growth. Due to this phenomenon, a large volume of data is still kept without being studied. Data mining is a research field that tries to overcome this problem, processes some methods for the extraction of significant and potentially useful patterns from these large collections of data.

Data mining technique is used to find valid, novel, useful and ultimately understandable patterns in data [1]. In general, there are different kinds of patterns that can be discovered from data. For example, association rules can be mined for market basket analysis, classification rules can be used for accurate classifiers, clustering can be used for customer relationship management.

Most current studies on frequent pattern mining adopt an Apriori-like approach, which is based on an "anti-monotone Apriori heuristic[1][2]: if any length k-pattern is not frequent in the database, its length (k+1) super pattern can never be frequent". The original Apriori algorithm requires k scans or passes over the data where k is the length of the longest frequent itemset.

In this paper, we define Adjacency matrix base
Apriori algorithm and analyzed its performance. Paper is organized as follows. Section 2 introduces related work on apriori algorithm and existing algorithm based on adjacency matrix. Section 3 presents our proposed algorithm for frequent itemsets mining. Section 4 presents the time complexity of an existing algorithm

and our proposed algorithm. Section 5 conclusions and summarizes the paper.

II. RELATED WORK

Data mining is a process to discover hidden information or knowledge automatically from huge database [3]. First presented by Agrawal was to find frequent itemsets using mining of association rules[1]. In the following section, we try to show the concepts of frequent itemsets mining. Let TDB (Transactions databse) = {T1, T2, T3, …, Tm} where m is number of transactions. Each transaction Ti = {i1, i2, i3, …}contains a set of items from I = {i1, i2, i3, …, in} where n is number of items. An itemset X with k items from I is called a k-itemset. A transaction Ti contains an itemset X if and only if X ∊ Ti [4]. The support of a number of transactions in TDB containing X. An itemset is frequent if its support, supp(X) is greater than a support threshold called *minimum support*. For example, suppose we have a
TDB = {T1, T2, T3, T4, T5,T6} and
I = {A, B, C, D, E} where
T1 = {B, C, D, E},
T2 = {B, C, D},
T3 = {A, B, D},
T4 = {A, B, C, D, E},
T5 = {A, B, C}, and
T6 = {B, E}.

Thus, for instance, support({A}) = 3, can be achieved because A occurs only in $T_{3,}$ $T_{4,}$ and T_5 transactions. For k itemsets where $k \geq 2$, for instance, support({A, E}) = 1, because it occurs only one time in all transactions. And support({A, B, D, E}) = 1 can be computed from all transactions using the same way for generating all frequent itemsets.

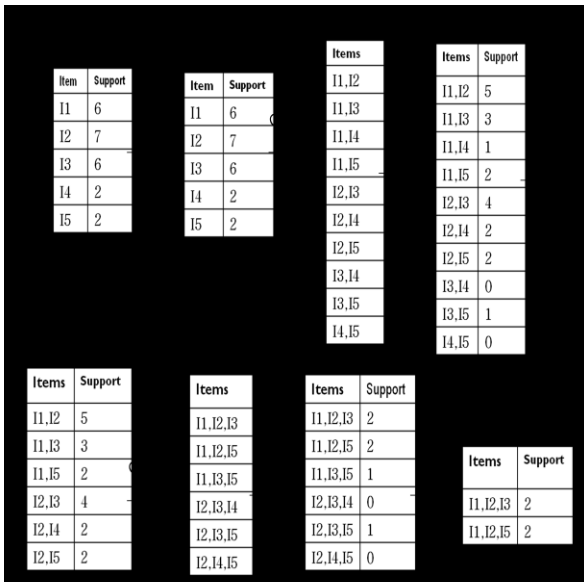
A. Apriori Algorithm

The Apriori algorithm for finding frequent itemsets was originally presented by Agrawal and Srikant [5]. It finds frequent itemsets according to a user-defined minimum support. In the first pass of the algorithm, it constructs the candidate 1-itemsets. The algorithm then generates the frequent 1-itemsets by pruning some candidate 1-itemsets if their support values are lower than the minimum support. After the algorithm finds all the frequent 1-itemsets, it joins the frequent 1-itemsets with each other to construct the candidate 2-itemsets and prune some infrequent itemsets from the candidate 2-itemsets to create the frequent 2-itemsets. This process is repeated until no more candidate itemsets can be created.

Consider the Apriori algorithm with the example shown in Figure 1 for the transaction database shown in Table 1 and also assume that minimum support is 2. In this C1 represent1-candidate itemset and after pruning it generate L1 which represents 1-frequent itemset. After L1 join operation applies and then C2 is generated. The same way other C3, L2, and L3 are generated. However, there are no 4-itemsets in pass 4 created because no candidate 4-itemsets can be created. Thus, the process stops.

