Volume 4, Issue 4, April 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Improved Apriori Algorithm VIA Frequent Itemsets Prediction

Dr. Ayman E. Khedr
Information Systems Department,
Faculty of computer Science
Helwan University, Cairo Egypt

Fahad Kamal Alsheref
Information Systems Department,
Modern Academy, Cairo
Egypt

Abstract--Apriori algorithm is a classical algorithm of association rule mining. It is used to understand what products and services customers tend to purchase at the same time. This classical algorithm is inefficient due to so many scans of database. And if the database is large, it takes too much time to scan the database. Based on this algorithm, this research indicates the limitation of the original Apriori algorithm of wasting time for scanning the whole database searching for the frequent itemsets, and presents an improvement on Apriori by reducing that wasted time depending on scanning the database transactions once *to predict the* ***frequent pattern with different itemset sizes. The research findings - based on comparative experiments on both the original Apriori algorithm and the proposed algorithm using a well-known database of transactions - show that the proposed algorithm reduces the database scan time by 66.66% in comparison to the original Apriori algorithm. Moreover, the proposed algorithm can be used as a predictive algorithm for associations with fewer number of database scans in comparison with the original Apriori algorithm.***

Keywords: Apriori algorithm, association rules, candidate-itemsets, data mining.

I. INTRODUCTION

The amount of data around us is so huge; there are valuable information that should be extracted. Data mining is the process of discovering interesting and valuable information from huge data stored in data warehouse, OLAP (On-line Analytical Process), databases and other repositories of information. This data may exceed several terabytes. Data mining is also called Knowledge Discovery in Databases (KDD), and it integrates techniques from many disciplines such as statistics, neural networks, database technology, machine learning, information retrieval, and so on. Interesting patterns are extracted at reasonable time by KDD techniques. KDD process has several steps which are performed to extract patterns for users, such as data cleaning, data selection, data transformation, data pre-processing, data mining and pattern evaluation. [1]

The architecture of data mining system has the following main components: data warehouse, databases or other repositories of information; a server that fetches the relevant data from repositories based on the user's request; a knowledge base used as a search guide according to defined constraints; a data mining engine including a set of essential modules, such as characterization, classification, clustering, association, regression and analysis of evolution; a pattern evaluation module that interacts with the data mining modules to strive towards the interested pattern; finally, graphical user interfaces through which the user can communicate with the data mining system, thus allowing the user to interact.[1][2]

This research is organized as follows. The first section shows the related works; the second section shows the original algorithm of association rule then the proposed algorithm. The final part contains a case study to verify the proposed hypothesis.

II. RELATED WORK

Association rule mining is one of the important techniques in data mining which is used to find association between various itemsets in large transaction database. Apriori algorithm is one of the widely used algorithms for association rule mining. It was introduced by Agarwal in 1993 [3]; it is a strong algorithm which is used to find frequent itemset. A basic property of Apriori algorithm is "every subset of frequent itemsets is still frequent itemset, and every superset of a non-frequent itemset is not a frequent itemset". Apriori algorithm is very costly in terms of space and time complexity since it requires repeated database scans [4]. Many researchers have worked on improving the Apriori algorithm:

- X. Luo and W. Wang have converted an event database into matrix database, hence improving the computing time.[5]
- JI and Zhang in [6] have provided an optimized Apriori algorithm on two items generation, thus improving the time and space complexity. However, meaningless frequent itemsets still exist in Apriori algorithm.
- Y. Shaoqian [7] uses weighted support and confidence to optimize the Apriori algorithm.
- L. Lu and P. Liu in [8] have designed an algorithm which scans the database only once and without pruning.
- T. Junfang in [9] discusses the advantage of a compressed database that tremendously decreases the time of searching associated items.
- R.Agarwal in [10] introduces N-dimensional inter-transaction association rule; support and confidence measures are defined.

This research aims at improving the performance of Apriori algorithm by reducing the scanning process through predicting the frequent itemsets via single database scan.

