Volume 2 No. 5, May 2012 ISSN 2223-4985

International Journal of Information and Communication Technology Research

http://www.esjournals.org

An Improved Frequent Pattern Algorithm for Mining Association Rules

D. Gunaseelan, P. Uma
College of Computer & Information Systems
JAZAN University, Kingdom of Saudi Arabia

ABSTRACT

Data mining, also known as Knowledge Discovery in Databases (KDD) is one of the most important and interesting research areas in 21st century. Frequent pattern discovery is one of the important techniques in data mining. The application includes Medicine, Telecommunications and World Wide Web. Nowadays frequent pattern discovery research focuses on finding co-occurrence relationships between items. Apriori algorithm is a classical algorithm for association rule mining. Lots of algorithms for mining association rules and their mutations are proposed on the basis of Apriori algorithm. Most of the previous algorithms Apriori-like algorithm which generates candidates and improving algorithm strategy and structure but at the same time many of the researchers not concentrate on the structure of database. In this research paper, it has been proposed an improved algorithm for mining frequent patterns in large datasets using transposition of the database with minor modification of the Apriori-like algorithm. The main advantage of the proposed method is the database stores in transposed form and in each iteration database is filtered and reduced by generating the transaction id for each pattern. The proposed method reduces the huge computing time and also decreases the database size. Several experiments on real-life data show that the proposed algorithm is very much faster than existing Apriori-like algorithms. Hence the proposed method is very much suitable for the discovering frequent patterns from large datasets.

Keywords: *Data mining, frequent pattern mining, transposition of database, Apriori algorithm.*

1. INTRODUCTION

Data mining is one of the most dynamic emerging research in today's database technology and Artificial Intelligent research; the main aim is to discover valuable patterns from a large collection of data for users. In the transaction database, mining association rule is one of the important research techniques in data mining field. The original problem addressed by association rule mining was to find the correlation among sales of different items from the analysis of a large set of super market data. Right now, association rule mining research work is motivated by an extensive range of application areas, such as banking, manufacturing, health care, medicine, and telecommunications. There are two key issues that need to be addressed when applying association analysis. The first one is that discovering patterns from a large dataset can be computationally expensive, thus efficient algorithms are needed. The second one is that some of the discovered patterns are potentially spurious because they may happen simply by chance. Hence, some evaluation criteria are required.

Agrawal and Srikant (1994) proposed the Apriori algorithm to solve the problem of mining frequent itemsets. Apriori uses a candidate generation method, such that the frequent (k+1)-itemset in one iteration can be used to construct candidate (k+1)-itemsets for the next iteration. Apriori terminates its process when no new candidate itemsets can be generated. It is a multi-pass algorithm.

Unlike Apriori, the FP-growth method was proposed by Han et al. (2000) uses an FP-tree to store the frequency information of the transaction database. Without candidate generation, FP-growth uses a recursive divide-and-conquer method and the database projection approach to find the frequent itemsets. However, the recursive mining process may decrease the mining performance and raise the memory requirement.

Most of the reviews are presented in Section 2.2.A lot of algorithms were proposed to optimize the performance of the Apriori-like algorithm. In this research paper it has been presented an efficient and improved frequent pattern algorithm for mining association rules in large datasets. It is a two-pass algorithm.

The remainder of the paper is organized as follows: In Section 2, it has been described in brief an Apriori algorithm, and the relative researches of association rules. In Section 3 provides definitions for the mining method, and detailed steps on the proposed algorithm in mining frequent itemsets. An illustration is demonstrated in Section 4. In Section 5, the design of the experiment and performance analysis is discussed; finally, in Section6 offers conclusions.

2. BACKGROUND

At first, the data mining technique for association rule mining is the support-confidence framework established by Agrawal et al. [AIS 93]. The most important time-consuming part of the association rule algorithm is to discover large itemsets, while the generation of association rules from the given large itemsets is straightforward. This paper has been focused on the discovery of large itemsets. For description, some well-known methods and notions based on this framework is used throughout this paper.

