A system for Incremental Association rule mining without candidate generation

Archana Gupta[1], Akhilesh Tiwari[2] and Sanjeev Jain[3]

1. **Research Scholar, Madhav Institute of Technology & Science, Gwalior (M.P.), India**
Email: archana100gupta@gmail.com
2. **Department of CSE & IT, Madhav Institute of Technology & Science, Gwalior (M.P.), India**
Email: atiwari.mits@gmail.com
3. **PDPM- Indian Institute of Information Technology, Design & Manufacturing, Jabalpur (M.P.), India**
Email: dr_sanjeevjain@yahoo.com

Abstract : Association rule mining can be used almost in all application for variety of purpose i.e. decision making, finding correlation among the items, to control dependent parameters and many more. Many standard algorithms work very well with respect to time and space complexity for the ARM. But if the transactional database is incremental then these standard algorithms impose huge time and space complexity. Thus for the incremental database some system is required which should not require the rescanning of existing database and multiple scanning of incremental database. It is known from the starting that the tree based ARM algorithms drastically reduce number of scans to the database and thus the time complexity. In this paper, a system is shown for incremental ARM which is based on the tree based data structure and requires different steps to generate association rules. The system consist of many phases from the phase 1 of taking input data to last phase of generation of Association Rules.

1. Introduction:

One of the objectives of data mining is to find the patterns that are hidden in large amount of data and to retrieve the related knowledge from the data for the analysis or improvement. Association Rule mining does not required much explanations, there are many ARM algorithms, models and technologies available. Some of the methods work very well for the static data such as Apriori Algorithm or FP tree algorithm which is based on tree based data structure and reduces number of scan to the database. However, these datamining methods doesnot work well with dynamicity in data. These algorithms cannot deal effectively and efficiently with the dataset that consist of old data as well as new data or modified data.

This paper presents a system for Association Rule mining to support dynamic data. Association Rule Mining works in two phase, first to find all the Frequent Itemsets and Second to generate Association Rules. Both phases are controlled by external parameters Support and Confidence respectively. This is called an efficient system as it handles dynamic data in all respect and converting the given input into some tree based datastructure to improve the efficiency of the system and to reduce the overall complexity of the system in terms of time and space. It also supports constraint mining.

Since input data is not getting used in its original form for the generation of frequent itemsets that's why a modules is added in the system to convert the given input in the COMVAN Tree []. To perform this conversion it is required to preprocess the data to achieve the same thus a module of preprocessing is added before conversion.

Since data is converted into COMVAN tree and all the required information is stored in tree only in different forms , thus a special algorithm is required to extract frequent itemsets from the tree.

Once frequent itemsets are available, Association rules can be generated based on the available supports.

This system is capable enough to handle dynamicity in the input data. Data can be updated, inserted and deleted easily from the system without any requirement of rescanning the old data unlike Apriori Algorithm and FP tree Algorithm for the ARM.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

The rest of the paper is organized as follows: section 2 gives the previous work done by different researcher in the field of association rule mining, section 3 gives the system proposed for incremental association rule mining, section 4 gives the detailed working of the system with respect to all the stages, the system will be finally concluded in section 5.

2. Literature survey:

Rakesh Agrawal et al. [1] introduces the problem of mining a large collection of basket data This paper introduces the problem of mining a large collection of basket data type transactions for association rules between sets of items with some minimum specified confidence, and presents an efficient algorithm for this purpose. An example of such an association ruleis the statement that 90% of transactions that purchase bread and butter also purchase milk. The antecedent of this rule consists of bread and butter and the consequent consists of milk alone. The number 90% is the confidence factor of the rule. This paper focuses on the problem of mining association rules between sets of items in a large database of customer transactions. Each transaction consists of items purchased by a customer in a visit. This work is to find the association based on the minimum support value and minimum confidence value.

Rakesh Agrawal et al. [3], considered the problem of discovering association rules between items in a large database of sales transactions. We present two new algorithms for solving this problem that are fundamentally dierent from the known algorithms. Empirical evaluation shows that these algorithms outperform the known algorithms by factors ranging from three for small problems to more than an order of magnitude for large problems. We also show how the best features of the two proposed algorithms can be combined into a hybrid algorithm called AprioriHybrid Scale up experiments show that AprioriHybrid scales linearly with the number of transactions. AprioriHybrid also has excellent scaleup properties with respect to the transaction size and the number of items in the database.

