Finding Association Rules through Efficient Knowledge Management Technique

Anwar M. A.
College of Engineering and Computing
Al Ghurair University
Dubai Academic City, UAE

***Abstract*— One of the recent research topics in databases is Data Mining, to find, extract and mine the useful information from databases. In case of updating transactions in the database the already discovered knowledge may become invalid. So we need efficient knowledge management techniques for finding the updated knowledge from the database. There have been lot of research in data mining, but Knowledge Management in databases is not studied much. One of the data mining techniques is to find association rules from databases. But most of association rule algorithms find association rules from transactional databases. Our research is a further step of the Tree Based Association Rule Mining (TBAR) algorithm, used in relational databases for finding the association rules .In our approach of updating the already discovered knowledge; the proposed algorithm Association Rule Update (ARU), updates the already discovered association rules found through the TBAR algorithm. Our algorithm will be able to find incremental association rules from relational databases and efficiently manage the previously found knowledge.**

Keywords- Data Mining; Co-occurrences; Incremental association rules; Dynamic Databases.

I. INTRODUCTION

At the very abstract level of data mining, it is part of Artificial Intelligence. One of the data mining techniques for finding useful information from the database is association rule. Association rules find the co-occurrences among item sets in the database. For example in a customer transaction database we want to find that whenever customer purchases item A, item B is purchased how many times. These co-occurrences are found through finding the large item sets. As mentioned in [1] to find the large item sets, it should be greater than the minimum support threshold, which is the minimum number of transactions from the database having that item set.

There are two issues related to association rules.

- Finding the preprocessing algorithm for association rules
- Update algorithm for association rules. The update algorithm enables to efficiently update the already discovered information .So the update algorithm depends very much on the preprocessing algorithm used.

Most of the association rules algorithms like Apriori [2], DHP [5], OCD [9] and [12] find association rules from transactional databases. In case of association rules from relational databases TBAR [10] algorithm was developed as a loosely couple approach.

The most recent algorithms for the update algorithms like FUP [3], MLUP [4], FUP2 [8], UWEP [7], and SWF [11] etc find updated association rules from the transactional databases. In our research we have developed a new update algorithm for finding the updated information from the relational database on the basis of the TBAR algorithm. Our performance study shows that the proposed solution is 2.1 to 2.3 times faster as compared to TBAR algorithm. We present an efficient algorithm, ARU, for finding association rules and apply a new knowledge management technique, to reuse the previously discovered knowledge from the relational databases. Precisely rather than finding large item sets from scratch, the large item sets found through the TBAR algorithm are stored and reused.

In association rules we find the co-occurrences among item sets through finding the large item sets. An item set is large if it is above the minimum support threshold .For example in a database if the minimum support threshold is 5%, then all the item sets from the database having more than 5% occurrence will be included in large item sets. So the main problem in maintenance of association rules is updating the large item sets. In our prototype system we have been able to update the large item sets more efficiently as compared to the previous approach of TBAR.

II. PRELIMANARIES

Let $I = \{i_1, i_2, \ldots\ldots, i_m\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Each transaction is associated by an identifier, called TID. Let X be a set of items. A transaction T is said to contain X if and only if $X \subseteq T$. An association rule is an implication of the form $x \Rightarrow y$, where $x \subseteq I$, $y \subseteq I$ and $X \cap Y = \varnothing$. The rule $x \Rightarrow y$ holds in the transaction set D with confidence c if c% of transactions in D that contain x also contain y.

The rule $x \Rightarrow y$ has support s in the transaction set D if s% of the transactions in D contains $X \cup Y$. For a given pair of confidence and support threshold, the problem of mining association rules is to find out all the association rules that have confidence and support greater than the corresponding thresholds. As there is lot of research for finding the association rules, given large item sets, our focus will be to find the large item sets from the updated database. The notion of item must be redefined in a relational database. An item will be a pair a: v

where a is the attribute and v is the value of a. a fundamental property of an item in a relational database is that they cannot contain more than one item per table column if a1:v1 and a2:v2 belong to an item set, then a1≠a2 which is the consequence of the First Normal Form (1NF) in databases: a relation is in 1NF if it's attribute domain contain atomic values only. This justifies our distinction between items in transactional and items in relational databases.