In this section it has been presented the formal statement of association rule mining and the description of Apriori algorithm and related research review.

2.1 Formal Statement of the Problem

The following is a formal statement of association rule mining for transactional databases.

Let $I = \{i_1, i_2, i_3, \ldots, i_n\}$ represents a set of 'n' distinct data items. Generally, a set of items is called an itemset, and an itemset with k items is denoted as a k-itemset. Database D is a set of transactions, where the i^{th} transaction T_i denotes a set of items, such as $T_i \subseteq I$. |D| is the total number of transactions in D, and $|T_i|$ is the number of distinct items in transaction T_i. Each transaction is associated with a unique identifier, which is termed as TID. An association rule is an implication of the form $X \rightarrow Y$, where $X, Y \subseteq I$, and $X \cap Y = \phi$. There are measures of quality for each rule in support of itemset $X \cup Y$ and confidence of rule $X \rightarrow Y$. First, we need to calculate the support of itemset $X \cup Y$, which is the ratio (denoted by s%) of the number of transactions that contain the $X \cup Y$ to |D|. Next, the confidence of rule $X \rightarrow Y$ is the ratio (denoted by c%) of the number of transactions containing $X \cup Y$ to the number of transactions that contain X in database D. The problems of association mining rules from database D can be processed in two important steps: (1) locate all frequent itemsets whose supports are not less than the user-defined minimum support threshold ξ, where $\xi \in (0, 1)$, and, (2) obtain association rules directly from these frequent itemsets with confidences not less than the user-defined minimum confidence threshold. The most time-consuming part of mining association rules is to discover frequent itemsets.

2.2 Review of Apriori Algorithm

In conventional Apriori-like methods, the level wise process of identifying sets of all frequent itemsets is in a combination of smaller, frequent itemsets. In the *k*th level, the Apriori algorithm identifies all frequent *k*-itemsets, denoted as L_k. C_k is the set of candidate *k*-itemsets obtained from L_{k-1}, which are suspected frequent *k*-itemsets. For each transaction in *D*, the candidate *k*-itemsets in C_k contained within the transaction are determined, and their support count is increased by 1/|*D*|. Following scanning (reading) and contrasting with the entire *D*, when the supports of candidate *k*-itemsets are greater than or equal to user-defined minimum support threshold ξ, they immediately become frequent *k*-itemsets. At the end of level *k*, all frequent itemsets of length *k* or less have been discovered. During the execution, numerous candidate itemsets are generated from single itemsets, and each candidate itemset must perform contrasts on the entire database, level by level, while searching for frequent itemsets. However, the performance is significantly affected because the database is repeatedly read to contrast each candidate itemset with all transaction records of the database.

2.3 Related Researches of Association Rules

In 1995, Savasere et al. proposed the partition algorithm to improve the efficiency of Apriori algorithm, it does so by efficiently reducing the number of scans in the database, however, considerable time is still wasted scanning infrequent candidate itemsets [3]. In 1996, Pork et al. proposed an efficient and fast algorithm called DHP (direct hashing and pruning) for the initial candidate set generation. This method efficiently controls the number of candidate 2-itemsets, pruning the size of the database [4]. In 1999, Han et al. proposed a top-down method, which investigates progressively deeper, into the data was developed for the efficient mining of multiple-level association rules from large transactional databases based on the classical Aprioir principle. In 1996, Toivonen proposed a sampling algorithm which reduces the number of database scan to a single scan, but still wastage considerable time on candidate itemsets [9]. In1996, Brid et al. proposed the dynamic itemset count (DIC) algorithm [5] for finding large itemsets, which uses fewer passes over the data than classical algorithms, and yet uses fewer candidate itemsets than methods based on sampling. In addition, in 1999, Dunkel et al. proposed a column-wise apriori algorithm for frequent itemsets and in 2001, Berzal et al. proposed a tree based association rule mining which transformed the storage structure of the data, to reduce the time needed for database scans, improving overall efficiency.