Jiawei Han et al. [8] proposed a novel frequent-pattern tree (FP-tree) structure, which is an extended prefix-tree structure for storing compressed, crucial information about frequent patterns, and develop an efficient FP-tree based mining method, FP-growth, for mining the complete set of frequent patterns by pattern fragment growth. Efficiency of mining is achieved with three techniques: (1) a large database is compressed into a condensed, smaller data structure, FP-tree which avoids costly, repeated database scans, (2) our FP-tree-based mining adopts a pattern-fragment growth method to avoid the costly generation of a large number of candidate sets, and (3) a partitioning-based, divide-and-conquer method isused to decompose the mining task into asset of smaller tasks for mining confined patterns in conditional databases, which dramatically reduces the search space. This study shows that the FP-growth method is efficient and scalable for mining both long and short frequent patterns, and is about an order of magnitude faster than the Apriori algorithm and also faster than some recently reported new frequent-pattern mining methods.

C. I. Ezeife Et al. [12] In this paper DB Tree is proposed which is a generalized form of FP Tree. It stores all items in the database as well as frequency count of all items in decreasing order of support. It is constructed in the same way as FP Tree except it includes all items instead of only frequent 1-iutemsets as in the FP Tree. Same as FP Tree, it needs exactly two scan of the transactional database to construct DB Tree. It will have more nodes as compare to the FP Tree as it includes all item of the transactional database irrespective of their count. It can be consider as special FP Tree with support = 0. Then also DB Tree is much compact then the original database because many transactions share the common path in the DB Tree.

Carson Kai-Sang Leung et al. [13] In this paper, a novel tree structure, called CanTree (Canonical-order Tree) is proposed, that captures the content of the transaction database and orders tree nodes according to some canonical order. By exploiting its nice properties, the CanTree can be easily maintained when database transactions are inserted, deleted, and/or modified. For example, the CanTree does not require adjustment, merging, and/or splitting of tree nodes during maintenance. No rescan of the entire updated database or reconstruction of a new tree is needed for incremental updating. Since its introduction, frequent-pattern mining has been the subject of numerous studies, including incremental updating. Many existing incremental mining algorithms are Apriori-based, which are not easily adoptable to FP-tree based frequent-pattern mining. The

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

construction of the CAN tree requires only one scanning of the transaction database unlike FP-Tree constructions where two scanning of database are required. In CAN tree constructions, all items in the transactions must be arranged in some canonical order, which is determined by the user prior to the mining process.

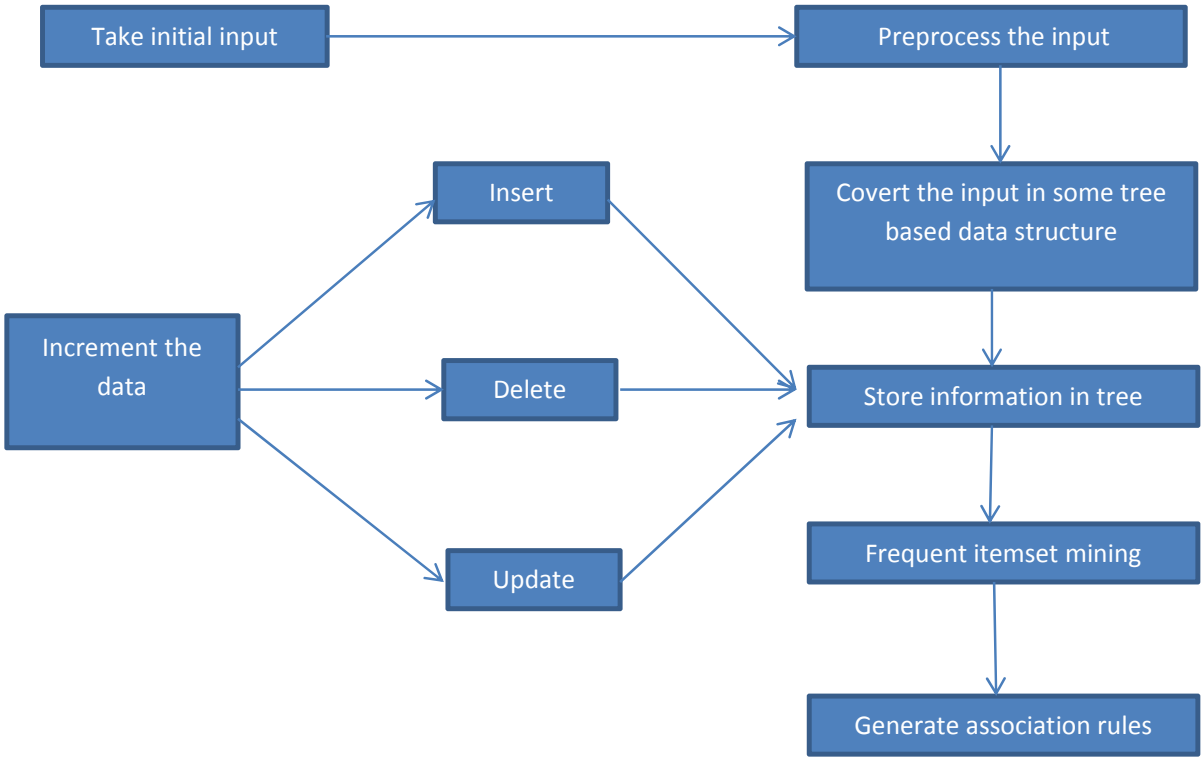
William Cheung et al. [14] In this paper, a novel data structure called CATS Tree is proposed. CATS Tree extends the idea of FP Tree to improve storage compression and allow frequent pattern mining without generation of candidate itemsets. The proposed algorithms enable frequent pattern mining with different supports without rebuilding the tree structure. There are many advantages of CATS Tree algorithms over the existing algorithms. 1) Once a CATS Tree is built, frequent pattern mining with different supports can be performed without rebuilding the tree. The benefit of "build once, mine many" increases with the number of frequent patterns mining performed, i.e., interactive mining with different supports; the cost of CATS Tree construction is amortized over multiple frequent patterns mining. 2) CATS Tree allows single pass frequent pattern mining. 3) CATS Tree algorithms allow addition and deletion of transactions in the finest granularity, i.e., a single transaction.

