A Support-Less Confidence-Based Association Rule Mining Algorithm Using Relevance

Amiya Tripathy, G. Sunantha, James Baby, Rameez Bhimani, Joy Britto and Avinav Jami
Don Bosco Institute of Technology, Mumbai, India
{tripathy.a,sunantha1998,joybritto1,jami.avinav}@gmail.com
{james_it,bhimanrameez9786}@rediffmail.com

Abstract—Traditionally Association Rule Mining (ARM) algorithms work with the concept of frequent itemsets. These algorithms are useful in determining the high frequency patterns in data. This data can then be profitably used by businesses to benefit from commonly seen user trends. Hence the results displayed are those having high occurrence in the transaction database. In certain cases, these results may not be relevant to the user. Sometimes high confidence rules may need to be generated from itemsets which have low frequency i.e. low support and hence may be ignored by traditional ARM algorithms. In this paper, a ‘Support-less Confidence-based Association Rule’ (SCAR) algorithm has been proposed that can be used in conjunction with scenarios such as web search engines to improve the quality and relevance of the information retrieved by the user. This algorithm is an adaptation of the Support-Confidence based Association Rule Mining (ARM) algorithm. SCAR algorithm tries to look beyond the concept of frequent itemsets and display results most relevant to the user.

Index Terms— Rule Mining, Data Mining, Web Mining, ARM, Semantic Web

I. INTRODUCTION

The world has become metaphorically small as the Internet and Web technologies are continuously being updated and improved to provide an ensemble of information that is both relevant and related. The state of the web as it stands today can be described as “Organized Chaos”. Attempts have been made to organize this mess with the help of features like hyperlinks. However, the results of these attempts, though impressive, still offer further scope for improvement. This is because the machines filtering the content of the web for the user are ‘dumb’ i.e. they do not understand what they are operating on. ‘Understanding’ in a machine’s context is quite different from ‘understanding’ in context of humans. In the machine’s context, understanding the content means to be able to judge which page is most likely related to which one. There is a need to enable machines to gain this understanding. In this work an attempt has been made to generate these relations with the help of Association Rule Mining (ARM). Traditionally, ARM algorithms are used to mine frequent patterns in data to discover interesting relationships among a large set of data items [12]. In the context of this work, the ARM algorithm has been modified to look for relevant patterns in data instead of frequent patterns. Progress in the field of ARM algorithms have focused on improvements in identifying frequent itemsets. These improvements have resulted in the creation of various efficient algorithms, like frequent pattern (FP) tree algorithms and Apriori algorithms [6][8][9][12]. These improvements are most effective when searching for rules with high support and confidence [11]. In certain cases, the support might not be relevant. An example of such a situation might be that of an e-shopping site using association rules to suggest additional items to a user based on the items already present in the user’s shopping cart. The items present in a particular customer’s cart might not be high-frequency items. But irrespective of that, rules have to be generated to provide suggestions. Use of traditional ARM algorithms in this scenario will necessitate the setting of the minimum support to a very low value to remove the possibility of eliminating some relevant item with low support [11]. Thus all optimizations or benefits obtained due to improvements in determining frequent patterns are negated. Also, since these suggestions are to be generated on-the-fly and on the basis of user input, it is not feasible to generate all possible rules every single time the algorithm is run. Instead, only those rules should be generated which will be relevant in that particular scenario. In this work, SCAR algorithm, tentatively named as Support-less Confidence-based Association Rule (SCAR) Algorithm, uses the concept of relevance along with Association Rule Mining (ARM) in order to improve the quality of results obtained. It has altered the traditional ARM algorithm to provide an efficient performance and better results in the field of information search and retrieval. Though the primary focus is on the World Wide Web, SCAR algorithm is equally valid in scenarios such as providing suggestions to customers shopping in an e-store or determining the item a customer will most probably order in a hotel.

