A New Approach of Eliminating Redundant Association Rules

Mafruz Zaman Ashrafi, David Taniar, and Kate Smith

School of Business Systems, Monash University, Clayton, Vic 3800, Australia
{Mafruz.Ashrafi,David.Taniar,Kate.Smith}@infotech.monash.edu.au

Abstract. Two important constraints of association rule mining algorithm are support and confidence. However, such constraints-based algorithms generally produce a large number of redundant rules. In many cases, if not all, number of redundant rules is larger than number of essential rules, consequently the novel intention behind association rule mining becomes vague. To retain the goal of association rule mining, we present several methods to eliminate redundant rules and to produce small number of rules from any given frequent or frequent closed itemsets generated. The experimental evaluation shows that the proposed methods eliminate significant number of redundant rules.

1 Introduction

Association rule mining is an iterative and interactive process that explores and analyzes voluminous digital data to discover valid, novel and meaningful rules, using computationally efficient techniques. It searches for interesting relationships among items in a given dataset. The main advantage of association rule mining is that it has ability to discover hidden associations with in the digital data.

Two important constraints of association rule mining are *support* and *confidence* [1]. Those constraints are used to measure the interestingness of a rule. Therefore, most of the current association rule mining algorithms use these constraints in generating rules. However, choosing support and confidence threshold values is a real dilemma for association rule mining algorithms. For example, discovering association rules with high support threshold removes rare item rules without considering the confidence value of these rules. On the other hand, when support threshold is low, it generates large number of rules, and consequently it becomes very difficult, if not impossible, for end user to utilize these rules.

It is widely recognized that number of association rules grows as number of frequent itemsets increases. In addition, most of the traditional association rules mining algorithms consider all subsets of frequent itemsets as antecedent of a rule [7]. Therefore, when resultant frequent itemsets is large, these algorithms produce large number of rules. However, many of these rules have identical meaning or are redundant. In fact, the number of redundant rules is much larger than the previously expected. In most of the cases, number of redundant rules is significantly larger than that of essential rules [4]. In many cases such enormous redundant rules often fades away the intention of association rule mining in the first place.

F. Galindo et al. (Eds.): DEXA 2004, LNCS 3180, pp. 465 - 474, 2004.

To reduce redundant rules, there are number of frameworks that have been proposed [4-8]. Most of the proposed frameworks have several prior assumptions. Based on these assumptions, the frameworks identify redundant rules and prune them subsequently. However, these prior assumptions are not suitable in many situations, and subsequently redundant rules may still exist in the resultant rule set. Furthermore, some of the proposed frameworks [4, 8] mark rule r as redundant and eliminate it, in the presence of another rule R (consider r, $R \in R'$, where R' is resultant ruleset) without considering whether rule R characterizes the knowledge of rule r. For example, an algorithm will mark rule $AB \Rightarrow C$ as redundant in the presence of rule $A \Rightarrow C$. However, it is apparent from this example that rule $A \Rightarrow C$ is not fully characterized the knowledge of rule $AB \Rightarrow C$.

In order to eliminate redundant rules, in this paper, we propose several methods that remove redundant rules from the resultant ruleset without losing any important knowledge. We are motivated by the fact that one can only utilize association rules efficiently when resultant rule set is small in size. However, without an efficient redundant rule reduction technique, one cannot achieve this goal. The proposed methods mark a rule as redundant when it finds a set of rules that also convey the same knowledge. For example, the proposed method will mark rule $A \Rightarrow BC$ as redundant, if and only if the rule such as $A \Rightarrow B$ and $A \Rightarrow C$ are present in that set. Our experimental evaluation shows that the proposed method generates only small number of rules. Therefore it becomes very convenient for end users to make use of this knowledge.

The rest of paper is organized as follows: In Section 2 we describe the background of association mining and summarize the reasons of rule redundancy. We describe related work in section 3. In Section 4 we present our proposed algorithms. The performance evaluation and comparison are described in section 5 and we conclude at Section 6.

2 Redundant Rules: A Background

One of the key issues of association rule mining is redundant rule [4-8]. Before further discussion on redundant rules is carried out, let us briefly discuss some of the key tasks of association rule mining algorithms.