III. ASSOCIATION RULE

The association rule represents an unsupervised learning method that attempts to capture associations between groups of items. Association rules have also been referred to in the literature as Market Basket analysis or Affinity analysis. Affinity analysis is a data analysis and data mining technique that discovers co-occurrence relationships among activities performed by (or recorded about) specific individuals or groups. In general, this can be applied to any process where agents can be uniquely identified and information about their activities can be recorded. In retail, affinity analysis is used to perform [11] market basket analysis, in which retailers seek to understand the purchase behaviour of users. This information can then be used for purposes of cross-selling and up-selling in addition to influencing sales promotions, loyalty programs, store design, and discount plans. Data mining for association rules is a two-stage process:

- Finding all frequent itemsets.
- Generating strong rules from frequent itemsets.[12]

III.I. Formal Definitions

- **Support** shows the frequency of the patterns in the rule; it is the percentage of transactions that contain both A and B.[13]

  Support = Probability (A and B)

- **Confidence** is the strength of implication of a rule; it is the percentage of transactions that contain B if they contain A.[13]

  Confidence = Probability (B if A) = P(B/A)

- **Strong Rules** is the rules whose minimum support and confidence values are bigger than the support and confidence threshold.[13]

- **Frequent Itemsets**
  - An itemset is simply a set of items.
    - An itemset containing k items is called a k-itemset.
  - The support of an itemset is similar to the support of a rule:
    - The number of transactions that contain that itemset is expressed as a percentage of the total number of transactions.
  - An itemset whose support is at least equal to the minimum support threshold is called a frequent itemset.[13]

III.II. Finding the Frequent Itemsets

Most of the computational effort needed to mine for association rules is used to find all the frequent itemsets. If m distinct items are contained then there are 2 m possible itemsets, and any practical algorithm must drastically prune this search space. This process consists of two main phases:

- **Phase 1: The Apriori Algorithm**: is an algorithm for finding patterns in data. It is based on a really simple observation: if very few people go to Starbucks and McDonald's on the same day, then there cannot be a lot of people going to Starbucks, McDonald's, and Taco Bell on the same day. So if you want to find combinations of three stores that lots of people go to on the same day, you do not have to look at combinations that include two stores that very few people go to on the same day. This tremendously reduces the number of combinations you need to look at. The basic idea is that all non-empty subsets of frequent itemsets must themselves be frequent. If an itemset i is not frequent then probability of i is less than the minimum support threshold, and if another item a is added to i then the resulting new itemset is i union a clearly probability of i union a is less than or equal probability of i hence Probability of i union a is less than the minimum support threshold.[14]

  Thus the frequent (k+1)-itemsets can be derived efficiently if the frequent k-itemsets are already known. The (k+1)-itemsets are formed by joining pairs of itemsets from the frequent k-itemsets, eliminating all those having subsets that are not in the frequent k-itemsets.

  Steps to Perform Apriori Algorithm: [14]

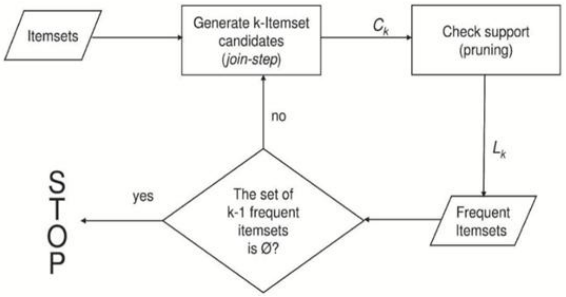

Figure 1 Apriori Algorithm

- **Phase 2: Generating Association Rules from Frequent Itemsets:**
  For all pairs of frequent itemsets (assume they are called A and B) such that A union B is also frequent, calculate c, the confidence of the rule: c = support (A U B) / support (A) and if c is bigger than or equal the minimum confidence threshold, then the rule is strong and should be included. The itemset that matches the confidence rule is the frequent itemset. The following pseudo code shows the original Apriori algorithm: [14]

```
Input: D: Database of transactions; min_sup: minimum support
threshold
Output: L: frequent itemsets in D
Method:
   L1=find_frequent_1-itemsets(D);
   For(k=2;Lk-1≠Φ; k++)
     {
      Ck=apriori_gen(Lk-1);
      for each transaction t∈D
      {
         Ct=subset(Ck,t);
         for each candidate c∈Ct
         c.count++;
       }
       Lk={ c∈Ck |c.count≥min_sup };
      }
   Return L=UkLk ;

Procedure apriori_gen( Lk-1: frequent (k-1)-itemsets)
   for each itemset l1∈ Lk-1
    {
      for each itemset l2∈ Lk-1
      {
      if(l1 [1]= l2 [1])∧ (l1 [2]= l2 [2]) ∧...∧(l1 [k-2]= l2 [k-2]) ∧(l1 [k-1]< l2 [k-1])
      then      {
       c=l1 l2;
          for each itemset l1∈Lk-1
          {
            for each candidate c ∈Ck
             {
               if l1 is the subset of c then
               c.num++;
             }
          }
       }
    }
  }
 }
C'k={ c∈Ck |c.num=k};
Return C'k;
```