3. Proposed System:

In the proposed system, initially data preprocessing is done to fulfil the requirement of application on of the system. COMVAN tree is used to store the database and its required information such as count, itemsets, etc. The proposed system is a complete model to generate association rules for the incremental data. This system is efficient to handle increment data. There is no loss of data while converting the database into COMVAN tree. After getting frequent itemsets based on the given support, this system is capable of deriving association rules.

3.1 Architecture Diagram:

The architectural diagram of the proposed system is as below:



https://sites.google.com/site/ijcsis/
ISSN 1947-5500

3.2 List of Modules:

- Data preprocessing
- COMVAN tree Generation
- Reflect the hidden information of the datasets
- Apply FEEPAMT algorithm for Frequent itemset generation
- Deriving association Rules
- Handling of Increment data

4. COMVAN : tree based data structure:

COMVAN tree construction is explained in detail in []. It is a tree based data structure that is dependent on the order of items that are defined prior to the construction of tree. COMVAN tree has many feature that are given as below:

a. Initially Root will point to the first item in the first record..

b. Associated with every node in the tree, a list of values is there. A value at a particular position k represents the total number of records in which that item is at $(n-k+1)^{th}$position where n is the level at which that node is present in the tree.

c. Every next item in the record will be added as a child of the preceding item in the record.

d. If the item is already there in the children of the preceding node then no new node will be added, only the corresponding value in the list will get increment by 1.

e. If the first item of the record is not in the children of the root, then it will search for the first item in the tree using DFS. Once item is found in the tree, then it will follow the same procedure to enter the record in the tree. If that item is not found in the tree then it will be added as a child of the root.

f. The length of the list associated with every node will depend on the level of the tree at which that node is located.

4.1 COMVAN Algorithm:

Algorithm : COMVAN Tree Builder

Input : Transactional Database, lexicographic order of the items given by order[].

Output :COMVAN Tree structure

```
while (end of the database)
{
read one record of n itemset at a time;
separate all items of the record , given by tupleitem[]
//items are arranged as per predefined order

If TupleItem [1] is at order[1] then
{
search for the TupleItem[1] at level 1;

If it is available at level 1 then
        Update list corresponding to item;
Else
{
        Add a child node to the root with name TupleItem[1];
        update list;
}
        }
```


https://sites.google.com/site/ijcsis/
ISSN 1947-5500

```
Else
{
search in tree for TupleItem[1] using DFS till level r, where TupleItem[1] is at order r,
if it is at level f then
        update the corresponding value in list of that node;

Else

        Add a child node to the root with name TupleItem[1];
        Update its list value;
}
For k = 2 to n
{
If (TupleItem[k] is in the children of TupleItem[k-1]) then
        Update corresponding list value
 Else
        Create a child node of  TupleItem[k-1]  with  label  TupleItem[k]
        Update corresponding list value;

   }
}
```