2.1 Association Rule Mining: A Brief Overview

Algorithms for association rule mining usually have two distinct phases *(i) frequent itemset* and *(ii) rule generation* [2]. In order to find all *frequent itemsets*, we need to enumerate the support of all itemsets of the database. The support of an itemset can be defines as follows:

Definition: Let $\mathcal{D}$ be a transaction database has n number of items and I is a set of items such that $I = \{a_1, a_2, a_3, \ldots\ldots a_n\}$, where $a_i \subset n$. Consider N be the total number of transactions and $T = \{t_1, t_2, t_3, \ldots\ldots t_N\}$ be the sequence of transaction, such that $t_i \subset \mathcal{D}$. The support of each element of I is the number of transactions in $\mathcal{D}$ containing I and for a given itemset $A \subset I$.

Itemset A is *frequent* if and only if $support(A) \geq minsup$ where *minsup* is a user-defined support threshold value. However, the enumeration of itemsets is computationally and I/O intensive [1]. For example, if the database has m number of distinct items, the search space for enumerating all frequent itemsets is 2^m. If m is large, then generating support for all itemsets requires long period of time and may exhaust the main memory limit.

Since frequent itemset generation is considered as an expensive operation, an approach known as Frequent Close Itemset (*FCI*) [7] was introduced. Itemset A is *closed* if there exists no itemset A' such that A' is a proper superset of A and all transactions containing A also contain A'. The total number of frequent *closed* itemsets generated from a given dataset is smaller than the total number of frequent itemsets, especially when the dataset is dense and have enough information so one can generate association rules from it [7]. For example, if "*B*" and "*B C*" are two frequent itemsets that have occurred in the same number of times in a given dataset, then only the itemset "*B C*" will be considered as FCI.

In the frequent itemset approach, *rule generation* task is relatively straightforward. Association rule mining algorithms use *frequent* itemsets in order to generate rules. An association rule R is an implication of two frequent itemsets $F_1, F_2 \in I$; such that $F_1 \cap F_2 = \{\}$ and can be expressed as $F_1 \Rightarrow F_2$.

Using the FCI approach, association rules are generated in the same manner as it does for the frequent itemsets. However, when the rules are generated from FCI, there is a chance that it will not consider some of the important rules. For example, in a given dataset if itemsets such as "*A*", "*A B*","*A C*" and "*A B C*" have the same support, then the *closed* itemset will only consider itemset "*A B C*" as *frequent*. Therefore one cannot generate $A \Rightarrow B$, $A \Rightarrow C$ or $A\ B \Rightarrow C$, even though these rules have very high confidence value.

2.2 Overview of Rule Redundancy

The frequent itemsets based association rule mining framework produces large a number of rules, because it considers all subsets of frequent itemsets as antecedent of a rule. Therefore, the total number of rules grows as the number of frequent itemsets increases. Number of redundant rules is larger than the previously suspected and often reaches in such extend that sometimes it is significantly larger than number of essential rules.

Rule	Support	Confidence
$X \rightarrow YZ$	$S(X \cup Y \cup Z)$	$S(X \cup Y \cup Z)/S(X)$
$XY \rightarrow Z$	$S(X \cup Y \cup Z)$	$S(X \cup Y \cup Z)/S(X \cup Y)$
$XZ \rightarrow Y$	$S(X \cup Y \cup Z)$	$S(X \cup Y \cup Z)/S(X \cup Z)$
$X \rightarrow Y$	$S(X \cup Y)$	$S(X \cup Y)/S(X)$
$X \rightarrow Z$	$S(X \cup Z)$	$S(X \cup Z)/S(X)$

Fig. 1. Redundant Rules

Consider five different rules generated from a frequent itemset '*XYZ*' at a given support and confidence threshold s and c as shown in the figure 1. However, Aggarwal et al. [4] argue that if rule $X \Rightarrow YZ$ meet s and c, then rules such as $XY \Rightarrow Z$,

$XZ \Rightarrow Y$, $X \Rightarrow Y$, and $X \Rightarrow Z$ are redundant. This is because the support and confidence values $X \Rightarrow YZ$ are less than the support and confidence values for the rules $XY \Rightarrow Z$, $XZ \Rightarrow Y$, $X \Rightarrow Y$, and $X \Rightarrow Z$.