III.III. Limitations of Apriori Algorithm

Apriori algorithm suffers from some weakness in spite of being clear and simple. The main limitation is costly wasting of time to hold a vast number of candidate sets with much frequent itemsets, low minimum support or large itemsets. For

example, if there are 104 from frequent 1-itemsets, it need to generate more than 107 candidates into 2-length which in turn they will be tested and accumulate [15]. Furthermore, to detect frequent pattern in size 100 (e.g.) v1, v2… v100, it has to generate 2100 candidate itemsets that yield on costly and wasting of time of candidate generation. So, it will check for many sets from candidate itemsets, also it will scan database many times repeatedly for finding candidate itemsets. Apriori will be very low and inefficiency when memory capacity is limited with large number of transactions. [15]

IV. PROPOSED ALGORITHM

As mentioned in the previous section, the performance of Apriori algorithm deteriorates in databases of huge size; Apriori algorithm needs to scan the database several times to find the frequent patterns which slows down the performance. The proposed algorithm scans the database transactions once then it predicts the frequent items sets with different sizes.

IV.I. Hypothesis

Our proposed algorithm depends on reducing the number of database scans to improve the algorithm performance. In frequent pattern prediction algorithm for any given dataset, the minimum support value is bigger than 50%. This value is chosen for the following hypothesis:

Propose there are the following:

K database transactions

L is the list of items that is forming the database transactions

li is one of the items in L

So our hypothesis is:

If the support values of several li in L are bigger than 50%, it means that there is at least one or more database transaction containing these items together. Hence, it is concluded that frequent patterns exist. The following diagram illustrates the proposed hypothesis:[16]

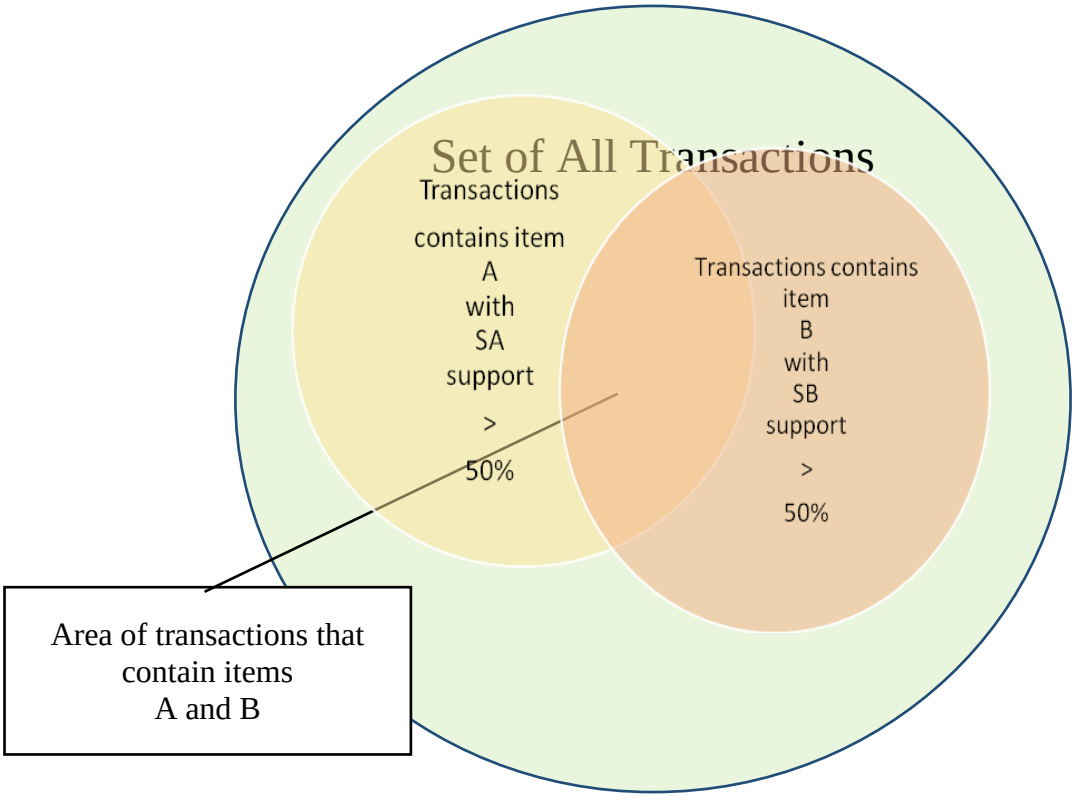

Figure 2 proposed algorithm Venn diagram [16]

Applying the proposed hypothesis in this diagram:

Number of all transactions =K

Support of item A= S (A) (Transactions that contain item A) and it is bigger than 50% (K/2)

Support of item B=S (B) (Transactions that contain item B) and it is bigger than 50% (K/2)

So if support of A > K/2 and support of B > K/2 then there must be an intersection area between A and B that contains AB. The intersection of sets A and B is the set containing all the elements that are common to both sets A and B which means support of items AB.