3. PROPOSED ALGORITHM

The proposed algorithm improvement mainly concentrated on (1) for reducing frequent itemset and (2) for reducing storage space as well as the computing time. In the case of large datasets like Wal-Mart datasets, the proposed algorithm is very much useful for reducing the frequent patterns and also reducing the database size for every subsequent passes. For example, in the improved algorithm, the number of occurrence of frequent k-itemsets when k-itemsets are generated from (k-1)-itemsets is computed. If k is greater than the size of the database D, there is no need to scan database D which is generated by (k-1)-itemsets according to the Aprior property and it can be remove automatically.

Transposition of Database

A given database as a relation between original and transposed representations of a database is defined in Table 1. The itemsets are $D = \{I_1, I_2, \ldots, I_n\}$ and transaction ids are $TID = \{T_1, T_2, \ldots, T_m\}$. A string notation for itemsets is used, for example, $I_1I_4I_5$ denotes the itemset $\{I_1, I_4, I_5\}$ and T_2T_4 denotes the transaction ids set $\{T_2, T_4\}$. This dataset is used in all the examples between two sets: a set of items (attributes) and a set of transactions (tuples).

International Journal of Information and Communication Technology Research

http://www.esjournals.org

Table 1: Database D and transposition Database D^T

Transaction IDs	Items
T1	I1, I2, I3
T2	I2, I3, I4
T3	I1, I3, I4
..	…

Items	Transaction IDs
I1	TI, T3
I2	TI, T2
I3	T1, T2, T3
I4	T2, T3

Notations used

Notations	Description
D	Given database
D^T	Transposed database
CT	Candidate transaction IDs
CT_1	Candidate transaction IDs of size-1
LT_1	Large transaction IDs of size-1
CT_{k-1}	Candidate transaction IDs of size-k-1
LT_{k-1}	Large transaction IDs of size-k-1
s	Minimum support
c	Minimum confidence
Count	Frequency

At first, the given transaction database file D is transposed to database D^T and count the number of item and number of transaction string generated for each item and sort the item numbers. Now apply Apriori-like algorithm in which first calculate the frequent transactions CT_1. It reduces infrequent transactions and its item details. For the subsequent passes Apriori-gen has been applied and finds the subsequent frequent transactions.

Lemma 1: All the subsets of a frequent transaction must also frequent. In other words, all the supersets of a frequent transaction must also infrequent.

Improved Algorithmic steps are described as below:

1. First the function *apriori-gen*(LT_{k-1}) is called and to generate candidate k-transaction set by frequent k-transactions.

2. Checking whether candidate transactions *CT* are joined into candidate k-transactions or not. It proceeds by calling function recursively *has_infrequent_transactions*(ct, LT_{k-1}). If it is true, it means the set of transactions are not frequent and should be removed. Otherwise, scan database D^T.

3. The occurrence of frequent k-transaction is computed by generating (k-1)-transactions from k-transactions. If k-transaction is greater than the size of database D^T, it is not needed to scan database D^T which is generated by (k-1)-transactions based on the lemma 1, and it can be deleted.

4. If the size of database D^T is greater than or equal to k, then call function *subset*(CT_k, d^t), which computes frequent pattern using a subsequent iterative level-wise approach based on candidate generation.

Algorithm 1: ***Improved Algorithm***

Input: A transposed database D^T and the user defined minimum support threshold s.
Output: The complete set of frequent patterns

Step 1: Convert Database D into transpose form D^T
Step 2: Compute CT_1 candidate transaction sets of size-1 and finds the support count.
Step 3:Compute the large transaction sets (*LT*) of size-1. (i.e., for all CT_1 is greater than or equal to minimum support.)