II. LITERATURE SURVEY

The current web holds immense volumes of data. This data could be used effectively to develop new web services. Many techniques are employed to improve and enhance the quality of these services. One way is to use better mining approaches. Cardoso (2006) used data mining techniques to
manage workflow Quality of Service (QoS) which were created by web services and applied mining to predict QoS of workflow [4]. These efforts talk about to improve the web services. Agrawal et al. (1996) revolutionized the field of ARM by introducing the Apriori algorithm which optimized the large itemset generation process greatly improving the performance [9]. Bayati et al. (2009) suggested a log structure, and based on this along with certain parameters, they used Association rule mining techniques to create a knowledge repository. This repository would help to predict the best composition based on user’s behaviors and processes [1]. Margahny et al. (2005) has suggested the use of the specialized “TreeMap” data structure and the “ArrayList” technique to reduce the cost of frequent pattern mining [10]. Shaofei (2009) talked about how traditional mining algorithms could be suited to be used on the web. His new frequent path algorithm uses maxim forward segmentation and aims to help the e-commerce unit design better websites based on frequent paths that satisfy minimum support and minimum confidence in the transaction database [5]. Zhang et al. (2009) suggested partitioning the database in smaller sets and finding frequent itemsets in each part then merge the several partitions to generate other frequent free item sets and count the support. Item sets’ matching is the main problem in extracting frequent item sets and the work aims to overcome this bottleneck [6].

These algorithms however focus on the frequent itemsets occurring in the databases. In certain situations, frequent itemsets may not be relevant. For example, when suggestions are to be made to users on the basis of rules generated relevant to data/search terms already entered by the user, generating all the frequent itemsets and associated rules is unnecessary and irrelevant. Ding et al. has suggested the use of a vertical data structure, the Peano Count Tree (P-Treec) and the Tuple Count Cube (T-Cube) to derive high confidence rules regardless of their support level [11].

In the current work an attempt has been made to generate relevant rules irrespective of the support but without using any specialized data structures or data warehousing-related operations. In this work a search term/seed itemset is used to generate only those itemsets and rules relevant to the user.

III. METHODOLOGY

The SCAR algorithm was developed in particular to address the problem of “support-less” itemsets. Traditional ARM algorithms are generally geared to handle high support, high confidence patterns. When the support count becomes low, most optimizations of traditional algorithms fail. The attempt made by this paper is to address this shortcoming by shifting the focus from frequent patterns to relevant patterns.

In order to understand the algorithm the following concepts need to be cleared. A brief explanation of each term is provided below.

Key Terms

- **Singleton Itemset**: An itemset having a single item. For example: {Milk}.
- **Complex Itemset**: An itemset having more than 1 item. For example: {Milk, Bread, Cheese}.
- **Seed Itemset**: The value entered by the user as a query to the algorithm is referred to as seed itemset. It is the itemset for which the user wishes to generate the rules. Suppose the user wants to find all rules pertaining to bread. The seed itemset will be {Bread}. The seed itemset could also be a complex itemset. For example: {Bread, Cheese}.
- **Itemset Seeding Rule**: Itemset seeding rule. This rule is similar to the ‘Large Itemset’ Rule of Apriori Algorithm.
- **SCAR-feed**: The practical implementation of Itemset Seeding Rule is referred to as the SCAR-feed algorithm.
- **Input Database**: The database of itemsets which is acts as the basic input to any ARM algorithm is the Input Database.
- **Base Table**: Base table is the table of singleton itemsets relevant to the seed itemset entered by the user. It is the collection of key words mined from the database table depending on the seed item.
- **Base Table Generator**: Base table generator or BTG is a function used to generate the base table.
- **Support Table**: The support of the various relations from the relation table is calculated and stored in the support table.
- **Support Table Generator**: Support Table Generator or STG is a function used to generate the Support Table.
- **Association Rules**: Association rules are a way of denoting relations between various terms. Written as \( A \rightarrow B: \langle \text{confidence} \rangle \); Here, A and B denote itemsets, and \( A \rightarrow B \) defines the probability of the occurrence of B in the case that A has occurred. Higher the confidence of the rule, greater the probability.