Table 1. Transaction Dataset

TID	List Of Items IDs
T1	I1, I2, I5
T2	I2, I4
T3	I2, I3
T4	I1, I2, I4
T5	I1, I3
T6	I2, I3
T7	I1, I2
T8	I1, I2, I3, I5
T9	I1, I2, I3



Item	Support
I1	6
I2	7
I3	6
I4	2
I5	2

Item	Support
I1	6
I2	7
I3	6
I4	2
I5	2

Items
I1,I2
I1,I3
I1,I4
I1,I5
I2,I3
I2,I4
I2,I5
I3,I4
I3,I5
I4,I5

Items	Support
I1,I2	5
I1,I3	3
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

Items	Support
I1,I2	5
I1,I3	3
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2

Items
I1,I2,I3
I1,I2,I5
I1,I3,I5
I2,I3,I4
I2,I3,I5
I2,I4,I5

Items	Support
I1,I2,I3	2
I1,I2,I5	2
I1,I3,I5	1
I2,I3,I4	0
I2,I3,I5	1
I2,I4,I5	0

Items	Support
I1,I2,I3	2
I1,I2,I5	2

Figure 1. The apriori example

B. Improved Apriori Algorithm based on Adjacency matrix

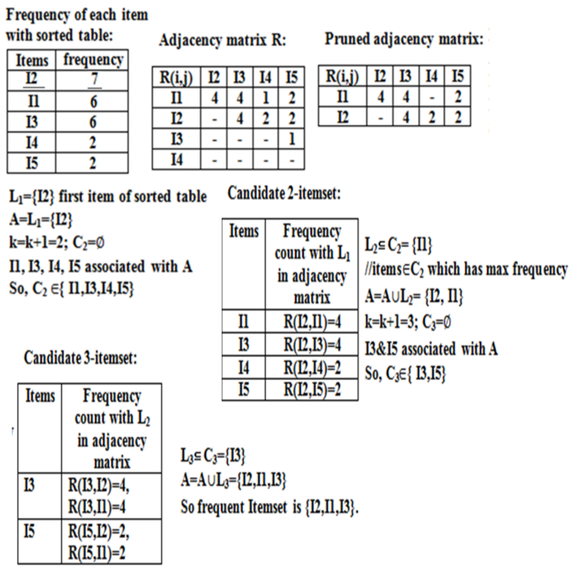
Now in this section, we discuss FI-generator algorithm proposed by Archana Singh and Jyoti Agarwal [6]. In this method used Adjacency matrix as a data structure.

In this method [6] first scan the database and generate the frequency-wise sorted table of items in descending order. In first pruning step all the items, having frequency count less than minimum support is removed from the sorted table (TS). Again scan the database and create the adjacency matrix say R. In pruning step all the entries from adjacency matrix with frequency count less than minimum support are pruned. Now select the first item from the sorted table as 1-frequent itemset say L1. Then generate the candidate 2-itemset is the item set which has relation with L1 in adjacency matrix R. From the candidate 2-itemset select the item whose frequency is more and take union with L1, so we get 2-frequent itemset say L2. Then generate candidate 3-itemset is the item set which has relation to all items of L2 in adjacency matrix (R). Thus the size of Candidate set is reduced in the successive iteration.

Working of an adjajency matrix based [6] apriori algorithm

Let's take the example of given algorithm based on one sample database transactions shown in Table 1. Assume the minimum support threshold is 2. Example of given algorithm for frequent itemset generation shown in Figure 2 [7].

For example shows first the sorted table, second the adjacency matrix R and third pruned adjacency matrix. In the example, we select {I2} as a 1-frequent itemset from a sorted table whose frequency is higher than other. Then generate candidate 2-itemset which associated with {I2}. Next, we select {I1, I2} is a 2-frequent itemset whose frequency is higher than other. Same ways finally select {I1, I2, I3} as a 3-frequent itemset.