Furthermore, by observing at abovementioned scenario one may think that if the FCI method is used instead of frequent itemset, then one can avoid those redundant rules. However, the FCI will also consider those itemset as frequent if itemsets "*X*" "*XY*" "*XZ*" and "*XYZ*" do not occur the same number of times in the dataset. Therefore we cannot avoid these kinds of redundant rules, should we generate rules by using FCI. Based on this rationale, we can define redundancy rules as follows:

Definition: In the context of association rule mining, a set of rules $\Re$ which is generated from a set of frequent itemsets $\Im$, such that each element $r \in \Re$ satisfy both *support* and *confidence* thresholds. A rule r in $\Re$ is said to be ***redundant*** if and only if a rule or a set of rules S where $S \in R$ possess same intrinsic meaning of r.

For example, consider a rule set $\Re$ has three rules such as $milk \Rightarrow tea$, $sugar \Rightarrow tea$, and $milk, sugar \Rightarrow tea$. If we know the first two rules i.e. $milk \Rightarrow tea$ and $sugar \Rightarrow tea$, then the third rule $milk, sugar \Rightarrow tea$ becomes redundant, because it is a simple combination of the first two rules and as a result it does not covey any extra information especially when the first two rules are present.

2.3 Why Does Redundancy Occur?

It is very important to have a clear idea how current support and confidence based association mining algorithms work in order to identify the reasons that cause redundancy. In the current approach, first we enumerates all frequent itemsets, and then we find all candidate rules (i.e. rules those confidence value are not verified) by combining subsets of every frequent itemset in all possible way.

The validity of association rules (i.e. rules that meet a given *confidence* value) is verified simultaneously during the time of candidate rules generation. All traditional association rule mining algorithms subsequently prune away candidate rules that do not meet the *confidence* threshold. Due to the fact that the traditional algorithms generate association rules in two phases based on *support* and *confidence* values, they generate a large number of rules especially when the user-specified threshold values of support and confidence are low. It is worth to mention that the total number of subset elements grows proportionally as the length of frequent itemset increases. Consequently the number of rules may increase unwieldy when the average length of frequent itemset is long.

From the above discussions one may think that if user-specific *support* and *confidence* threshold values are high, the rule redundancy problem should be solved. Indeed such assumption reduces the ratio of redundant rules but is not able to eliminate redundant rules completely. Because traditional algorithms find association rules based on confidence value, they consider all candidate rules as valid rules when these rules have confidence above the threshold value. Nevertheless from a frequent itemset we can construct different rules. And many of these rules may meet the high support and confidence values. However, when a frequent itemset generates many valid rules then without a doubt we can say that many of those rules will fall under redundant rules category. For example a simple rule ($A \Rightarrow B$) can be represent in two

different ways ($A \Rightarrow B$, $B \Rightarrow A$), if we interchange the antecedent and confidence itemset. Furthermore the confidence value is changed as we interchange the *antecedent* and *consequence* itemset of a rule. The traditional approach only checks whether the confidence is above the user-specified threshold or not. Swapping the *antecedent* itemset with *consequence* itemset of a rule will not give us any extra information or knowledge.

However, choosing the support and confidence threshold values is a real dilemma for all association rules mining algorithms. If the *support* value is set too high, we will not find rules from rare itemsets although there might be some rules with very high confidence. On the other hand when the support and confidence threshold value is low then one can find many rules that do not make any sense.

3 Related Work

One of the main drawbacks of association rule mining is redundant rule. To overcome redundant rules a number of research works [4-7] had been found in the data mining literature. In this section, we discuss some of the previously proposed redundant rule reduction techniques.