LT_1 = {Large 1-transaction set (*LT*)};
For (k=2; LT_{k-1} = 0; k++) do
  Begin
    CT_k = *Apriori-gen*(LT_{k-1},*ct*);
    //new candidate transaction sets
  End
Return $LT = \cup_k LT_k$;

Algorithm 2: ***Apriori-gen*** **(LT_{k-1}), Generate candidate sets**

For all transactions p ∈ LT_{k-1} do begin
For all transactions q ∈ LT_{k-1} do begin
If p.transaction$_1$ = q.transaction,…, p.transaction$_{k-2}$=q.transaction$_{k-2}$, p.transaction$_{k-1}$< q.transaction$_{k-1}$ then begin
ct=p ∞ q;

If *has_infrequent_subset*(*ct*, LT_{k-1}) then
delete ct;
Else
For all transaction set t∈ D^T do begin
If count(t) <k then delete t;
Else begin
Ct=subset(CT_k, t);
End; End

For all candidate transactions ct∈CT_i do begin
CT.count = *CT.count* + 1;

End; End;
$LT_k = \{ct \in CT_k \mid CT.count \geq s\}$;
End; End;
End; End;
Return CT_k;

Algorithms 3: has_infrequent_subset(ct, LT_{k-1})

// checking the elements of candidate generation
For all (k-1)-sub transaction set of ct do
Begin
If $t \in LT_{k-1}$ then return true;
else return false;
End.

The main advantage of the proposed algorithm for frequent patterns discovery are, it reduces the size of the database after second pass and, the storage space and saves the computing time.

4. PERFORMANCE EVALUATION

The following is an example shows the processing steps of the proposed algorithm. Figure 1 shows the original Database D and the transposed database D^T. There are 15 transaction IDs in the database D^T, that is $|D^T| = 9$ and minimum support s = 20%. The improved algorithm for mining frequent patterns in D^T is used.

1. Scan the database D^T for support count of each candidate transactions.

In the first iteration of the improved algorithm, all transaction sets are the member of the set of candidate 1-transactions, CT_1. The proposed method scans all the itemsets in D^T and count the number of occurrences of each itemset.

2. Compute the support count with minimum support.

The user defined minimum support *s* is 20%, that the required support count is 2. Based on the minimum support, we can determine the set of frequent set of 1-transaction IDs(LT_1). That means all the candidate 1-transaction IDs are satisfied with user defined minimum support *s*.

3. Generate all candidate transactions of size-2 i.e., CT_2 from LT_1 and count the support count.

The algorithm generates candidate transactions CT_2 from large transaction set of size-1, LT_1. Compute the number of occurrences in each transaction set by scanning the database D^T. Accumulate the total number of sub-transaction IDs with their support count.

4. Compare the number of occurrences of candidate transaction IDs with their minimum supports.

The Large transaction ID sets of size-2, LT_2 are determined by computing the number of occurrences of each candidate transaction IDs CT_2 with the minimum support s. Based on LT_2, we can determined a new modified transposed database D^T_2.

5. Generate candidate transactions of size-3 from LT_2 by scanning new modified database D^T_2 and finds the support count of CT_3.

First, combine the large transactions of size-2, LT_2 with LT_2 to determine CT_3. Based on the lemma 1, we can determine the four letter candidate transaction IDs C_3 cannot possibly be frequent transactions and therefore prune from CT_3. This is one of the advantages of saving time to count the number of occurrence of transaction IDs unnecessarily during the subsequent scan of D^T_2 for finding LT_3.

6. Compare the support count of candidate transaction IDs with minimum support.

The modified database D^T_2 is scanned by computing LT_3. i.e., the large transaction IDsof size-3, LT_3 are determined by computing the number of occurrences of each candidate transaction IDs CT_3 with the minimum support s.

7. Repeat the steps 4 to 6 until no more candidate transaction IDs are generated.

That is the algorithm terminates, having found all of the frequent transaction IDs. Also, it creates the modified database D^T_3, D^T_4, etc., based the size of the transaction IDs.

The following are the explanation of the proposed algorithm with an example.