Frequency of each item with sorted table:

Items	frequency
I2	7
I1	6
I3	6
I4	2
I5	2

Adjacency matrix R:

R(i,j)	I2	I3	I4	I5
I1	4	4	1	2
I2	-	4	2	2
I3	-	-	-	1
I4	-	-	-	-

Pruned adjacency matrix:

R(i,j)	I2	I3	I4	I5
I1	4	4	-	2
I2	-	4	2	2

L_1={I2} first item of sorted table
$A=L_1$={I2}
$k=k+1=2$; $C_2=\emptyset$
I1, I3, I4, I5 associated with A
So, $C_2 \in$ { I1,I3,I4,I5}

Candidate 2-itemset:

Items	Frequency count with L_1 in adjacency matrix
I1	R(I2,I1)=4
I3	R(I2,I3)=4
I4	R(I2,I4)=2
I5	R(I2,I5)=2

$L_2 \subseteq C_2$= {I1}
//items$\in C_2$ which has max frequency
$A=A\cup L_2$= {I2, I1}
$k=k+1=3$; $C_3=\emptyset$
I3&I5 associated with A
So, $C_3 \in$ { I3,I5}

Candidate 3-itemset:

Items	Frequency count with L_2 in adjacency matrix
I3	R(I3,I2)=4, R(I3,I1)=4
I5	R(I5,I2)=2, R(I5,I1)=2

$L_3 \subseteq C_3$={I3}
$A=A\cup L_3$={I2,I1,I3}
So frequent Itemset is {I2,I1,I3}.

Figure 2. The FI-generator example

III. PROPOSED WORK

In this proposed method, we are improving the performance of an apriori like algorithm. In this proposed method we can enhance the efficiency of an enhanced apriori algorithm based on adjacency matrix [6].

In this proposed method, we take adjacency matrix as our data structure. In adjacency matrix R(i,j) = frequency count of item (i) and item(j) appears together in database. Below show the proposed algorithm:

Pseudo Code of Proposed Algorithm

Input: Transaction Database T, min_sup
Output: Frequent item set

Method:

Step 1: Create Transaction IDs table
Scan the transaction database and create T_ids table. T_ids first column consist item and second column consist transaction id's associated with that item.

Step 2: Creation of adjacency matrix (R)
Create an adjacency Matrix (R), where rows and column shows the different items from T_ids table.

Rows i=1 to n-1 //where n is number of items Columns j=2 to n In adjacency matrix R(i,j) =frequency count of item(i)and item(j) appears together in database.

Step 3: Frequent item set generation:

Initialize vector C & FI // empty

```
For i=1 to n-2
    Initialize a=0
    For j=i+1 to n
    If R(i,j) value ≥ min_support then
        C(a)=j
        Increment a
    End If
    End For
    Initialize b=0
    For k=b to a-1
        For l=k+1 to a
        If R(C(k),C(l)) value ≥
        min_support then
            Insert into FI Frequent
            Item Set (i,C(k),C(l))
          End If
            Increment l
          End For
    End For
End For
```

Step 4: Display result and stop

Display FI contains frequent item set

Working of an proposed algorithm

Let's understand this step through an example, for that use Table 1 as sample transaction database. First, create Transaction_id table shown in Table 2. Then generate adjacency matrix (R) from that Transation_id table shown in Table 3. Finally, apply search technology on this adjacency matrix. In this example we take min_support value is 2.

Table 2. Transaction_id

Item	T_id's						
I1	1	4	5	7	8	9	
I2	1	2	3	4	6	8	9
I3	3	5	6	7	8	9	
I4	2	4					
I5	1	8					

Table 3. Adjacency matrix R

R(I,j)	I2	I3	I4	I5
I1	4	4	1	2
I2	-	4	2	2
I3	-	-	0	1
I4	-	-	-	0

Steps for search technique:

Step 1: fetch the row for i=1 to n-2where n = number of items(5).

Here first take I1 as a first-row item

Step 2: With this row find the associated value whose value greater or equal to min_support (User input=2).

So here associated with I1 are I2, I3, and I5 whose value is 4, 4, and 2 respectively in the adjacency matrix.

Step 3: Check the (I2, I3), (I2, I5) and (I3, I5) are frequent or not. Means here find R(2,3)= 4, R(2,5) = 2 and R(3,5) = 1 from adjacency matrix. So here we find (I2, I3), (I2, I5) are frequent.

Step 4: Display frequent item set

In step 3 find (I2, I3) and (I2, I5) which associate with row I1, so we display (I1, I2, I3)and (I1, I2, I5) as frequent item set.

Step 5: Repeat step 1 to step 4 for next row detail given below.

For the second row is I2.

Find I3, I4, and I5 associated with I2 and whose value is 4, 2, and 2 respectively.Here find R(3,4) = 0, R(3,5) = 1 and R(4,5) = 0, so here no one are frequent.

Now we check for the third row is I3

Find I4 and I5 associated with I3 and whose value is 0 and 1 respectively. Here all values are less than min_support, so here no one is frequent.

End of the procedure because search up to n-2 which is 3 was completed.

