Asian Journal of Information Technology 7 (12): 524-530, 2008
ISSN: 1682-3915

Reverse Apriori Algorithm for Frequent Pattern Mining

[1]Kamrul Abedin Tarafder, [2]Shah Mostafa Khaled, [1]Mohammad Ashraful Islam,
[3]Khandakar Rafiqual Islam, [4]Hasnain Feroze, [4]Mohammed Khalaquzzaman and [5]Abu Ahmed Ferdaus
[1]Infrastructure Management (Information Technology), Grameen Phone Ltd, Dhaka 1212, Bangladesh
[2]Institute of Information Technology, University of Dhaka, Dhaka 1000, Bangladesh
[3]HSBC Bangladesh Ltd, Dhaka 1000, Bangladesh
[4]Intelligent Network and Value Added Services,
LM Ericsson Bangladesh Ltd, Dhaka 1213, Bangladesh
[5]Department of Computer Science and Engineering,
University of Dhaka, Dhaka 1000, Bangladesh

Abstract: Association rule mining, one of the most important and well-researched techniques of data mining, investigates the possibility of simultaneous occurrence of data. Given a collection of data, critical analysis can be made on the statistical frequency of data, relationship between different data items and their comparative frequency, etc. This study proposes a new algorithm Reverse Apriori Frequent Pattern Mining, which is a new approach for frequent pattern generation of association rule mining. The proposed algorithm works efficiently, when the number of items in the large frequent itemsets is close to the number of total attributes in the dataset, or if number of items in the large frequent itemsets is predetermined.

Key words: Association rule mining, apriori algorithm, frequent pattern mining, data mining, critical analysis

INTRODUCTION

Data mining (Han and Kamber, 2006) is the process of extracting non-trivial, implicit, previously unknown and potentially useful information from large information repositories such as: relational database, data warehouses, XML repository, etc. The information and knowledge gained can be used for applications ranging from business management, production control, market analysis, to engineering design and science exploration. Simply stated, data mining refers to extracting or mining knowledge from large amounts of data.

Data mining is applicable to different fields of information technology. This includes relational databases, data warehouses, transactional databases, advanced database systems, flat files and the world wide web. There are 2 classes Zhao and Bhowmick (2006) of data mining: descriptive and prescriptive. Descriptive mining is to summarize or characterize general properties of data in data repository, while prescriptive mining is to perform inference on current data, to make predictions based on the historical data. There are various types of data mining techniques such as frequent pattern mining Zhao and Bhowmick (2006) and Koh *et al*. (2008), Classification Zhao and Bhowmick (2006) and Clustering Koh *et al*. (2008a). Frequent pattern mining is a part of association rule mining (Hipp *et al*., 2000). It aims to extract interesting correlations, frequent patterns, associations among sets of items in the transaction databases or other data repositories. Frequent patterns are widely used in various areas such as telecommunication networks, market and risk management, inventory control, etc.

A set of items is referred to as an itemset. An item set that contains k items is called k-itemset. The set (computer, financial_management_software) is a 2-itemset. The occurrence frequency of an itemset is the number of transactions that contain the itemset. This is also known as support count or count of the itemset. Let, itemset $I = I_1, I_2, \ldots, I_k$ be set of k distinct attributes or items. T be a transaction that contains a set of items such that $T \subseteq I$, D be a database with different transaction records T_s. Frequent pattern is a set of items generated from I, where all items satisfy a predefined value. An association rule (Agrawal *et al*., 1993) is an implication in the form of $X \Rightarrow Y$, where, $X, Y \subset I$ are sets of items called itemsets and $X \cap Y = \emptyset$. X is called antecedent while, Y is called consequent, the rule means X implies Y.

Corresponding Author: Kamrul Abedin Tarafder, Infrastructure Management, IT, Grameen Phone Ltd, Dhaka 1212, Bangladesh

There are two important basic measures for association rules, support (s) and Confidence (C) (Yun *et al*., 2003). Support ($X \cup Y$) of an association rule is defined as the percentage or fraction of records that contain $X \cup Y$ to the total number of records in the database. Confidence of an association rule is defined as the percentage or fraction of the number of transactions that contain $X \cup Y$, Y to the total number of records that contain X.