Support of AB (intersection area) increases when both items' support value is more than 50%, which confirms the existence of transactions that contain both items without scanning the transactions many times. That leads to the following rule:

For any two itemsets A and B:

If S(A) > 50% and S(B) > 50% then it is a predictor to suggest AB as 2 size frequent itemset and vice versa.

IV.II. Improved Algorithm

The following figure represents improved algorithm flowchart:

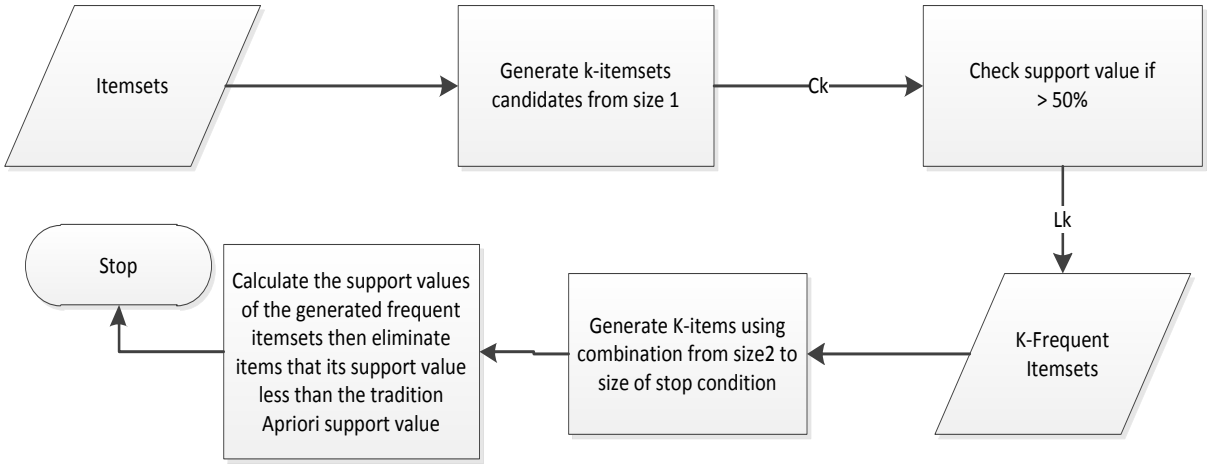

Figure 3 flowchart of the proposed algorithm

The algorithm starts by taking the itemsets and the database transactions; the next step is generating the itemsets candidates from size 1 while supporting each element in the itemsets. Each item whose support value is bigger than 50% is taken as frequent itemset and is added to the frequent itemsets.

After this step all frequent itemsets with size 1 become the same as the original Apriori algorithm, then the algorithm generates the frequent itemsets from size 2 to the size of the stop condition by combining the itemsets.

The generated frequent itemsets is a prediction of the frequent itemsets that should be generated by the original Apriori algorithm. After this step, the algorithm calculates the support values of the generated frequent itemsets and they are compared to the support value of the original Apriori algorithm and the results are verified. The generated itemsets whose support value is less than the original Apriori support value are then eliminated.

Proposed algorithm pseudo code:

Input: D: Database of transactions; **min_sup= 50%;** S=Stop itemsets size; T_sup= tradition Apriori support value;