Aggarwal et al. [4] classify the redundant rule in two groups, such as: *simple redundant* and *strict redundant*. The authors proposed different methods to remove redundant rules. They proposed that a rule bears *simple redundancy* in the presence of other rules if and only if those rules are generated from same frequent itemset and the support values for the those rules are the same but the confidence value for one of them is higher than the others. For example, the rule $AB \Rightarrow C$ bears simple redundancy with respect to rule $A \Rightarrow BC$. But this approach recognizes rule $BC \Rightarrow A$ as non-redundant with respect to $AB \Rightarrow C$ because item $BC \not\subset AB$. Notice that both rules are generated from same itemset ABC. Furthermore, rule $BC \Rightarrow A$ does not convey any extra information, if rule $A \Rightarrow BC$ is present, because it only swaps the antecedent and consequent itemset.

The authors considered rules as *strict redundancies* that are generated from two different frequent itemsets but one is the subset of another. For example, this approach consider rule $C \Rightarrow D$ as redundant with respect to $A \Rightarrow B$ if $A \cup B = X_i$ and $C \cup D = X_j$ and $C \supseteq A$, where $X_i \supset X_j$. But we argue that this property is not true in all situations. Consider two rules such as $R1$: $AC \Rightarrow BDE$; $R2$: $ACD \Rightarrow B$ (where $X_i = ABCDE$; $X_j = ABCD$ so $X_j \supset X_i$ and $ACD \supseteq AC$), and if we say rule $R2$ is redundant with respect to $R1$ and remove it, we then might lose a important rule because rule $R2$ does not fully characterize the knowledge of rule $R1$.

Jaki et al. [7] present a framework based on FCI (Frequent Close Itemset) that reduces number of redundant rules. The authors used FCI to form a set of rules and inferred all other association rules from that. Since FCI is used for choosing a set of rules based on the confidence value, the number of rules grows as the number of FCI increases. Therefore, it becomes difficult for end-users to infer other rules when there is a large number of FCI. Additionally, since this approach uses FCI, it may prune some important rules without considering the confidence and support value of those rules.

Liu et al. [5] present a technique to summarize the discovered association rules. It selects a subset of rules called direction-setting rules, in order to summarize the discovered rules. For example, we have rules such as *R*1*: A⇒C, R*2*: B⇒C and R*3*: A, B⇒C*. Rule *R*3 intuitively follows *R*1 and *R*2 and for this reason rule *R*3 is non-essential. The main drawback of this algorithm is that it focuses only on those association rules that have a single item in the consequence. Therefore, it is not able to remove redundant rules that have multiple items in the consequence. This algorithm also selects the target attributes (i.e. consequence) before the algorithm starts the rule mining task. Consequently it fails to find rules that have different consequence itemset. Since association rule mining is an unsupervised learning approach, if we choose the target attributes earlier, we may unable to generate some useful rules that do not belong to the target groups. Liu et al [6] also propose another algorithm known as multilevel organization and summarization of discovered rules. In this algorithm, rules are summarized in a hierarchical order, so that end users can browse all rules at different levels of details. However, this algorithm only summarizes the rules and redundant rules may still exist in the final model.

Similar to all of the abovementioned redundant rule reduction algorithms, our main objective of this paper is to eliminate redundant rules. However, our proposed methods have *two distinct features* that distinguish it from all other existing algorithms.

- The proposed methods are not based on any bias assumptions. In addition it verify each rule with set of rules in order to find redundant rule. Hence it eliminates redundant rules without losing any important knowledge from the resultant rule set.
- The proposed methods are case independent. It verifies all rules that have one or more items in the consequence. Therefore, it has the ability to eliminate redundant rules that contain single or multiple items in the consequence.

4 Proposed Methods

One can classify association rules in two different types based on the number of items in the consequence: rules having single items in the consequence and rules having multiple items in the consequence. Depending on the application requirements, association rule mining algorithms produce ruleset, which may contains rules of both types. However, it is worth to mention that redundant rules exist in both types. To eliminate redundant rules of these two types, we propose two methods: removing redundant rules with fixed *antecedent* rules, and with fixed *consequent* rules. The proposed methods not only remove redundant rules that are generated from frequent itemset but also have the ability to remove redundant rules when rules are being generated from the frequent closed itemset.