Support, probability of (XY), is defined as the percentage or fraction of records that $X \cup Y$ to the total number of records in the database. The count for each item is increased by one every time the item is encountered in different transaction T in database D during the scanning process. Support (s) is calculated by the following formula Borgelt and Kruse (2002):

$$\text{Support } (X \cup Y) = \frac{\text{Support conut of XY}}{\text{Total number of transaction in D}}$$

An itemset satisfies minimum support if the occurrence frequency of the itemset is greater than or equal to the product of minimum support and the total number of transactions in D. The number of transactions required for the itemset to satisfy minimum support is referred to as the minimum support count. If an itemset satisfies minimum support, then, it is a frequent itemset (Koh and Rountree, 2005). A frequent-k itemset contains k number of items, where all k items satisfy minimum support. A candidate itemset is a set of items, which compete to be a member of frequent itemset (Koh *et al*., 2008). Before the mining process, users can specify the minimum support as a threshold, which means they are only interested in certain association rules that are generated from those itemsets whose supports exceed that threshold. That is, for a rule R, sup (R) $\geq$ minsup and conf (R) $\geq$ minconf, there exists an association rule R (Koh and Rountree, 2005). The data item that satisfies minsup is called frequent and the confidence of rules composed of the frequent data items are investigated to discover the ultimate association rules.

Confidence is a measure of strength of the association rules. Confidence (c), conditional probability of (XY), is defined as the percentage or fraction of the number of transactions that contain $X \cup Y$ to the total number of records that contain X, where if the percentage exceeds the threshold of confidence an interesting association rule $X \Rightarrow Y$ can be generated with the following formula (Borgelt and Kruse, 2002):

$$\text{Confidence } (X \Rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$$

Association rule mining finds out association rules that satisfy the pre-defined minimum support and confidence from a given database. Association rule mining is a two-step process (Zaho and Bhowmick, 2006):

- Finding those itemsets whose occurrences exceed a predefined threshold in the database, these itemsets are called frequent or large itemsets
- Generating association rules from those large itemsets with the constraints of minimal confidence

Suppose one of the large itemsets is L_k, $L_k = \{I_1, I_2, \ldots, I_{k-1}, I_k\}$. Association rules with this itemsets are generated in the following way: the first rule is $\{I_1, I_2, \ldots, I_{k-1}\} \Rightarrow I_k$ by checking the confidence this rule can be determined as interesting or not. Then other rules are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interestingness of them (Bayardo, 1998). This process is iterated until the antecedent becomes empty or no rules satisfy minimum confidence.

Frequent pattern mining finds out frequent patterns that satisfy the predefined minimum support and minimum confidence for a given database (Yun *et al*., 2003). Sometimes even the itemsets are not as frequent as defined by the threshold from frequent pattern generation, the association rules generated from them are still important. For example, in the super market some items are very expensive, consequently, they are not purchased so often as the threshold required, but association rules between those expensive items are as important as other frequently bought items to the retailer. Mining low support but high confidence rules can be important sometimes. We call such rules sporadic because they represent rare cases that are scattered sporadically through the database but with high confidence of occurring together. A top rule is an association rule whose confidence is exactly 100%. Typicallym association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold (Srikant *et al*., 1997).

Frequent pattern discovery technique enables to identify significant rare data associated with specific data in a way that the rare data occur simultaneously with the specific data more frequently than the average co occurrence frequency in the database. The most frequent pattern discovery techniques have identified the association that may happen only among the data that satisfy the support and confidence set by the user's (Srikant *et al*., 1997). In common mechanisms for discovering the frequent patterns, if the support is unique

throughout the whole database leads to assumption that each data included in the database occurs in a similar frequency (Yun *et al*., 2003). In reality, however, the data composing the database may occur either relatively frequently or not. The rarely occurring data in the database may be significant enough to be of good use. Nevertheless, the common frequent pattern discovery techniques do not consider the occurrence frequency pattern of data and discover the association rules using the same support on the whole data, so the discovered rules with regard to rare data may be redundant and as a result, unnecessary rules may be generated.

In this study, a new algorithm Reverse Apriori Frequent Pattern Mining for discovery of frequent itemsets has been proposed. This algorithm starts to find large frequent itemsets considering maximum number of items at first time. If it fails to discover a frequent itemset with the largest number of items, it considers 1 less than maximum number of items. In this way, this algorithm finds large frequent itemsets starting from higher number of items to gradually less number of items. The proposed algorithm works efficiently when the number of items in the large frequent itemsets is close to the number of total attributes in the dataset, or if number of items in the large frequent itemsets is predetermined, which is close to the total number of attributes in the dataset.