![Figure 1. SCAR Algorithm Block Diagram](Image)

© 2010 ACEEE
DOI: 01.IJRTET 04.02.229
As can be seen in Figure 1, the basic input to the SCAR algorithm is the database. The database table is processed with the seed itemset. After the seed itemset is searched in the database to generate additional relevant itemsets, all the relevant itemsets are sorted and stored in the base table. The itemsets from the base table are collected and all possible combination of these itemsets are made. These combinations are nothing but all possible relations that can be formed with those itemsets. For \( n \) itemsets from the base table, \( 2^n - 1 \) relations are possible. The support of all the relations is generated from the database. All the values below a threshold values are discarded and represented in a proper manner with the searched query and all related itemset along with their support. Higher the support, higher is the probability of the itemset being associated with the searched query.

Traditional ARM algorithms work on the concept of ‘Frequent Itemset’. According to this concept, itemsets having a high degree of occurrence i.e. high frequency are given preference while other itemsets are eliminated. While this is an efficient approach to be followed when all rules having high support and confidence are to be generated processing of the results takes a very long time i.e. all the algorithm related optimizations fail if the support value is very low.

In certain scenarios, for example, when ARM is to be used to provide suggestions for online shopping sites, or suggest relevant results for a search engine, the support of the generated rules do not matter. The expected rules in the result may be generated from the itemsets which are not frequent but their relevance to the user may be very high. For example, an engineering student might be searching results related to the programming language, Python. However, the term ‘Python’ with reference to the animal (snake) ‘Python’ may be a frequent itemset, as compared to ‘Python’, the programming language. Hence the traditional ARM algorithm with a moderately high support constraint will eliminate all itemsets relating to the latter and display results only relating to the former. To include the relevant result for the user (i.e. ‘Python’ a programming language) the support constraint will have to be set to a very low value which will lead to a drastic decrease in the performance of the algorithm and wastage of all algorithm related optimizations.

In contrast, the SCAR algorithm will generate only those itemsets related to python the programming language irrespective of its support value. Now only those rules need to be generated which have the search term on the left hand side because those rules will denote the relation of the search term with other terms not entered by the user. Hence, only one rule will be generated per itemset as opposed to \( 2^n - 2 \) rules per \( n \)-item itemset in the traditional algorithm. All other rules are irrelevant. The SCAR algorithm will generate only the relevant rules. Hence, the performance of the algorithm becomes independent of the support count.

In the context of the semantic web, the benefits of shifting the focus away from support and towards the relevance of the itemset becomes very obvious. The intention of shifting the focus away from the support is two pronged. The traditional approach does not consider the relevance of the generated rules for the user. This means given a database, the algorithm will generate a set of rules depending upon the specified support and confidence constraints and expects the needs of the user to be satisfied by the existing rule set. If the user happens to look for an item which is not frequent the traditional algorithm does not cater to his/her needs. Only results pertaining to the frequent items are displayed. Also, since the user is not being considered while generating the rules a very large database of rules has to be generated requiring a large amount of processing which is not feasible for web oriented scenarios. On the other hand, the SCAR algorithm works in the following manner.

Initially, a lookup table has to be generated. This is probably the stage which will consume maximum time and processing capacity. An algorithm to perform this in an efficient manner is discussed in the subsequent paragraph. From the generated lookup table, association rules will be generated with the condition that only those rules will be generated which have all of the searched terms on the Left Hand Side (LHS). Now the rules are sorted on basis of decreasing confidence and the data can be sorted on the basis of these rules with the rules having higher confidence being given more priority.