Original Database D

Transaction ID	Item ID
T1	1, 14
T2	2, 4, 6, 7, 13, 15
T3	4, 6, 10, 11, 12, 14
T4	2, 3, 6, 13
T5	1, 3, 5, 8, 10, 11, 12, 14
T6	1, 5, 7, 12, 13, 14
T7	3, 5, 7, 10, 11, 12, 13, 14
T8	1, 2, 9, 12
T9	7, 15

Transposed Database $|D^T| = 15$

Item ID	Transaction ID
1	T1, T5, T6, T8
2	T2, T4, T8
3	T4, T5, T7
4	T2, T3
5	T5, T6, T7
6	T2, T3, T4
7	T2, T6, T7, T9
8	T5
9	T8
10	T3, T5, T7
11	T3, T5, T7
12	T3, T5, T6, T7, T8
13	T2, T4, T6, T7
14	T1, T3, T4, T5, T6, T7
15	T2, T9

Now apply the improved algorithm;

Minimum Support (s)=20%

Pass 1: Generate candidates for k=1

C_1={ T1, T2, T3, T4, T5, T6, T7, T8, T9}

C_1	T1	T2	T3	T4	T5	T6	T7	T8	T9
Sup	2	6	6	4	8	6	8	4	2

L_1 = {T2:6, T3:6, T4:4, T5:8, T6:6, T7:8, T8:4}

Pass 2: Generate candidates for k=2

C_2 = {(T2,T3), (T2,T4), (T2,T5), (T2,T6), (T2,T7), (T2,T8), (T3,T4), (T3,T5), (T3,T6), (T3,T7), (T3,T8), (T4,T5), (T4,T6), (T4,T7), (T4,T8), (T5,T6), (T5,T7), (T5,T8), (T6,T7), (T6,T8), (T7,T8) } - 21 candidate sets

After applying improved algorithm, the candidate sets of size 1 and their respected supports are tabulated below:

C_1	T_2T_3	T_2T_4	T_2T_5	T_2T_6	T_2T_7
Sup	2	3	0	2	2

T_2T_8	T_3T_4	T_3T_5	T_3T_6	T_3T_7	T_3T_8
1	2	4	2	4	1

L_2 = {(T2T4):3, (T3T5):4, (T3T7):4, (T4T7):3, T5T6):4, (T5T7):3, (T6T7):3} – 7 large transaction sets only

C_1	T_4T_5	T_4T_6	T_4T_7	T_4T_8	T_5T_6
Sup	2	2	3	1	4

C_1	T_5T_7	T_5T_8	T_6T_7	T_6T_8	T_7T_8
Sup	3	2	4	2	1

Based on L_2, we can prune infrequent transaction sets from the transposed database D^T. After pruning, the new modified transposed database D^T_2 with number of itemsets is 11 only. Previously it was 15.

$$|D^T| = 11$$

Item ID	Transaction ID
1	T1, T5, T6, T8
2	T2, T4, T8
3	T5, T7
5	T5, T6, T7
6	T2, T3, T4
7	T2, T6, T7, T9
10	T3, T4, T5, T7
11	T3, T4, T5, T7
12	T3, T4, T5, T6, T7, T8
13	T2, T6, T7
14	T1, T3, T4, T5, T6, T7

Pass 3: Generate candidates for k=3

C_3 = {(T3 T5 T7), (T5 T6 T7)}

C_3	T3 T5 T7	T5 T6 T7
Support	4	3

L_3 = {(T3 T5 T7), (T5 T6 T7)}

Based on L3, we can prune infrequent transaction sets from the transposed database D^T. After pruning, the new modified transposed database D^T_3 with number of itemsets is 8 only. Previously it was 11.

$$|D^T_3| = 8$$

Item ID	Transaction ID
1	T1, T5, T6, T8
5	T5, T6, T7
7	T2, T6, T7, T9
10	T3, T4, T5, T7
11	T3, T4, T5, T7
12	T3, T4, T5, T6, T7, T8
13	T2,T6, T7
14	T1, T3, T4, T5, T7

Pass 4: Generate candidates for k = 4

C4 = { (T3 T5 T6 T7): 1 }

L4 = {ϕ}

5. EXPERIMENTAL RESULTS

To evaluate the efficiency and effectiveness of the improved algorithm, we performed an extensive study of two algorithms: Apriori-like and improved algorithm, on both real time synthetic data sets with different ranges. All the experiments were examined on Pentium IV machine 1GB RAM, running Microsoft Windows 7. Two algorithms, Apriori and Improved algorithm were implemented in Java 2.0.