Output: L: frequent itemsets in D

Method:

```
L1=find_frequent_1-itemsets(D);
C1=apriori_gen(L1, min_sup);
for each transaction t∈D
{
    Ct=subset(Ck,t);
    for each candidate c∈Ct
    c.count++;
 }
 L1={ c∈Ck |c.count > min_sup };

For(k=2;k<=S; k++)
   {
     Ck += apriori_gen(Lk);
    }
  L=L1 U Ck
For each transaction t∈D
{
Ct=subset(L,t);
For each candidate c∈Ct
c.count++;
 }
```



```
Lfinal={ c∈Ck |where c.count >= T_sup };
Return Lfinal;

Procedure apriori_gen( Lk-1: frequent (k-1)-itemsets)
 for each itemset l1∈ Lk-1
 {
  for each itemset l2∈ Lk-1
  {
   if(l1 [1]= l2 [1])∧ (l1 [2]= l2 [2]) ∧…∧(l1 [k-2]= l2 [k-2]) ∧(l1 [k-1]< l2 [k-1]) then      {
    c=l1 l2;
        for each itemset l1∈Lk-1
        {
          for each candidate c ∈Ck
          {
            if l1 is the subset of c then
            c.num++;
          }
        }
      }
  }
 }
C'k={ c∈Ck |c.num=k};
Return C'k;
```

IV.III. An Example of the Improved Algorithm

The following table shows nine transactions stored in the database; these transactions consist of five items I1, I2, I3, I4, and I5:

Table 1
Customers' Transactions

TID	List of Itemsets
T1	I1,I2,I5
T2	I2,I4
T3	I2,I3
T4	I1,I12,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

- **Applying Original Apriori Algorithm:**
  Suppose to run the classic Apriori algorithm with min_support=22% and min_confidence =50%
  Apriori requires 17 scans for the database given in Table 1. Frequent itemsets are listed in table 2 and the strong relations are listed in table 3.

Table 2
Frequent Itemsets Generated from Apriori

Itemset	Support >= min_support
I1	6

I2	7
I3	6
I5	2
I2,I3	4
I1,I3	4
I1.I2	4
I1.I5	2
I2.I5	2
I1,I2,I3	2

Table 3
Strong Relationships

Itemset	Support >= min_support (%)
I1→I2	66.67
I1→I3	66.67
I1→I5	33.33
I2→I1	57.14
I2→I3	57.14
I2→I5	28.57
I3→I1	66.67
I3→I2	66.67
I5→I1	100
I5→I2	100
I 1, I2 →I3	50
I 1, I3 → I2	50
I 3, I2 → I1	50

The Apriori algorithm started to generate the size 2 frequent itemsets then it continued to rescan the database once more to calculate the support values for the new generated candidates until it stops when the frequent itemsets is empty.

- **Improved Apriori algorithm:**

The improved algorithm scans the database once to find the support for each single item, and the items with support more than 50% will be taken as frequent items. The next step is to generate the frequent itemsets whose size exceeds 1 by combining extracted frequent itemsets.
The algorithm gives a prediction that the itemsets in table 5 is the frequent itemsets, by comparing itemsets in both table 5 and table 3, it becomes evident that the accuracy of the improved algorithm verifies the hypothesis with fewer database scans.

Table 4: Frequent Itemsets Generated from Improved Apriori

Itemset	Support >= min_support	Decision
I1	6	Taken
I2	7	Taken
I3	6	Taken
I5	2	Not Taken
I2,I3	4	Taken
I1,I3	4	Taken

I1.I2	4	Taken
I1.I5	2	Not Taken
I2.I5	2	Not Taken
I1,I2,I3	2	Taken

Table 5
Generated Frequent Itemsets with 70% Accuracy

Generated Itemsets
I1
I2
I3
I1,I2
I1,I3
I2,I3
I1,I2,I3

Based on our hypothesis after the first scan of the database transactions with 50% support value, frequent itemsets with size 1 similar to the original Apriori are given. The generated frequent sets are combined to generate other frequent itemsets of bigger size in order to reach the stop condition; the generated frequent itemsets are a predication to the frequent itemsets; the same as if the original Apriori algorithm has been run.

Table 4 shows that the proposed algorithm based on our hypothesis eliminates I5, (I1, I5) and (I2, I5) suggesting that the accuracy of the prediction, presented in table 5, is 70%. The accuracy is calculated by:

Accuracy= (frequent itemsets generated by the improved algorithm) / (frequent itemsets generated by the original Apriori algorithm). Like any prediction, there is a possibility to be a correct or a wrong prediction. The previous small example shows that the difference between the two algorithms is so small. This algorithm has been tested on several datasets as shown in the next section.

V. EXPERIMENTAL RESULTS

The performance of the proposed algorithm has been evaluated by comparing its execution time and results to the execution time and results of the original Apriori algorithm.