The SCAR-feed algorithm was developed to optimize the process of lookup table generation. It utilizes an optimized process for the generation of relevant itemsets based on the itemset ‘seeding’ rule. The smallest possible valid complex itemset is the itemset comprising only of the searched terms. For example, if ‘Scorpio SUV’ were searched, then Scorpio SUV is the smallest valid itemset. The logic is as follows: Generate all possible combinations of all the relevant itemsets excluding those in the seed itemset. Now insert the singleton itemsets of the seed itemset in the generated combinations in proper alphabetical order. The resultant set of itemsets is the set of valid relevant itemsets. These itemsets are then used by the SCAR algorithm to generate relevant rules.

A pattern is identified where if \( n \) singleton itemsets are present then the total relations generated are \( 2^n - 1 \). The number of relations to be computed could be reduced by reducing the number of singleton itemsets.
TABLE 1
RULES GENERATED BASED ON ITEMSET LENGTH.

<table>
<thead>
<tr>
<th>Itemset Length</th>
<th>No. of Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
</tr>
</tbody>
</table>

Also at a later stage, the number of rules generated can be estimated by the relation of $2^m-2$ where $m$ is the number of items in an itemset. Table 1 indicates the number of rules that will be generated for an itemset of a particular length. If the numbers of relations are reduced then itemset number will be reduced and thus save on time to compute unnecessary rules which are of no relevance.

IV. ALGORITHM

The algorithms of SCAR and SCAR feed are specified below.

Algorithm of SCAR feed:
1. Retrieve information to be mined from the information repository.
2. Extract transactions relevant to the search term/seed itemset entered by the user.
3. From these transactions extract singleton itemsets relevant to the search term/seed itemset.
4. Generate large itemsets from these relevant singleton itemsets using the itemset-seeding rule.
5. Calculate support for all the generated large itemsets.

Algorithm of SCAR:
1. Generate relevant rules from these large itemsets.
2. Calculate the confidence for the generated rules.
3. Sort the rules on the basis of the confidence.

The mathematical model of the SCAR and the SCAR feed algorithms are as follows:

SCAR feed:
Input:
- $S$ //Seed itemset entered by user
- $B$ //List of relevant singleton itemsets

Output:
- $S$ //Seed itemset entered by user
- $Se$ //Support of the Seed itemset
- $C$ //Relevant itemsets
- $S_e$ //Support of all relevant itemsets

SCAR feed algorithm:
- $B_1 = \{x \mid (x \in B) \land (x \neq S)\};$
  //List of relevant itemsets after removing Seed itemset
- $C = $ All possible combinations of the itemsets in $B_1$
- $C_i = \phi;$
  for each $c \in C$ do
    $c_i = c \cup S;$
    //Concatenate the Seed itemset to each of the generated itemsets
    $c_i = $ Sort($c_i$);
    //Sort the items in the concatenated itemset in alphabetical order
    $C_i = C_i \cup c_i;$
    $Se_i = $ Support of the Seed itemset
    $S_e = $ $S_e \cup s;$
    //Calculate the support of all itemsets

As shown in the mathematical model above, the SCAR feed algorithm generates relevant complex itemsets along with support for the SCAR algorithm.

SCAR:
Input:
- $S$ //Seed itemset entered by user
- $Se_i$ //Support of the Seed itemset
- $C$ //Relevant itemsets
- $S_e$ //Support of all relevant itemsets

Output:
- $R$ //Relevant Rules
- $C_f$ //Confidence of relevant rules

SCAR algorithm:
- $R = \phi;$
  for each $c \in C$ do
    $R = R \cup \{S \rightarrow c\};$ //Generate all relevant rules
- $C_f = \phi;$
  for each $s \in S_e$ do
    $C_f = C_f \cup \{(s/Se_e)*100\};$
    //Calculate the confidence of the rules

As shown in the mathematical model above, the SCAR algorithm generates the relevant rules along with confidence.