COMMON APPROACHES OF FREQUENT PATTERN MINING

There are different algorithms and approaches for frequent pattern discovery. Techniques to discover the association among data, such as AIS (Zhao and Bhowmick, 2006), SETM (Zhao and Bhowmick, 2006) and Apriori (Borgelt and Kruse, 2002) have been widely studied.

Apriori is a great improvement in the history of association rule mining, Apriori algorithm was first proposed by Agrawal *et al*. (1993). The AIS is just a straightforward approach that requires many passes over the database, generating many candidate itemsets and storing counters of each candidate while most of them turn out to be not frequent. Apriori is more efficient during the candidate generation process for two reasons, Apriori employs a different candidates generation method and a new pruning technique.

There are two processes to find out all the large itemsets from the database in Apriori algorithm. First the candidate itemsets are generated, then the database is scanned to check the actual support count of the corresponding itemsets. During the first scanning of the database the support count of each item is calculated and the large 1-itemsets are generated by pruning those itemsets, whose supports are below the pre-defined threshold. In each pass only those candidate itemsets that include the same specified number of items are generated and checked. The candidate k-itemsets are generated after the k-1th passes over the database by joining the frequent k-1-itemsets. All the candidate, k-itemsets are pruned by check their sub (k-1)-itemsets, if any of its sub (k-1)-itemsets is not in the list of frequent (k-1)-itemsets, this k-itemsets candidate is pruned out because it has no hope to be frequent according the Apriori property. The Apriori property says that every sub (k-1)-itemsets of the frequent k-itemsets must be frequent. Pseudo code in presents Apriori algorithm as following:

Input:
Database D
Minimum Support ε
Minimum Confidence ξ

Output:
R_t All association Rules

Method:
01 $L1$ = Large 1-itemsets
02 for ($k = 2$; $L_{k-1} \neq \varnothing$; k++) do begin
03 C_k = apriori-gen (L_{k-1});//generate new candidates from L_{k-1}
04 for all transactions $T \in D$ do begin
05 C_t = subset (C_k,T); //candidates contained in T
06 for all candidates $C \in C_t$ do
07 Count (C) = Count (C)+1; // increase support count of C by 1
08 End
09 $L_k = \{C \in C_t \mid \text{Count } (C) \geq \in \times |D|\}$
10 End
11 $L_r = U_k L_k$;
12 R_t = Generate Rules (L_r, ξ)

In the process of finding frequent itemsets, Apriori avoids the effort wastage of counting the candidate itemsets that are known to be infrequent. The candidates are generated by joining among the frequent itemsets level-wisely, also candidate are pruned according to the Apriori property. As a result the number of remaining candidate itemsets ready for further support checking becomes much smaller, which dramatically reduces the computation, I/O cost and memory requirement.

Apriori algorithm still inherits the drawback of scanning the whole databases many times. Based on Apriori algorithm, many new algorithms were designed with improvements. Generally, there were two approaches: one is to reduce the number of passes over the whole database or replacing the whole database with only part of it based on the current frequent itemsets, another approach is to explore different kinds of pruning techniques to make the number of candidate itemsets smaller. Apriori-TID and Apriori-Hybrid (Agrawal *et al*., 1993), DHP (Park *et al*., 1995), SON (Savesere *et al*., 1995) are modifications of the Apriori algorithm.

Most of the algorithms introduced above are based on the Apriori algorithm and try to improve the efficiency by making some modifications, such as reducing the number of passes over the database; reducing the size of the database to be scanned in every pass; pruning the candidates by different techniques and using sampling technique. However, there are two bottlenecks of the Apriori algorithm. One is the complex candidate generation process that uses most of the time, space and memory. Another bottleneck is the multiple scan of the database.

REVERSE APRIORI FREQUENT PATTERN MINING

Apriori algorithm collects candidate itemsets. A candidate itemset includes the items that have possibility to be member of the frequent itemsets and is used to discover frequent itemsets. It then discovers the frequent itemsets from the candidate itemsets (Zhao and Bhowmick, 2006).