4.1 Finding Redundant Rules with Fixed Antecedent Rules

To remove redundant rules with fixed antecedent, we propose following theorem:

Theorem 1: Consider rule $A \Rightarrow B$ satisfying the minimum confidence threshold such that antecedent A has i items and consequent B has j items where $i \geq 1$ and $j > 1$. The

rule $A \Rightarrow B$ is said to be redundant if and only if n number of rules such as $A \Rightarrow e_1$, $A \Rightarrow e_2, \ldots A \Rightarrow e_n$ satisfy minimum confidence threshold where $\forall e \subset B$ and $n=j$.

Proof: Since $\forall e \subset B$

Then, $Support(A \cup e) \geq Support(A \cup B)$

$$\therefore \frac{Suppport(A \cup e)}{Suppport(A)} \geq \frac{Suppport(A \cup B)}{Suppport(A)}$$

$\therefore Confidence(A \Rightarrow e) \geq Confidence(A \Rightarrow B)$

Hence, if the rule $A \Rightarrow B$ is true in a certain level of support and confidence, then same must be true for all rules $A \Rightarrow e$ where $\forall e \subset B$. □

Example: Let us apply this theorem to a ruleset R that has three rules such as $\{AB \Rightarrow X, AB \Rightarrow Y$ and $AB \Rightarrow XY\}$. Consider the rule $AB \Rightarrow XY$ has s% support and c% confidence. Then, the rules such as $AB \Rightarrow X$ and $AB \Rightarrow Y$ will also have at least s% support and c% confidence because $X \subset XY$ and $Y \subset XY$. Since $AB \Rightarrow X$ and $AB \Rightarrow Y$ dominate $AB \Rightarrow XY$ both in support and confidence, for this reason $AB \Rightarrow XY$ is redundant.

4.2 Finding Redundant Rules with Fixed Consequence Rules

The traditional association rule mining algorithms produce many rules that have the same consequence but have different antecedents. To remove this kind of redundant rules, we propose following theorem:

Theorem 2: Consider rule $A \Rightarrow B$ that satisfies minimum confidence threshold such that antecedent A has i items and consequent B has j items where $i > 1$ and $j \geq 1$. The rule $A \Rightarrow B$ is said to be redundant if and only if n number of rules such as $e_1 \Rightarrow B$, $e_2 \Rightarrow B, \ldots e_n \Rightarrow B$ satisfy minimum confidence threshold where $\forall e \subset A$, $n=i$ and each e has $(i-1)$ items.

Proof: Since, $\forall e \subset A$ and $e_1 \cup e_2 \cup \ldots \cup e_n = A$

So, $support\ (e) \geq support\ (A)$

$\therefore Support(e \cup B) \geq Support(A \cup B)$

Thus if rules such as $e_1 \Rightarrow B$, $e_2 \Rightarrow B, \ldots e_n \Rightarrow B$ have a certain confidence threshold, then rule $A \Rightarrow B$ is redundant because $A = e_1 \cup e_2 \cup \ldots \cup e_n$. □

Example: Let us apply this theorem to a rule set R that has three rules such as $\{XY \Rightarrow Z, X \Rightarrow Z$ and $Y \Rightarrow Z\}$. Suppose rule $XY \Rightarrow Z$ has s% support and c% confidence. If n (i.e. number of items in the antecedent) number of rules such as $X \Rightarrow Z$ and $Y \Rightarrow Z$ also satisfy s and c then, the rule $XY \Rightarrow Z$ is redundant because it does not convey any extra information if rule $X \Rightarrow Z$ and $Y \Rightarrow Z$ are present.

4.3 Proposed Algorithms

Based on the abovementioned theorems, we propose two algorithms to discover redundant rules. The pseudocode of these algorithms is shown in figures 2 (a) and (b). The first algorithm finds those redundant rules that have multiple items in the consequence but have the same antecedent itemset in the antecedent. It first iterates

through the whole rule set and finds those rule that have multiple itemsets in the consequence. Once it comes across such a rule, checking is carried out to see whether *n* number of (*n-1*)-*itemset* of the consequence are in the rule set with the same antecedent. If it finds *n* number of rules in the rule set then we delete that rule from the rule set otherwise that rule remains in the rule set.