V. RESULTS AND DISCUSSION

Testing of the algorithm was done on data from a hotel’s sales transactions. The hotel simulation had 39 transactions with 32 distinct items. The algorithms of SCAR and SCAR feed have been specified in previous section.
'Royal Stag' was selected as seed for the present test case. It should be noted that if a traditional algorithm would have been used then it would have worked with all the 32 singleton itemsets. The tables below indicate the number of records across all the component tables.

<table>
<thead>
<tr>
<th>TABLE 2</th>
<th>SEED: ROYAL STAG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SCAR</td>
</tr>
<tr>
<td>Base Table</td>
<td>7</td>
</tr>
<tr>
<td>Relation Table</td>
<td>127</td>
</tr>
<tr>
<td>Support Table</td>
<td>127</td>
</tr>
</tbody>
</table>

Now, in table 2, the SCAR-feed algorithm considers only 7 itemsets out of the total 32 as relevant to the searched term. Figure 2 shows the 7 relevant singleton itemsets for the test case. Hence instead of generating $2^{32} - 1 = 4294967295$ results, only $2^7 - 1 = 127$ results are generated, resulting in a drastic improvement in the performance. Figure 3 shows the last section of the 127 results that were computed using the SCAR-feed algorithm. The number of results may vary as the search term changes but it will definitely give a better performance that the traditional algorithm in every case. Figure 4 shows the rules generated by the algorithm. Only those rules were generated which are relevant to the seed itemset. In another example to demonstrate the efficiency of the SCAR algorithm, a simplified database of 10 transactions was created and both the SCAR and the traditional ARM algorithm were run on it. This simplified database contained 15 singleton itemsets. The intention was to generate rules relevant to the seed itemset (e.g. 'Dal').

The ARM algorithm was executed with 2 sets of values of support and confidence. Case-1 with 10 Support, 10 Confidence and case-2 with one Support, one Confidence. In both cases, the Operational Database of itemsets was of the same size. The exact values are given in the table below.

<table>
<thead>
<tr>
<th>TABLE 3</th>
<th>ARM ALGORITHM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Table</td>
<td>15</td>
</tr>
<tr>
<td>Relation Table</td>
<td>32767</td>
</tr>
<tr>
<td>Support Table</td>
<td>32767</td>
</tr>
</tbody>
</table>

In the case-1, all the relevant itemsets got eliminated due to them having low support values in spite of setting such a low constraint on support. In case-2, all the relevant rules were generated. At the same time, many irrelevant rules were also generated causing heavy wastage of processing time and power.

In case-1, four rules were generated with none of them being relevant. At the same time, in case-2, 684 rules were generated with 7 of them being relevant. On the other hand, the SCAR algorithm generated only the 7 relevant rules directly from the relevant itemsets in the database saving significant amount of time. Observation of these cases have thrown up the conclusion that the SCAR algorithm comes up with considerable improvement in performance vis-a-vis the traditional approach and the improvement in performance becomes more and more pronounced as databases grow bigger.

As can be seen, the procedure involves a significant amount of processing to be performed in addition to the search time in search engine scenario. This can prove to be a drawback since time-lags and processor strain are still issues despite drastic improvements in efficiency of the algorithm. In case of the search engine scenario, the algorithm can be executed during run-time and the result-
set will be stored so as to avoid future delays for the same keyword search. Also, as more and more searches are made, the algorithm will provide a better solution.

V. CONCLUSION

Association Rule Mining algorithms have been used in the field of Data Mining for a long time and these algorithms have been primarily developed for usage in data warehouses on huge amounts of data. These algorithms are less suitable in scenarios where information is dynamically processed and user preferences are an important consideration. The proposed SCAR algorithm attempts to bridge this gap. It has been observed that the SCAR algorithm comes up with considerable improvement in performance vis-a-vis the traditional approach and the improvement in performance becomes more and more pronounced as databases grow bigger.

ACKNOWLEDGMENT

This work has been done in the Project Laboratory of Don Bosco Institute of Technology, Mumbai, India and is fully supported by the Institute.

REFERENCES