Apriori at first finds candidate-1 itemsets. It then finds frequent-1 itemsets by pruning candidate-1 itemsets. The pruned items are those which don't satisfy minimum support value. Then, it generates candidate-2 itemsets from frequent-1 itemsets. From candidate-2 itemsets, Apriori generates frequent-2 itemsets in the same process as it generated frequent-1 itemsets. This process continues until large frequent itemsets are generated. Large frequent itemsets Zhao and Bhowmick (2006) are those from which candidate itemsets cannot be generated, that means no item of the candidate itemsets satisfies the user defined minimum support value. Thus, Apriori generates large frequent itemsets, where number of items in the candidate set starts from 1 and gradually increases to bigger number.

The algorithm proposed in this study works in the reverse way of Apriori algorithm. The proposed approach is different from Apriori in that it generates large frequent itemsets starting from considering maximum possible number of items in the data set. It generates this large frequent itemsets only if it satisfies user specified minimum item support. It then gradually decreases the number of items in the itemsets until it gets the largest frequent itemsets. At first, the proposed algorithm checks for large frequent itemsets with all the combinations of the distinct values for all the items in the target dataset. If it is satisfied by minimum support value, then large frequent itemsets is revealed at first checking. If it is not satisfied, then checks for next large frequent itemsets by combinations of next large number of items and thus generates large frequent itemsets by checking the minimum support value. The algorithm is presented in pseudo code of Reverse Apriori algorithm as following:

```
Input:
        Database D
        Minimum Support ε

Output:
        Large Frequent Itemsets

Method:
01      k = maximum number of attributes
02      i = 0
03      for all combinations of (k-i) number of attributes
04      do
05      generate candidate-k-i itemsets
06      generate frequent itemsets from candidate-k-i itemsets
07      where, support count of generated itemsets > = ε
08      If successful, then go to step 10
09      Else i = i+1 and go to step 03
10      Return sets of large frequent itemsets
11      End
```

Whenever, reverse Apriori algorithm finds frequent itemsets, it does not go to next searching step to check larger frequent itemsets like Apriori does. In steps 01-02, total number of attributes is calculated and is kept in a variable k. An incremental variable i is initialized to 0. In steps 03-07, for all combinations of k number of attributes, every combination is checked by predefined support count. If it satisfies the support value, then the combination is a member of frequent itemsets. And the itemsets is the large frequent itemsets. If steps 03-07, fail to generate large frequent itemsets, then step 09 increments i by 1 and jumps to step 03. This process continues until large frequent itemsets is generated. When a large frequent itemsets is generated, then the program jumps to step 10. Step 10 shows the output and exits.

EXPERIMENTAL SETUP AND PERFORMANCE ANALYSIS

The synthetic dataset that is used for performance analysis has been collected from WEKA (Witten and Frank, 2005). WEKA is a software developed by the University of Waikato, Hamilton, New Zealand. The synthetic dataset has 5 attributes. There are 12 distinct items for five attributes. Table 1 presents the names and values of items in synthetic dataset. For performance analysis, Quest software TOAD, version 9.0.1 has been used. The configuration of the machine where experiments have been done is-Intel Pentium (R) Dual 2.0 GHz, 1 GB of RAM. The performance of the new algorithm Reverse Apriori has been analyzed compared with Apriori algorithm, with respect to number of table scan, row counts and execution time. For the convenience of understanding, the sample dataset used is given in Table 2.

Minimum support value of 3 has been used in this study. Using Apriori algorithm in the example dataset, the frequent itemsets are generated. At first pass, candidate-1 and frequent-1 itemsets are generated. In second pass, candidate-2 and frequent-2 itemsets are generated. In the third pass, candidate-3 and frequent-3 itemsets are generated. Though frequent-3 itemsets is the large frequent itemsets for the example dataset, but after pass 3 there is no way to understand frequent-3 itemsets is the large frequent itemsets. That is why Apriori generates candidate-4 itemsets. As no items of candidate-4 itemsets satisfies minimum support value, now it is clear that frequent-3 itemsets is the large frequent itemsets. The total number of item counts at each scan is as in Table 3.

Applying Reverse Apriori in the example dataset to generate frequent itemsets, at first pass candidate-5 itemsets is generated. As no member of the candidate-5 itemsets is frequent, in second pass, candidate-4 itemsets is generated. There are also no frequent items in the candidate-4 itemsets. So, in the third pass, candidate-3 itemsets is generated. There are 4 items in candidate-3 itemsets, which satisfy predefined support value 3. So, frequent-3 itemsets is the large frequent itemsets for the example dataset. The total number of item counts at each scan is shown in Table 4.