In order to test the performance of the two algorithms, an extensive experiment has been done. The procedures of the experiment are as follows:

1. Preparing the dataset: WEKA program [17] with its Supermarket dataset has been used. It is a real-world transaction dataset from a small New Zealand supermarket that contains 4627 transactions with 217 items. The items are aggregated to the department level in the supermarket, so, for example, "bread and cake" covers a number of different products, and a value of "t" indicates that the customer's shopping cart contains at least one product from that department.
2. An implementation plan for the original Apriori and the improved algorithm has been developed.
3. The original Apriori algorithm has been tested on the supermarket dataset several times with different support and confidence values and the stop condition generates frequent itemsets with size 3.
4. The proposed algorithm will be tested on the same dataset and the stop condition generates frequent itemsets with size 3.
5. The predicted frequent itemsets generated from the proposed algorithm are compared to those generated from the original Apriori algorithm to verify the results and determine the accuracy of the proposed algorithm.

V.I Experiment 1: Original Apriori algorithm with 20% support and 90% confidence

The first experiment is applying the original Apriori algorithm with 20% support value and 90% confidence value. Table 6 shows the number of transaction scans and frequent items number from size 1 to size 3.

Table 6
Experiment 1 with 20% support and 90 % confidence

	Number of frequent item list in size k
No. of items of the size 1	38
No. of items of size 2	225
No. of items of size 3	302

The number of database scans is three times for calculating the support value for each item to find the frequent items.

V.II Experiment 2: Original Apriori algorithm with 30% support and 90% confidence

The second experiment is applying the original Apriori with 30% support value and 90% confidence value. Table 7 shows the number of transaction scans and frequent items number from size 1 to size 3.

Table 7
Experiment 2 with 30% support and 90 % confidence

	Number of frequent item list in size k
No. of items of size 1	25
No. of items of size 2	69
No. of items of size 3	20

V.III Experiment 3: Original Apriori algorithm with 40% support and 90% confidence

The third experiment is applying the original Apriori with 40% support value and 90% confidence value. Table 8 shows the number of transaction scans and frequent items number from size 1 to size 3.

Table 8
Experiment 3 with 40% support and 90% confidence

	Number of frequent item list in size k
No. of items of size 1	18
No. of items of size 2	16
No. of items of size 3	-

The number of database scans is three times, there is no frequent itemsets of size 3 because the generated itemset of size 3 does not match the minimum support value of 40%.

V.IV Experiment 4: Original Apriori algorithm with 50% support and 90% confidence

The fourth experiment is applying the original Apriori with 50% support value and 90% confidence value. Table 9 shows the number of transaction scans and frequent items number from size 1 to size 3.

Table 9
Experiment 3 with 40% support and 90% confidence

	Number of frequent item list in size k
No. of items of size 1	9
No. of items of size 2	2
No. of items of size 3	-

The number of database scans is three times, there is no frequent itemsets of size 3 because the generated itemset of size 3 does not match the minimum support value of 50%.

V.V Experiment 5: The proposed algorithm with 50% support and 90% confidence

For applying the proposed algorithm, it has been tested it on the same dataset with support value of 50%, as mentioned in the hypothesis in section 4.1. The Apriori algorithm generated 9 items in the frequent list of size 1.
The proposed generated algorithm

Table 10
Experiment 3 with 40% support and 90% confidence

	Number of frequent item list in size k
No. of items of size 1	9
No. of items of size 2	45
No. of items of size 3	120

Number of database scans is 1 and the other frequent itemsets are generated by combination, as mentioned in the research hypothesis.

V.VI Discussion

The proposed algorithm has been compared to the original Apriori algorithm regarding the following two points:

1. Number of database scans to find the frequent items
2. The itemsets generated by both for calculating accuracy and precision

Table 11 and table 12 show the comparison between the original Apriori algorithm in the four experiments - with support values of 20%, 30%, 40% and 50% - and the proposed algorithm, and also imply the accuracy of the proposed algorithm of the generated item lists for each size. The accuracy was calculated using the binary classification between the generated frequent items from the two algorithms shown in figure 4.

	True	False
Positive	True positive	False positive
Negative	False negative	True negative

Figure 4 Binary classification of accuracy [18]

Accuracy is the proportion of true results (both true positives and true negatives) in the generated items. [18]

$$\text{accuracy} = \frac{\text{number of true positives} + \text{number of true negatives}}{\text{number of true positives} + \text{false positives} + \text{false negatives} + \text{true negatives}}$$

Table 11
Comparison between the original Apriori and the proposed algorithm

Experiment / Parameter	Experiement1	Experiement2	Experiement3	Experiement4	Experiement5 Proposed Algorithm
Minimum Support Value (%)	20 %	30 %	40 %	50 %	50 %
Number of scans	3	3	3	3	1
No of items of size 1	38	25	18	9	9
No of items of size 2	225	69	16	2	36
No of items of size 3	302	20	-	-	84