Also we got the real time medical database with 2280 itemsets and 4200 elements. The running time comparison between improved algorithm and Apriori algorithm are shown in the Figure 1 with minimum support ranges from 1 percentage (%) to 5 percentages (%).

The importance of improved algorithm is to reduce the number of items in each and every scan and also reduce the size of the original dataset. There are three aspects to make this algorithm better than the original one.

Firstly, when the candidates are being produced, instead of dealing with all the items of the previous large set, only the elements which having the same transaction ids are crossed. At the same time, generating frequent patterns, it may reduce the computing time dramatically and the size of the database is reduced. Secondly, by pruning, the number of elements in the candidate sets is decreased once more based on modified database. Finally, the computing time and storage space are saved.

6. CONCLUSION

In this research paper, it has been proposed an improved algorithm for mining frequent pattern based on Apriori-like algorithm. The main advantages of an improved algorithm are that it can reduce the number of scanning by the transposed database D^T, redundancy by the time of generating sub-transaction set tests and verifying them in the database. In order to discover frequent patterns in massive datasets with more columns than rows, it has been presented a complete framework for the transposition; the item set in the transposed database of the transposition of many classical transactions is given. Also it has been compared the classical Apriori algorithm with an improved algorithm. It has been presented the experimental results, using synthetic data, showing that the proposed algorithm always outperform Apriori algorithm. Hence, the proposed algorithm is very much suitable for a massive datasets.

ACKNOWLEDGEMENTS

The authors are extremely express gratitude to Dr. Omar Sayed Al-Mushayt, College Dean and Dr. Saeed Q Al-Khalidi, Vice Dean, College of Computer Science and Information Systems, JAZAN University, Kingdom of Saudi Arabia for having noble and continuous encouragement to complete this research. The special thanks also to the University President, JAZAN University, Kingdom of Saudi Arabia for inspiration and persistent support directly or indirectly for the completion of this research.

REFERENCES

[1] Agrawal, R. and Srikant, R., 1994. Fast algorithms for mining association rules in large databases. *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, San Francisco, USA, pp. 487-499.

[2] Barabasi, A. and Albert, R., 1999. Emergence of scaling in random networks. *Science*, Vol. 286, pp. 509-512.

[3] Bayardo, R. J., 1998. Efficiently mining long patterns from databases. *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, Seattle, USA, pp. 85-93.

[4] Brin, S. et al, 1997. Dynamic itemset counting and implication rules for market basket data. *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, Tucson, USA, pp. 255-264.

[5] Cooper, C., 2006. The age specific degree distribution of web-graphs. *Combinatorics, Probability and Computing*, Vol. 15, No. 5, pp. 637-661.

[6] Han, J. et al, 2000. Mining frequent patterns without candidate generation. SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, Dallas, USA, pp. 1-12.

[7] Han, J., Kamber, M and Pei. *Data Mining Concepts and Techniques*, The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann Publishers, July 2011.

[8] Purdom, P. W. et al, 2004. Average-case performance of the apriori algorithm. *SIAM J. on Comput.*, Vol. 33, pp. 1223-1260.

[9] Watts, D. J. 2004. The "new" science of networks. *Annual Review of Sociology*, Vol. 30, pp. 243-270.

[10] Zaki, M. J. and Ogihara, M., 1998. Theoretical foundations of association rules. *Proc. 3rd SIGMOD Worksh. Research Issues in Data Mining and Knowledge Discovery*, Seattle, USA, pp. 1-8.

[11] Zheng, Z. et al, 2001. Real world performance of association rule algorithms. *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, San Francisco, USA, pp. 401-406.