III. System Overview

Our algorithm is based on the TBAR algorithm, which finds the association rules from the relational database. Our incremental association rule algorithm is an improvement of that algorithm to find incremental association rules from the relational databases. We apply a new Knowledge Management technique, to find the incremental association rules from dynamic databases more efficiently as compared to finding the association rules from the database.

As shown in Figure 1, our algorithm is implemented as the data integration module to efficiently update the association rules. The large 1-item sets found through the TBAR algorithm is saved in the knowledge base .In our algorithm of update we have reused those large 1-item sets from the knowledge base and thus saved the CPU time and one scan of the database. As depicted in [6] we can couple association rule algorithm with the relational database in a number of ways. In our case we opted for the loosely coupled approach, as our data mining application process space is outside the database process space.

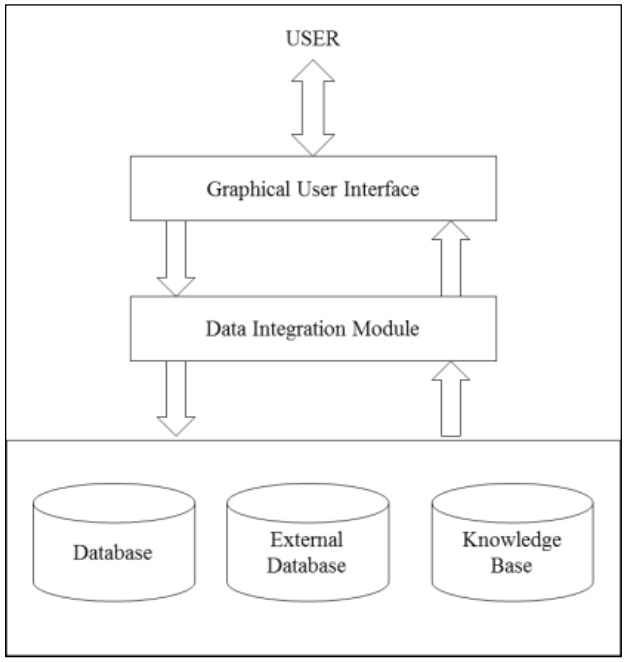

Figure 1. The System.

IV. TBAR Algorithm

The TBAR algorithm uses the item set tree data structure to efficiently store all L_{ks} .All L_{ks} are organized on the basis of levels.

TBAR Algorithm

```
Set.Init(minsup);
Itemsets=set.Relevants(1);
StoreL1(itemsets);                    (Step 4.3)
K=2;
While(k<=cols && itemsets >=k)
{
    itemsets =set.candidates(K) ;
    If(itemsets >0)
    Itemsets=set.Relevants(k);
    K++;
}
```

In this case *init* method creates and initializes the item set tree. The *set.Rrelevants(1)* method finds large *1-item* sets from the database. For finding subsequent large item sets it is checked that the item sets found should be greater than the number of columns. We first find candidate item sets from the previous large item sets and then find the subsequent large item sets from the database until all the large item sets are found from the database. In step 4.3 the TBAR algorithm has been modified to store all *L1*s in the knowledge base for subsequent reuse of that information.

V. ARU Algorithm

The ARU algorithm differs from all other update algorithms for association rules as it updates the large item sets in relational databases. So the large 1-item sets are related to a column in a table rather than a transaction in transactional databases. In our case we will find the support for each item set corresponding to a column value in the database.