Finally, from the example shown above, we get the output of 3-frequent itemset is {(I1, I2, I3), (I1, I2, I5)}. This result is same as apriori algorithm.

IV. COMPARISONS AND DISCUSSION

We compare proposed system and existing system with respect to time complexity as shown below:

Time to execute steps in pseudo code, where n is a number of items, m is a number of transactions and T is transaction database.

First, in existing System [6] require time to find the frequency of each item ∈ T is O (nm), Time to create of the sorted table and pruning of sorted table is O (n). In proposed system require time to create transaction_id table is O (nm).

Second, in the existing system requires time to create adjacency matrix is O (n) ^{2} and Time to prune adjacency matrix is O (n) ^{2}. So the overall time complexity is O ($2n^2$). Whereas in the proposed system require time to create adjacency matrix is O (n) ^{2}.

Third, in the existing system requires two times to generate frequent itemset is O (n-c), c is the number of items pruned in the candidate set. In proposed system require time to generate frequent itemset is O $(n)^2$.

So theoretically, we say that performance of the proposed system is better than the existing system. The proposed algorithm is yet to be implemented on Java platform and perform on dataset [6] and it will practically prove that proposed system is better than existing one.

V. CONCLUSION

Frequent itemsets mining is one of the most important areas of data mining. Existing implementations of the Apriori-based algorithms focus on the way candidate itemsets generated, the optimization of data structures for storing itemsets, and the implementation details. A key contribution of this research paper is to provide the user with a simple but yet powerful, adjacency matrix structure for efficient frequent pattern mining. There are various frequent itemset mining methods are available. These frequent itemset mining scans database multiple times and generates more candidate set. So currently available frequent itemset mining methods are lacking in speed. Current existing method based on adjacency matrix [6] does not generate all frequent itemset. The main objective of this research paper is to propose a new algorithm to find all frequent itemset without generating candidate sets by single time database scanning. So it reduces the execution time and memory requirement.

VI. REFERENCES

[1]. Agrawal and Rakesh,"Fast Algorithms for Mining Association Rules in Large Databases",Proceedings of the ACM SIGMOD International Conference Management of Data,Washington,Vol. 22,Issue 2,pp.207-216,1993.

[2]. Jiawei Han and Micheline Kamber,Book:"Data Mining,Concept and Techniques".

[3]. G. Piatetski-Shapiro,"Discovery,Analysis,and

[4]. Presentation of Strong Rules. In Knowledge Discovery in Databases",Piatetski-Shapiro,G.,editor,AAAI/MIT Press,pp. 229-248,1991.

[5]. Ye,Yanbin,and Chia-Chu Chiang. "A parallel apriori algorithm for frequent itemsets mining." In Software Engineering Research,Management and Applications",Fourth International Conference on,pp. 87-94. IEEE,2006.

[6]. R. Agrawal and R. Srikant,"Fast Algorithms for Mining

[7]. Association Rules," Proceedings of the 20th International Conference on Very Large Data Bases,pp. 487-499,1994.

[8]. Archana Singh and Jyoti Agarwal,"Proposed algorithm for frequent itemset generation",Seventh IEEE International Conference on Contemporary Computing,pp. 160-165,2014.

[9]. DixitaTandel,Neha Soni,"Survey on Improvement of Apriori Algorithm",International Journal of Innovative Research in Computer and Communication Engineering Vol. 5,Issue 1,ISSN 2320-9801, January 2017.

[10]. X. Luo and W. Wang,"Improved AlgorithmsResearch for Association Rule Based on Matrix," International Conference on intelligent computing and Cognitive Informatics,pp. 415–419,Jun. 2010.

[11]. Jiaoling Du,Xiangli Zhang,Hongmei Zhang and Lei Chen,"Research and Improvement of Apriori Algorithm",IEEE Sixth International Conference on Science and Technology,pp.117-121,2016.

[12]. Jaishree Singh, Hari Ram, Dr. J.S. Sodhi," Improving Efficiency of Apriori Algorithm Using Transaction Reduction",International Journal of Scientific and Research Publications Volume 3,Issue 1,ISSN 2250-3153,January 2013

[13]. Jiao Y.,"Research of an Improved Apriori Algorithm in Data Mining Association Rules",International Journal of Computer & Communication Engineering,2(1)),pp.25-27,2013.

[14]. Singh,Archana,Jyoti Agarwal,and Ajay Rana,"Performance measure of similis and FP-growth algorithm",International Journal of Computer Applications,2013.

[15]. Han,Jiawei,Jian Pei,and Yiwen Yin,"Mining frequent patterns without candidate generation.",ACM sigmoid record,Vol. 29. No. 2,2000.