Table 12
Accuracy of the proposed algorithm

Algorithm	Itemsets of size 1	Itemsets of size 2	Itemsets of size 3
Experiement1	23.86 %	16 %	27.81 %
Experiement2	36%	52.17%	23.08 %
Experiement3	50 %	44.44 %	0 %
Experiement4	100 %	5.55 %	0 %

As show in the previous tables, it is obvious that the proposed algorithm improves the execution time, in comparison with the original Apriori algorithm, by reducing database scan number, which reaches three times in the fourth experiment while reaching one time using the proposed algorithm, thus reducing the database scans by 66.66 %. For the accuracy algorithm, it varies in the four experiments and it increases when the minimum support values increase, as noticed in the four experiments. Figure 5 shows the average calculated accuracy. The average accuracy values for each experiment are listed in the following points:

- For minimum support value 20%, average accuracy is 22.56 %
- For minimum support value 30%, average accuracy is 37.08 %
- For minimum support value 40%, average accuracy is 47.22 %
- For minimum support value 50%, average accuracy is 52.78 %

Also the average accuracy varies for each itemset size, and it decrease when the itemset size increases, as shown in figure 6 and listed in the following points:

- For itemset with size 1, average accuracy is 52.46
- For itemset with size 2, average accuracy is 29.54
- For itemset with size 3, average accuracy is 12.72

However, in the same time all items generated by the proposed algorithm are frequent items, proving that the proposed algorithm can be used as a predictive algorithm for associations with fewer number of database scans than the original Apriori algorithm.