In study of Apriori, large frequent itemsets were generated after fouth pass of database, though the large frequent itemsets is itemsets-3. Here is an advantage of Reverse Apriori approach over Apriori. Reverse Apriori stops the scanning as soon as it gets an itemsets having items which satisfy minimum support value. A comparative study total generated rows between Apriori and Reverse Apriori is shown in the Fig. 1.

In Fig. 1, all generated rows for candidate and frequent itemsets are shown. Here, it is seen that Apriori and Reverse Apriori converges to the same point for frequent itemsets-3, which is the largest frequent itemsets. Reverse Apriori proceeds to generate candidate itemsets until it can generate an frequent itemsets having some output.

In Fig. 1, Reverse Apriori shows 0 frequent itemsets for candidate-5 and candidate-4 itemsets and shows 4 frequent itemsets for frequent-3 itemsets and it stops at this point. Apriori takes fourth scan to find large frequent itemsets, whereas, Reverse Apriori takes third scan to find large frequent itemsets. The comparison is shown in Fig. 2.

Though number of generated rows in Apriori is less than that of Reverse Apriori, total execution time of Apriori is more than that of Reverse Apriori. The execution time of Apriori and Reverse Apriori is shown in Fig. 3.

Apriori generates fewer rows than Reverse Apriori. Apriori takes more table scan to find large frequent itemsets than Reverse Apriori. That is why, the execution time of Apriori is more than that of Reverse Apriori. The comparison of execution time between Apriori and Reverse Apriori is shown in Fig. 3.

From Fig. 3, it is observed that though number of generated rows in Reverse Apriori is more than Apriori, but execution time is more in Apriori than Reverse Apriori. It is also observed that number of database scan in Apriori is more than that of Reverse Apriori. So, it can be concluded that the execution time is dependent on number of database scan. In Table 5 a comparison of execution time, total database scan and total generated rows between Apriori and Reverse Apriori is shown.

In Table 5, from the relationship of total database scan, total generated rows and total execution time between Apriori and Reverse Apriori, it can be said that the execution time depends on the total number of database scan.

Table 1: Names and values of items in synthetic dataset

Header of items	Values of items
Outlook	Sunny, overcast, rainy
Temperature	Hot, mild, cool
Humidity	High, normal
Windy	False, true
Play	Yes, no

Table 2: The sample synthetic dataset

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Table 3: Apriori row counts and execution time for example dataset

No. rows and execution time for different candidate and frequent sets	1		2		3		4		Total generated rows	Total execution time
	Candidate	Frequent	Candidate	Frequent	Candidate	Frequent	Candidate	Frequent		
No. generated rows	12	12	57	26	24	4	11	0	146	-
Execution time (msec)	16	15	31	16	47	15	31	15	-	186

Table 4: Row counts and execution time for reverse apriori

No. rows and execution time for different candidate and frequent sets	5		4		3		Total generated rows	Total execution time
	Candidate	Frequent	Candidate	Frequent	Candidate	Frequent		
No. generated rows	72	0	156	0	124	4	356	-
Execution time (m sec)	32	15	32	16	32	15	-	142

