# One Scan is Enough: Optimising Association Rules Mining

Mohamad Saraee
*School of Computing, Science and Eng.*
*University of Salford,*
*Manchester M5 4WT, UK*

Mohmoud Al-Mejrab
*School of Computing, Science and Eng.*
*University of Salford,*
*Manchester M5 4WT, UK*

## Abstract

Data mining is as a new area of research has taken its place as one of the most important techniques in the decision making process. Mining association rules is one of simple yet powerful technique in the data mining process The problem of mining association rules is composed of finding the large itemsets and to generate the association rules from these itemsets. Usually the dataset must be scanned many times in order to find the large itemsets. Many algorithms have been developed to increase the performance of mining association rules through reducing the number of scans over the dataset. This work aims to enhance and optimise the process even further by developing techniques to reduce the number of database scans to just only one.

**Keywords**: Data mining, Association Rules Mining, *SimpleARM*,

## 1. Introduction

The problem of mining association rules is composed of fining the large itemsets and to generate the association rules from these itemsets [1]. The process of finding large itemsets is complicated and computationally intensive task. Different algorithms have been developed but so far a practical and useful technique with acceptable performance has not been implemented. The algorithm presented here would allows generating frequent itemsets and accurate association rules with good performance. In this proposed framework we are presenting *SimpleARM* (*Simple A*ssociation *R*ules *M*iner) that counts items in the database of transactions and accumulates information in three arrays which can be used later to discover large (frequent) itemsets. Association rules can be generated from these large itemsets.

The first part of the problem requires many passes over the dataset to find all large itemsets. This process highly affects the overall performance of the association rules mining process. Multiple scans over the dataset are time consuming because in most cases huge amount of data are being mined. *SimpleARM* optimises the association rules mining process by reducing the number of passes over the database to just one and thus increasing the performance. Only one pass over the database generates both 2-Itemsets and 3-itemsets with a minimum user specified support. Information about the items are registered in three arrays: *General_Matrix ,Before_Matrix*, *After_Matrix*, These arrays are then used to generate the association rules in a very simple and efficient way.

## 2. *SimpleARM* Approach

The process of finding large itemsets is complicated and computationally intensive task[]. Different algorithms have been developed but so far a practical and useful technique with acceptable performance has not been implemented [2] [3] [4] [5]. The algorithm presented here allows generating large itemsets and accurate association rules with better performance. In this paper we present *SimpleARM* that counts the items and accumulates information in three arrays which can be used later to discover large (frequent) itemsets. Association rules can then be generated from these large itemsets. In addition *SimpleARM* uses previously discovered rules to incrementally update the matrices in order to discard obsolete rules and to speedup the mining operation.

The first step of the *SimpleARM* is to find all items and insert them in a one dimensional array *ItemList.* In only one scan over the database of transactions, information is accumulated. Transactions may have different length (number of items). For each single item its frequency, position are recorded. In addition the order of item in transaction is registered. Another array contains the number of occurrences of each item in dataset transactions and also occurrences of the same item with another item in transactions of length 2, 3, 4 and so on. At the same time the transaction number which contains the current item being processed is recorded in the array of that item e.g. item A has an array contains all the transaction number where attribute A is an entry in that transaction. These arrays will be used to find the actual support and confidence of the discovered rules. For the rest of this paper we use the database of transactions *T* shown in figure 1 as a working example.

| Transaction No. | Itemsets |
|---|---|
| 1 | A C D |
| 2 | B C E |
| 3 | A B C E |
| 4 | B E |

Figure 1: Database of Transactions *T*

*Item_List_Array*: As the transactions being scanned new item is added to this Item_List.

*GeneralMatrix*: A general matrix (2 dimesional array) with a size equal to (Item_List* Item_List) e.g. (5*5) for dataset *T* is created to record occurrences (frequency) of each item and the same item occurrences with every other item in the dataset. The resulted array shown in figure (2):

| | A | C | D | B | E |
|---|---|---|---|---|---|
| A | 2 | 2 | 1 | 1 | 1 |
| C | 2 | 3 | 1 | 2 | 2 |
| D | 1 | 1 | 1 | 0 | 0 |
| B | 1 | 2 | 0 | 3 | 3 |
| E | 1 | 2 | 0 | 3 | 3 |

Figure 2: GeneralMatrix for dataset *T*

For example, *GeneralMatrix*(1,1) = 2 represent the frequency of the attribute A, while *Generalmatrix* (2, 2) = 3 represent the frequency of C. The row corresponds the item A represent number of occurrences of A with each attribute in the dataset e.g. *Generalmatrix* (1, 2) = 2 represent the number of occurrences of A with C in the dataset transactions in any order A, C or C, A. *Before_Matrix:* contains similar information to the information contained in the *General_Matrix* as shown in figure 3. The only difference is that this array also includes the order of the items in the transactions e.g. number of times A was before B in the transactions.