The second algorithm finds those redundant rules that have multiple items in the antecedent but have the same antecedent itemset in the consequence. It is similar to first algorithm except that it finds antecedent that have multiple itemset. Once it comes across such rule a check is made to see whether *n* number of (*n-1*)-*itemset* of the antecedent are in the rule set with the same consequence. If it finds *n* number of rules in the rule set then we delete that rule from the rule set otherwise that rule remains in the rule set.

```
for all rules r ∈ R
   r = ∪{A, C}
   n = length(A)
   if (n > 1)
      for all (n − 1) − subsets e ∈ A
         if (r_i = ∪{e, C})
            e.i++;
      end for
      if (i == n)
         R = R − r
   end if
end for
```

(a)

```
for all rules r ∈ R
   r = ∪{A, C}
   n = length(C)
   if (n > 1)
      for all (n − 1) − subsets e ∈ C
         if (r_i = ∪{A, e})
            e.i++;
      end for
      if (i == n)
         R = R − r
   end if
end for
```

(b)

Fig. 2. Pseudo Code for finding redundant rules that have (a) same antecedent but different consequence (b) same consequence but different antecedent

5 Performance Study

We have done performance study on our proposed methods to conform our analysis of its effectiveness in eliminating redundant rules. We have chosen four datasets for this performance study. Table 1 shows the characteristics of those datasets. It shows the total number of items, average size of transaction and the total number of transactions of each dataset. It is worth to mention that many association rule mining algorithms had used all of these datasets as a benchmark.

Table 1. Dataset Characteristics

Name	Avg. Transaction Length	No. of Items	No. of Transactions
pumsb*	74	7117	49046
Connect-4	43	130	67557
T40I10D100K	20	1000	100000
BMS-WEB-View- 1	2	497	59602

The pumsb* and Connect-4 [9] datasets are very dense (there are large number of items very frequently occurred in the transaction) and are able to produce very large itemsets when the support threshold is high. Therefore, we use very high support threshold for generating rules from those datasets. The T40I10D100K and BMS-Web-

View-1 [10] datasets are relatively sparse (total number of items are large but only few of them are occurred frequently in the transaction) and due to this we generate rules from those datasets using low support threshold value.

In following experiments we examine the level of redundancy (i.e. redundant rules) present in the resultant rule set. The benchmark for measuring the level of redundancy is referred to the redundancy ratio [4] and is defined as follows:

$$\text{Redundancy Ratio } (\partial) = \frac{\text{Total Rules Generated (T)}}{\text{Essential Rules (E)}} \quad \ldots\ldots\ldots (1)$$

$$\text{Essential Rules (E)} = \text{T - R} \quad \ldots\ldots\ldots (2)$$

where R is the total number of redundant rules present in the resultant rule set.

To find redundancy ratio in the traditional approach, at first we used those datasets to generate frequent itemsets using the Apriori [11] association rule-mining algorithm. Since we use different support threshold values, the total number of frequent itemsets varies for different support values. After generating frequent itemsets we use those itemsets for the rule generation purpose. To generate association rules in traditional approaches, we choose a publicly available a third party rule generation program developed by Bart Goethals [11]. We have implemented our proposed methods and compare them with the traditional approaches as shown in the Figure 3.