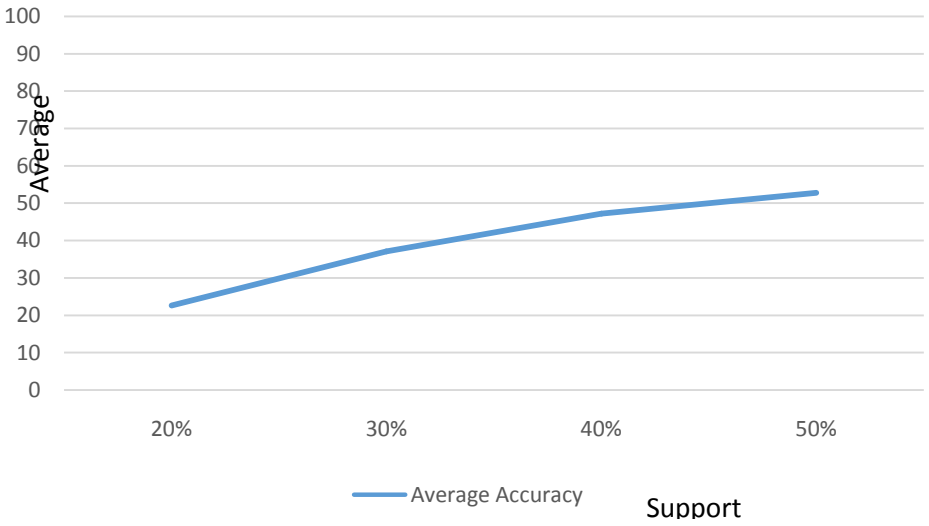

Figure 5 Average accuracy VS minimum support values

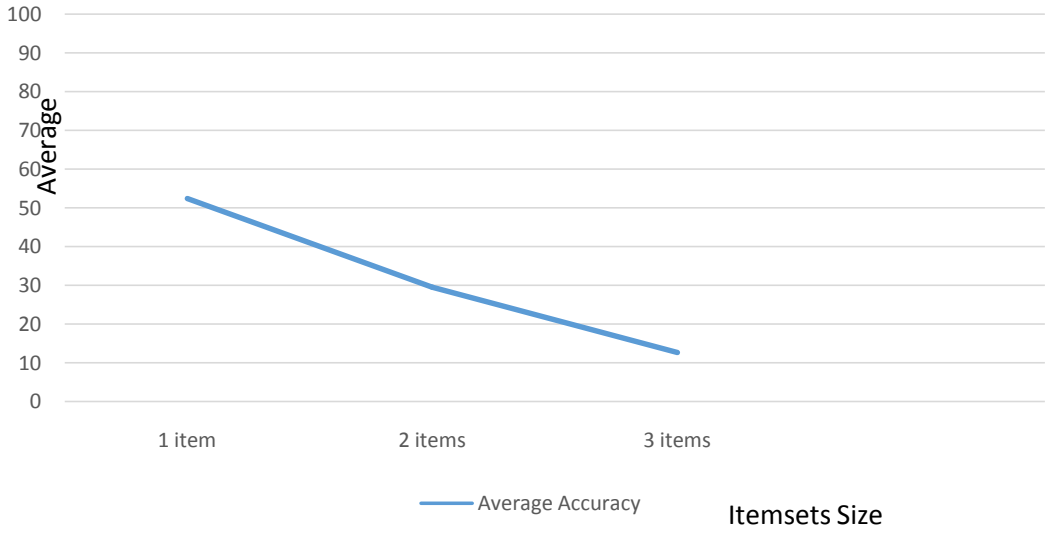

Figure 6 Average accuracy VS itemset size

VI. CONCLUSION

This research proposes an improved Apriori algorithm based on sets operation hypothesis that if the existence of several items exceed than 50% of the total number of transactions, they can form frequent itemsets by combining them. **Thus, items of bigger size can be found and considered as an initial prediction of frequent items of bigger size. This process is done through one database scan.**

The time consumed to generate frequent itemsets in the proposed algorithm is less than the time consumed in the original Apriori; the proposed algorithm reduces the time consumed by 66.66%. The proposed algorithm is useful as a frequent itemsets predictor with lower number of scans, as proved by the experiments. Experimental results reveal the usefulness of the proposed technique.

References

[1] Han, Jiawei, Micheline Kamber, and Jian Pei. Data mining: concepts and techniques. Morgan kaufmann, 2006..`

[2] Bigus, Joseph P., et al. "ABLE: A toolkit for building multiagent autonomic systems." IBM Systems Journal 41.3 (2002): 350-371.

[3] Agrawal, Rakesh, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases." ACM SIGMOD Record. Vol. 22. No. 2. ACM, 1993.

[4] Zaki, Mohammed Javeed. "Scalable algorithms for association mining." Knowledge and Data Engineering, IEEE Transactions on 12.3 (2000): 372-390.

[5] XianWen, Luo, and Wang Weiqing. "Improved Algorithms Research for Association Rule Based on Matrix." Intelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference on. IEEE, 2010.

[6] JI, Wei-dong, Long ZHANG, and Jun ZHANG. "An Improved Apriori Algorithm." Computer Engineering & Science 9 (2009): 024.

[7] Shaoqian, Yu. "A kind of improved algorithm for weighted Apriori and application to Data Mining." Computer Science and Education (ICCSE), 2010 5th International Conference on. IEEE, 2010.

April - 2014, pp. 748-758

[8] Lu, Lin, and Pei-qi Liu. "Stduy on an improved apriori algorithm and its application in supermarket." Information Sciences and Interaction Sciences (ICIS), 2010 3rd International Conference on. IEEE, 2010.
[9] Junfang, Tang. "An improved algorithm of Apriori based on transaction compression." 2011 2nd International Conference on Control, Instrumentation and Automation (ICCIA). 2011.
[10] Nandagopal, S., V. P. Arunachalam, and S. Karthik. "Mining of Datasets with an Enhanced Apriori Algorithm." Journal of Computer Science 8.4 (2012).
[11] Bounsaythip, Catherine, and Esa Rinta-Runsala. "Overview of data mining for customer behavior modeling." VTT Information Technology 18 (2001): 1-53.
[12] Hipp, Jochen, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. "Algorithms for association rule mining—a general survey and comparison." ACM SIGKDD Explorations Newsletter 2.1 (2000): 58-64.
[13] Yun, Hyunyoon, et al. "Mining association rules on significant rare data using relative support." Journal of Systems and Software 67.3 (2003): 181-191.
[14] Patil, Ms Aarti, Ms Seem Kolkur, and Ms Deepali Patil. "Advanced Apriori Algorithms."
[15] Haijun, Zhou Lijuan Li Shuang Geng. "An Improved Apriori Algorithm in Association Rules." Journal of Capital Normal University (Natural Science Edition) (2009): S1.
[16] Zadeh, Lotfi A. "Fuzzy sets." Information and control 8.3 (1965): 338-353.
[17] Swets, John A. "Measuring the accuracy of diagnostic systems." Science 240.4857 (1988): 1285-1293.
[18] Witten IH, Frank E .“Data Mining: Practical machine learning tools and techniques”, 2nd Edition, Morgan Kaufmann, San Francisco. (2005)