Inputs

DB=initial database before any updates

db=update portion of the database

DB + db=whole updated portion of the database

L1 DB = large 1-item sets item sets found in DB

attr = attribute in L1 DB

attr.number=attribute number

attr.value=attribute value

attr.count=support of the attribute value

Output

$L1_{DB+db}$=large 1-item sets in updated database DB+db

ARU ALGORITHM

If there is any insertion in the database (Step 5.1)
For L1 DB of attribute attr in database
Get the column number attr.number of the 1-item sets L1 DB
For all values attr.value from db for the attribute attr.number

If the value in the db for the attribute attr.number is also in L1DB
Find support of attr.value in db
Add support of DB and db
If the support of DB and db is large in the updated database
Update the support count in the large 1-item sets
End If
Else If the value in db is not in L1DB
Find support of attr.value in db
If attr.value is large in db
Find support of attr.value in DB
Add support of DB and db
If the support of DB and db is large in the updated database
Update the support count of the attr.value in the large 1-item sets
End If
End if
UL1 DB + db= updated L1 DB + db for attribute attr
End for
Else If no insertions are done in the database
UL1 DB + db= L1 DB for attribute attr
End If
Generate the item set tree for UL1 DB + db.
Generate all other Lk s from L1 stored in item set tree as in TBAR Algorithm

Generate association rules from all the Lk s found in DB + db that are above the minimum confidence threshold

End ARU algorithm

The *attr* in the inputs for our algorithm shows us particular attributes that are large in the original database DB. In the step 5.1 we will check to see if there are any insertions in the database, if there are any insertions then all the L1 $_{DB}$ from the knowledge base are reused to find subsequent L_{ks} in DB + db. If there are no updates all L1 $_{DB}$ are taken as the final updated L1s.In subsequent steps these L1s are reused to find all Lk_s from the database.

VI. Experimental Studies

We have checked our algorithm with the TBAR algorithm for 1000 tuples with minimum support threshold from 1 to 5. As shown in Figure 2, ARU algorithm takes much less CPU utilization as compared to TBAR.

In the scale up experiments, we have checked the performance of our algorithm TBAR for 2 % minimum support and with 1000 to 5000 tuples. In Figure 3 it is clear that our algorithm gives linear results in nature, which means that it can be adapted to large databases. Our algorithm is 2.1 to 2.3 times faster than TBAR algorithm.

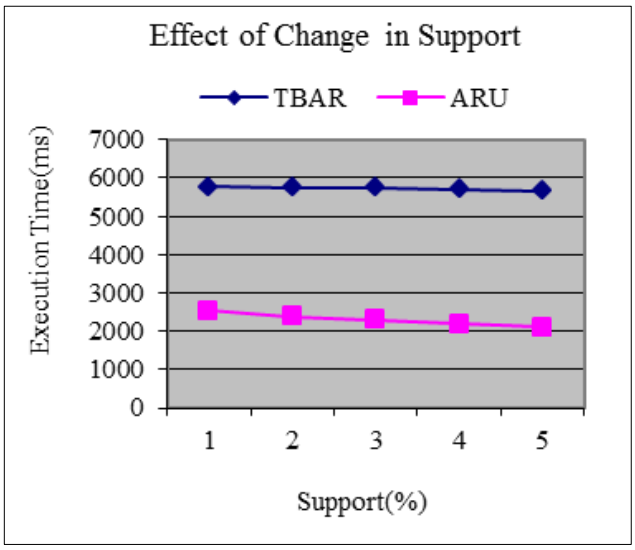

Figure 2. Effect of change in support.