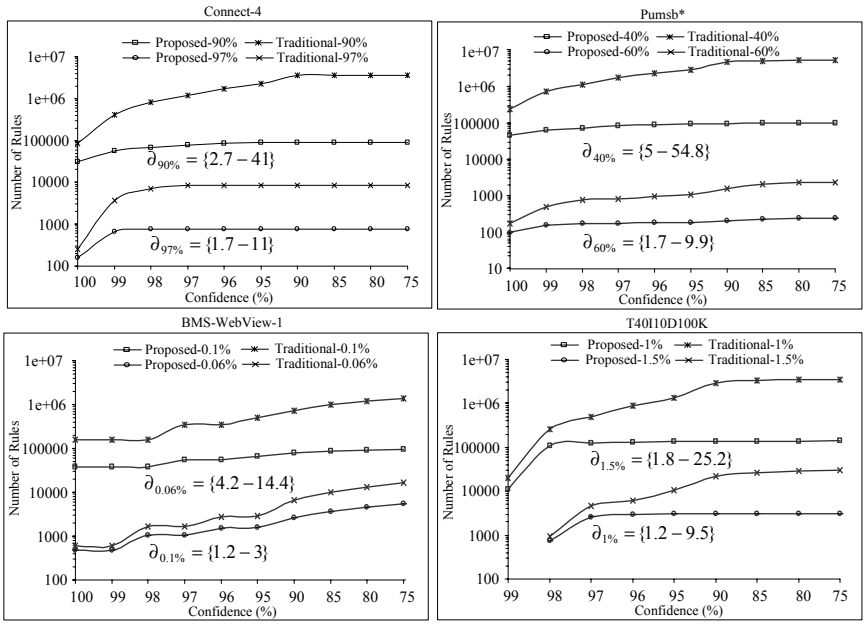

Fig. 3. Number of rule Proposed vs. Traditional

Figure 3 compares the total number of rules generated by the traditional methods with our proposed methods. It also depicts the redundancy ratio. From the above graph it is clear that the proposed methods reduce the total number of rules drastically. It generates 1.2 to 55 times less number of rules in compare with traditional approach. Since the traditional approach considers all possible subsets of a frequent itemsets as antecedent of a rule, it produces a large number of rules in all

datasets regardless of the support threshold. However, our proposed methods check every rule with a set of rules in order to find redundant rule. Therefore it only generates only few rules from each frequent itemset. In addition, the total number of rules grows as the support threshold decreases therefore the proposed methods reduce more number of redundant rules when support thresholds are low.

6 Conclusion

In this paper we examine various reasons that cause the redundancy problem in association rule mining. We also proposed several methods to eliminate redundant rules. The proposed methods rigorously verify every single rule and eliminate redundant rules. Consequently it generates a small number of rules from any given frequent itemsets in compare to all traditional approaches. The experimental evaluation also suggests that the proposed methods not only theoretically eliminate redundant rules but also reduce redundant rules from real datasets.

References

1. Rakesh Agrawal, Tomasz Imielinski and Ramakrishnan Srikant "Mining Association Rules between Sets of Items in Large Databases", *ACM SIGMOD*, pp 207-216, May 1993.
2. Mohammed Javeed Zaki, "Parallel and Distributed Association Mining: A Survey", *IEEE Concurrency*, pp. 14-25, October-December 1999.
3. Mohammed Javeed Zaki, "Scalable Algorithms for Association Mining" *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12 No.2 pp. 372-390 (2000).
4. Charu C. Aggarwal and Philip S. Yu, "A new Approach to Online Generation of Association Rules". *IEEE TKDE*, Vol. 13, No. 4 pages 527- 540.
5. Bing Liu, Minqing Hu and Wynne Hsu "Multi-Level Organization and Summarization of the Discovered Rules". *In the proc. KDD,* pp. 208-217, 2000.
6. Bing Liu, Wynne Hsu and Yiming Ma, "Pruning and Summarize the Discovered Associations". *In the proc. of ACM SIGMOD* pp.125 134, San Diego, CA, August 1999.
7. Mohammed Javed Zaki, "Generating non-redundant association rules" *In Proceeding of the ACM SIGKDD*, pp.34-43, 2000.
8. Bing Liu, Wynne Hsu and Yiming Ma, "Mining Association Rules with Multiple Minimum Supports". *In the proc. KDD,* pp. 337-341, 1999.
9. C.L. Blake and C.J. Merz. *UCI Repository of Machine Learning Databases*, University of California, Irvine, Dept. of Information and Computer Science, 1998, http://www.ics.uci.edu/~mlearn/MLRepository.html.
10. Ron Kohavi and Carla Brodley and Brian Frasca and Llew Mason and Zijian Zheng "KDD-Cup 2000 organizers report: Peeling the onion" , *SIGKDD Explorations*, Vol. 2 No.2 pp.86-98, 2000, http://www.ecn.purdue.edu/KDDCUP/ .
11. Bart Goethals, *Frequent Pattern Mining Implementations*, University of Helsinki-Department of Computer Science, http://www.cs.helsinki.fi/u/goethals/software/.