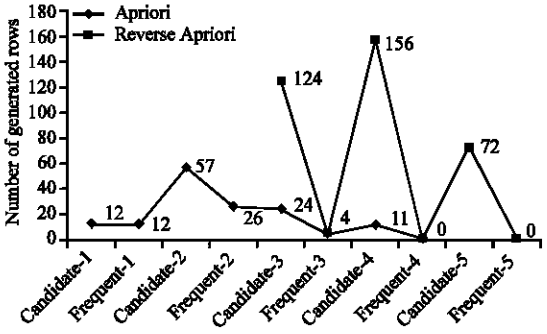

Fig. 1: Comparison of generated rows between apriori and reverse apriori

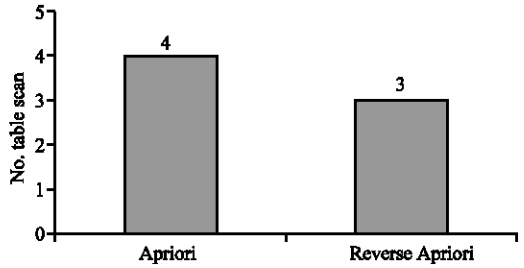

Fig. 2: Total table scan between apriori and reverse apriori

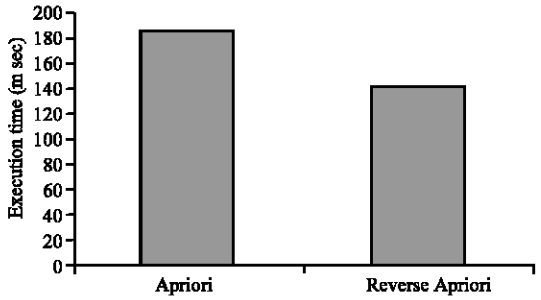

Fig. 3: Execution time between apriori and reverse apriori

Table 5: Comparison between apriori and reverse apriori

Algorithms	Total database scan	Total generated rows	Execution time (msec)
Apriori	4	146	186
Reverse Apriori	3	356	142

It is evident from Table 5 that execution time is not dependent on number of generated rows; rather it is dependent on number of database scans. Apriori and Reverse Apriori can be used in the convenient way, where either one accomplishes its task in fewer database scans.

DISCUSSION

Research under the broad area of data mining is still ongoing and the importance of data interpretation for decision making is becoming more and more relevant, especially in the recent few years when storage hardware has become cheap and different business and research entities are storing all possible types of data for analysis.

The focus of research has been being more and more pointed towards finding algorithms that can find out worthy data relations in minimum time.

This study proposed Reverse Apriori Frequent Pattern Mining algorithm to generate large frequent itemsets. The proposed algorithm finds interesting correlations between data more efficiently than Apriori. The proposed algorithm can be helpful in large databases where large frequent itemsets contain large number of attributes.

There is certain advantage of starting to generate large frequent itemsets from considering maximum number of attributes to gradually less number of attributes. The advantage is that the approach always automatically satisfies downward closure property (Srikant *et al*., 1997). Downward closure property states that all subsets of a frequent set must also be frequent.

There is no need to check this property in the proposed algorithm. Because, if itemsets (a-d) with support value 3 is generated, then any subsets of this set have at least support 3. Support value of (a-d) is 3. The combinations of a-d; such as a-c or a, b or a, c or a, d or b, c or c, d is also having support value of at least 3. Whenever, Reverse Apriori Frequent Pattern Mining finds frequent itemsets, it stops further searching.

This is because, it starts checking for frequent itemsets with the maximum number of items and gradually, it considers less number of items until frequent itemsets are generated. As a result, whenever it gets an itemsets having frequent items, no doubt it is the large frequent itemsets.

But this is not true for Apriori approach. Whenever, Apriori finds a frequent itemsets, it proceeds to next step to ensure whether, there is a larger frequent itemsets or not. Thus, the proposed approach has the advantage of less number of table scan compared with that of Apriori algorithm.

Reverse Apriori approach is not suitable if large frequent itemsets are found with less number of attributes

with respect to total number of attributes. For example, there are 50 attributes in a table and large frequent itemsets are found with 3-5 attributes.

In such a situation, Reverse Apriori approach is not suitable. The reason is starting to check large frequent itemsets from combinations of 50 attributes and gradually reaching towards 3-5 is expensive. After a lot of scanning, this approach will get desired output.

However, the proposed approach works fine, if number of attributes in the large frequent itemsets is predetermined to large value, or large frequent itemsets are found close to the number of attributes.

For example, if the important relationships among sets of data will have to be found having number of items >40, where number of attributes is 50 in the table, then Reverse Apriori approach works well, or if interesting relationships among data are generated, where number of items is very close to the total number of attributes, then Reverse Apriori Frequent Pattern Mining approach works well.

REFERENCES

Agrawal, R., T. Imielinski and A.N. Swami, 1993. Mining association rules between sets of items in large databases. In Proc., ACM SIGMOD International Conference on Management. California, USA, 9/6/2003-12/6/2003, pp: 207-216. http://doi.acm.org/10.1145/170035.170072. http://portal.acm.org/citation.cfm?doid=170035.170072.