| | A | C | D | B | E |
|---|---|---|---|---|---|
| A | 2 | 2 | 1 | 1 | 1 |
| C | 0 | 3 | 1 | 0 | 2 |
| D | 0 | 0 | 1 | 0 | 0 |
| B | 0 | 2 | 0 | 3 | 3 |
| E | 0 | 0 | 0 | 0 | 3 |

Figure 3: Before_Matrix of dataset *T*

*After_Matrix* contains a number which represents the number of the occurrences of an item as a postfix of another item in the dataset e.g. *After_Matrix* (2, 1) =2 represent the number of occurrences of C after A in the dataset *T* . There is an exception of those such as (1, 1), (2, 2),…,which represent the frequency of the dataset items. *After_Matrix* is shown in figure 4.

| | A | C | D | B | E |
|---|---|---|---|---|---|
| A | 2 | 0 | 0 | 0 | 0 |
| C | 2 | 3 | 0 | 2 | 0 |
| D | 1 | 1 | 1 | 0 | 0 |
| B | 1 | 0 | 0 | 3 | 0 |
| E | 1 | 2 | 0 | 3 | 3 |

Figure 4: After_Matrix of dataset *T*

*Items_Occurrences:* This array contains the number of occurrences of each item in a specific transaction length. The array column represent the transaction length while the row represents the number of item occurrences in each transaction length, e.g. attribute A occurred only two times in the dataset *D* one in a transaction of length 3 (3 items) and another in a transaction of length 4.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| A | 0 | 0 | 1 | 1 | 0 |
| C | 0 | 0 | 2 | 1 | 0 |
| D | 0 | 0 | 1 | 0 | 0 |
| B | 0 | 1 | 1 | 1 | 0 |
| E | 0 | 1 | 1 | 1 | 0 |

Figure 5 : Items in Transactions *T*

Array shown in figure 6 includes the number of transactions with length (number of items). There is no transactions of length 1, and there is only one with length 2. Also there are two transactions of length 3 and one with length 4.

This array gives an idea about the distribution of dataset transactions according to their length and also which transactions with a specific length are more frequent.

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |

Transaction length

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 0 |

Transaction number

Figure 6: Transaction length array

Each item in the transactions has an array to hold all transactions numbers and to rank the item in that transaction. The number of entries in each array depends on the number of occurrences of the item in the dataset transactions. The transaction arrays of items in dataset *T* are shown in figure 7.

| A | | C | | D | | B | | E | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 1 | 3 | 2 | 1 | 2 | 3 |
| 3 | 1 | 2 | 2 | | | 3 | 2 | 3 | 4 |
| | | 3 | 3 | | | 4 | 1 | 4 | 2 |

Figure 6: Transaction length array

By using this information we can find the actual support and confidence of the itemsets larger than 2-itemsets.

## 2.2 Generating itemsets

Instead of making multiple passes over the database to count the support of individual items and then to discover the large itemsets, *SimpleARM* uses the information collected in only one pass over the database to perform this task. Matrix in figure 5 and the one-dimensional array in figure 6 both have an idea about the density of the database (which transactions with a specific length are more frequent than others) in the example dataset *T* figure 1 it can be noticed from figure 6 that transactions with length of 3 items are more frequent than those with one or four items.

### 2.2.1 Generating 2-Itemsets

All dataset items are found in the transactions of three items length but from the general matrix we know that item D is not frequent according to a minimum support = equal to 40% and it will not be in any frequent item set. The support 40% is satisfied by the other items (A, B, C, E), these frequent items will construct the frequent itemset(s) but by checking up the Before matrix for 2-itemsets can be constructed from these 1-itemsets ({A}, {B}, {C}, {E}) which are ({A, B}, {A, C}, {A, E}, {B, C}, {B, E}, {C, E}). It is clear that the only 2-itemsets satisfy the minimum support (40%) are ({A, C}, {B, C}, {B, E}, {C, E}). The resulted support equals (50%, 75%, 75%, 50%) respectively which is greater than the minimum support (40%).

All two items rules can be discovered from the *Before_Matrix*, with a minimum confidence equal to 40%. All 2-itemsets can be tested to generate rules (A➔C, B➔C, B➔E and C➔E). Confidence of these rules is (sup.(AC) / sup(A)), (sup(BC) / (sup(B)), (sup(BE) / sup(B)) and sup(CE) / sup(C)), which are equal to (100%, 66%, 100%, and 66%) respectively. It is clear that all of them satisfy the minimum support and confidence. From the 2-itemsets satisfying the minimum support, the expected 3-itemsets (candidate 3-itemsets C3 in *Apriori* algorithm) can be constructed. The rule is that for any 3-itemset to be a frequent, all its subsets must be frequent. The 3-itemset ABC is not frequent because its subset AB does not satisfy the minimum support. Consequently the only expected item set is BCE which all their subsets satisfy the minimum support (40%), (BC 50%, BE 75% and CE 50%).