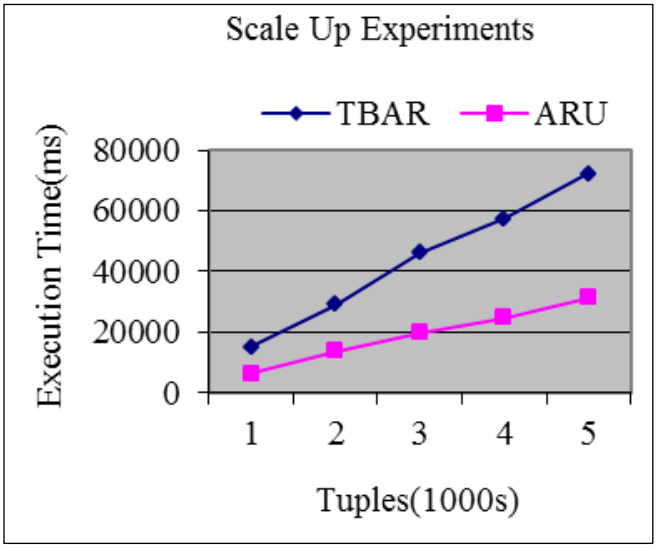

Figure 3. Scale up experiments.

VII. Conclusion

We have presented ARU algorithm, which outperforms the TBAR algorithm. Our proposed algorithm will be able to maintain large items sets by reusing the large item sets found through the initial mining algorithm. Our performance study shows that the proposed algorithm is 2.1 times to 2.3 times faster as compared to the TBAR algorithm. We found the incremental association rules from dynamic databases by employing a new knowledge management technique for relational databases. As a further step our knowledge management technique can be applied to other data mining techniques. Finding association rules from distributed databases is also important area of research.

Acknowledgment

The authors wish to acknowledge the financial support provided by the al Ghurair University.

REFERENCES

[1] R. Agrawal, T. Imielinski and A. Swami, Mining association rules between sets of items in large databases, Proceedings of the 1993 ACM SIGMOD International Conference on Management Of Data, Washington D.C., May 1993.

[2] Dogan and A. Y. Camurcu. "Association Rule Mining form an Intelligent Tutor", Journal of Educational Technology Systems, Volume 36, Number 4/2007 – 2008, pp 444 – 447, 2008.

[3] D.W. Cheung, J. Han, V.T. Ng and C.Y.Wong, Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique, 1996 International Conference on Data Engineering, New Orleans, Louisiana, February 1996.

[4] D. W. Cheung, V. T. Ng and B. W. Tam, Maintenance of Discovered Knowledge: A Case in Multi-level Association Rules, 2nd International Conference on KDD, Oregon, August 1996.

[5] J.S. Park, M.S. Chen and P.S. Yu, An effective hash-based algorithm for mining association rules, Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 1995.

[6] S. Sarawagi, S. Thomas and R. Agrawal, Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications, IBM Research Report, 1998.

[7] N.F. Ayan, A.U. Tansel, and E. Arkun. An Efficient Algorithm to Update Large Itemsets with Early Pruning. Proc. of 1999 Int. Conf. on Knowledge Discovery and Data Mining, 1999.

[8] D.Cheung,S.D.Lee and B.Kao.A General Incremental Technique for Updating Discovered Association Rules.Proc.International Conference On Database Systems For Advanced Applications,April 1997.

[9] H. Mannila, H. Toivonen and A.I. Verkamo, Improved Methods for Finding Association Rules, Department of Computer Science, University of Helsinki, Helsinki, Finland, December 1993 (Revised February 1994).

[10] TBAR: An efficient association rule mining for relational databases (1998).

[11] Chang-Hung Lee, Cheng-Ru Lin, and Ming-Syan Chen, Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining, ACM CIKM 2001.

[12] Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining — a general survey and comparison. SIGKDD Explorations,2 (1):58—64, July 2000.

AUTHORS PROFILE

Dr. Muhammad Abaidullah Anwar is working as Assistant Professor and Deputy Dean of College of Engineering and Computing in Al Ghurair University, UAE. He received his Doctorate of Engineering with specialization in Object-oriented Databases from Kyushu Institute of Technology, JAPAN in 2001. Since 2001, he has been affiliated with renowned universities in GCC and Pakistan. He has published many papers in International proceeding and journals.