Bayardo, R.J., 1998. Efficiently mining long patterns from databases. In Proc. ACM-SIGMOD International Conference of Management. Seattle, Washington, USA, Data, 02/06/1998-04/-06/1998, pp: 85-93. http://portal.acm.org/citation.cfm?id=276304.276313.

Borgelt, C. and R. Kruse, 2002. Induction of association rules: Apriori implementation. In Proc., 15th Compstat Conf., Berlin, Germany, Elsevier Science Publishers B.V., Amsterdam, 24/08/2002-28/08/2002, pp: 4712-4721. DOI: 10.1016/j.csda.2008.03.013. http://www.borgelt. net/papers/cstat_02.pdf.

Han, J. and M. Kamber, 2006. Data Mining-Concepts and Techniques. 2nd Edn. Morgan Kaufmann, San Francisco, CA. ISBN: 1-55860-489-81. http://www.cs.uiuc.edu/homes/hanj/bk2/.

Hipp, J., U. G¨untzer and G. Nakhaeizadeh, 2000. Algorithms for association rule mining. A general survey and comparison. ACM SIGKDD Explorations Newsletter, 2 (1): 58-64. DOI: http://portal.acm.org/citation.cfm?id=360421.

Koh, Y.S. and N. Rountree, 2005. Finding sporadic rules using apriori-inverse. Advances in Knowledge Discovery and Data Mining, 3518: 97-107. DOI: 10.1007/b136725. http://cat.inist.fr/?aModele=afficheN&cpsidt=16894993.

Koh, Y.S., N. Rountree and R. O'Keefe, 2008. Finding non-coincidental sporadic rules using apriori-inverse. Int. J. Data Warehous. Mining, 2 (2): 38-54. http://www.igi-global.com/articles/details.asp?ID=5745.

Koh, Y.S., N. Rountree and R.A. O'Keefe, 2008a. Mining interesting imperfectly sporadic rules. Knowledge and information systems. Springer, London, 14 (2): 179-196. DOI: 10.1007/s10115-007-0074-6. http://www.springerlink.com/content/g3363477r71768k1/fulltext.pdf.

Park, J.S., M.S. Chen and P.S. Yu, 1995. An effective hash based algorithm for mining association rules. In Proc. ACM SIGMOD Int. Conf. Management of Data. San Jose, California, pp: 175-186. http://doi.acm.org/10.1145/223784.223813. http://portal.acm.org/citation cfm?id=223784.223813.

Savesere, A., E. Omiecinski and S. Navathe, 1995. An efficient algorithm for mining association rules in large databases. In Proc. 21st International Conference of VLDB, Trondheim, Norway, Morgan Kaufmann, San Francisco, CA, USA. 30/08/1995-02/09/1995, pp: 432-444. DOI: 10.1.1.103.5437. http://www.vldb.org/conf/1995/P432.PDF.

Srikant R., Q. Vu and R. Agrawal, 1997. Mining association rules with item constraints. In Proc. Third International Conference of Knowledge Discovery in Databases and Data Mining (KDD), AAAI Press, California, USA, 14/08/1997-17/08/1997, pp: 67-73. http://almaden.ibm.com/cs/projects/iis/hdb/Publications/papers/kdd97_const.pdf.

Witten, I.H. and E. Frank, 2005. Data Mining: Practical Machine Learning tools and Techniques. 2nd Edn. Morgan Kaufmann. San Francisco, USA. ISBN: 0-12-088407-0 http://www.cs.waikato.ac.nz/ml/weka/index_citing.html. http://www.cs.waikato.ac.nz/~ml/weka/book.html.

Yun, H., H. Danshim, B. Hwang and K.H. Ryu, 2003. Mining association rules on significant rare data using relative support. J. Syst. Software, Elsevier, 67 (3): 181-191. DOI: 10.1016/S0164-1212(02)00128-0. http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V0N-48H8BRX-D-3W&_cdi=5651&_user=1960180&_orig=search&_coverDate=09%2F15%2F2003&_sk=999329996&view=c&wchp=dGLbVzW-zSkWz&md5=a85001 2b663a5b6ee14692248a4c0264&ie=/sdarticle.pdf.

Zhao, Q. and S.S. Bhowmick, 2006. Association rule mining: A survey (MS Thesis), Nanyang Technological University, Singapore. http://thesis.lib.ncu.edu.tw/ETD-db/ETD-search/view_etd?URN=944203006.