### 2.2.2 Generating 3-Itemsets

S*impleARM* starts with generating the expected candidate 3-Itemsets including those with support equal to 0 (zero) to generate rules such as the rule AB=>C from the 2-Itemsets satisfying the minimum support. The process is to get the first attribute (prefix) of the first 2-Itemset. Then check all other 2_itemsets to find out if there is any start with the same attribute. If it does find any then both 2-Itemsets will contribute to build a new 3-Itemset with unknown support which may be less than the minimum or equal to 0. From the 2-Itemsets {A, C}, {B, C}, {B, E}, {C, E}) attribute A will not be in any 3-Itemset as first attribute, C as well, will not be a start attribute of any 3-Itemset because there is no any other 2-Itemset start with C. Only 2-Itemsets {B, C} and {B, E} which both start with B will together build the one 3-Itemset {B, C, E}. It is important to find the actual support of this 3-Itemset and then to find its confidence if it is satisfying the minimum support.

Two rules can be found from the 3-Itemset {B, C, E}. The first is BC=>E and the second is B=>CE. *SimpleARM* can find the confidence of both rules, where all the frequencies of singles and pairs of items are stored in *Before_Matrix*. Then from the intersection of item transactions arrays B, C and E it can find the actual support of the 3-Itemset. The intersection of the three arrays gave two transactions (transaction# 2 and transaction# 3) that means the support of 3-Itemset {BCE} is 2/4 = 50% > 4%. Then the confidence of both rules = (sup. BCE / sup. BC) = 50 / 50 = 100% and (sup. BCE / sup. B) = 50 / 75 = 66%. It can be seen that both rules are holding and satisfying both minimum support and confidence. At the same time when *SimpleARM* discover the candidate 3-Itemsets it calculate the possible confidence for both possible rules can be found from one 3-Itemset such as the confidence of the rule (AB=>C) and (A=>BC) from the 3-Itemset {A, B, C}.

### 2.2.3 Generating itemsets of 4 and more items

*SimpleARM* uses the (n-1)-itemset to discover n-itemset in the same way of discovering 3-Itemsets from the 2-Itemsets in the previous section. To discover 4-Itemsets it uses 3-Itemsets satisfying the minimum support. *SimpleARM* will not find any 4-Itemsets from the 3-Itemset {B, C, E} as it is the only 3-Itemset satisfying the minimum support and it needs at least another 3-itemset such as {B, C, D} and it must satisfy the minimum support. *SimpleARM* in this case will catch the first 3-itemset {BCE} and will search for any other 3-Itemset with first two attributes (prefix) matches the first two attributes B and C in the first caught 3-Itemset {B, C, E}. *SimpleARM* generate the expected 4-itemset {B, C, E, D} (corresponding C4 in *Apriori* algorithm). The next step is to find the actual support of each item set by looking up item transaction arrays to find their intersections and then to discover rules and their actual confidence in the same way when rules discovered in the previous section.

*SimpleARM* is able to find the following rules from the 4-Itemset {B, C, E, D}, (B=>CED, BC=>ED, BCE=>D) with their actual support and confidence using the *Before_Matrix* and the previously discovered 3-Itemsets. In the same way *SimpleARM* calculates the rules confidence in the case of the 3-Itemsets. It also calculate the confidence of the rules can be discovered from 4-Itemsets such as the rules produced from the 4-Itemset {A, B, C, D} which are (A=>BCD), (AB=>CD) and (ABC=>D).

## 3. Conclusion

In this short paper we have presented SimpleARM data mining tool. It can be concluded that *SimpleARM* has succeed in collecting all information requires to generate association rules in scanning the database Discovery of the rules with more than one attributes as consequent is also possible. *SimpleARM* is simpler than many other algorithms in its discovery of large itemsets and generation of association rules on one hand but on the other hand it collects enough information from the dataset in only one scan. SimpleARM is in the early stage of its development.

## 4. References

[1] R. Agrawal and R. Srikant, ™Fast Algorithms for Mining Association Rules,º Proc. 1994 Int'l Conf. Very Large Data Bases, pp. 487±499, Santiago,Chile, Sept. 1994.

[2] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, ™Data Mining Using Two-Dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization,º Proc. 1996 ACM SIGMOD Int'l Conf. Management of Data, pp. 13±23, Montreal, June 1996.

[3] H. Mannila, H. Toivonen, and A.I. Verkamo, ™Efficient Algorithms for Discovering Association Rules,º Proc. AAAI '94 Workshop Knowledge Discovery in Databases (KDD '94), pp. 181±192, Seattle, July 1994.

[4] J.S. Park M.S. Chen, and P.S. Yu, ™An Effective Hash-Based Algorithm for Mining Association Rules,º Proc. 1995 ACM SIGMOD Int'l Conf. Management of Data, pp. 175±186, San Jose, Calif., May 1995.

[ 5] A. Savasere, E. Omiecinski, and S. Navathe, ™An Efficient Algorithm for Mining Association Rules in Large Databases,º Proc. 1995 Int'l Conf. Very Large Data Bases, pp. 432±443, Zurich, Sept. 1995.