4.2 Example dataset:

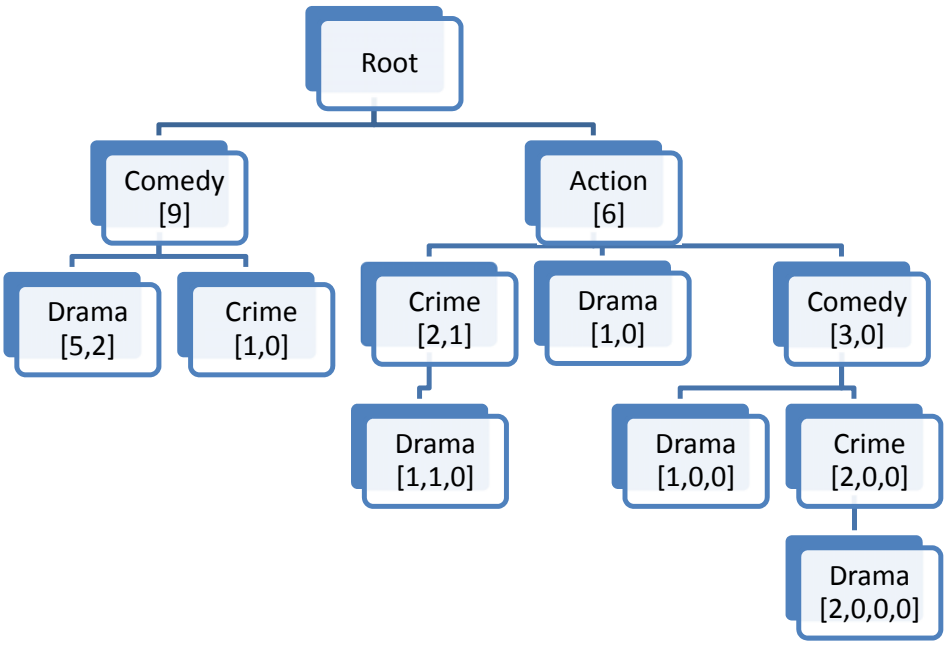
Following dataset of movies genres is considered to explain the working of the system:

movieId	Title	genres				
1	Toy Story (1995)	Drama	Animation	Children	Comedy	Fantasy
2	Jumanji (1995)	Adventure	Children	Fantasy		
3	Grumpier Old Men (1995)	Comedy	Romance			
4	Waiting to Exhale (1995)	Comedy	Drama	Romance		
5	Father of the Bride Part II (1995)	Comedy				
6	Heat (1995)	Action	Crime	Thriller		
7	Sabrina (1995)	Comedy	Romance			
8	Tom and Huck (1995)	Adventure	Children			
9	Sudden Death (1995)	Action				
10	GoldenEye (1995)	Action	Adventure	Thriller		
11	American President, The (1995)	Comedy	Drama	Romance		
12	Dracula: Dead and Loving It (1995)	Comedy	Horror			
13	Balto (1995)	Adventure	Animation	Children		
14	Nixon (1995)	Drama				
15	Cutthroat Island (1995)	Action	Adventure	Romance		
16	Casino (1995)	Crime	Drama			
17	Sense and Sensibility (1995)	Drama	Romance			
18	Four Rooms (1995)	Comedy				
19	Ace Ventura: When Nature Calls (1995)	Comedy				
20	Money Train (1995)	Action	Comedy	Crime	Drama	Thriller
21	Money Train (1998)	Action	Comedy	Crime	Drama	Thriller

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

4.3 COMVAN Tree:

After considering the genres Action, Comedy, Crime, Drama in this order only, resultant COMVAN tree is:



5. Modules Description:

5.1 Data preprocessing

During pre-processing of the data arrange all the dataitems in every transaction in the order taken as input during phase 1. That order matters while generation of tree based data structure. And that order is one of the important components that controls the entire working of the system.

Consider the Example database given in section 4.2 and the same order that was used to generate COMVAN tree.

Result after preprocessing is:

genres			
Comedy	Drama		
Comedy			
Comedy	Drama		
Comedy			
Action	Crime	Drama	

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Comedy	Drama		
Action	Drama		
Action	Crime		
Comedy	Drama		
Comedy			
Drama			
Action	Comedy	Drama	
Crime	Drama		
Drama			
Comedy	Crime		
Comedy	Drama		
Action	Comedy	Crime	Drama
Action	Comedy	Crime	Drama

5.2 COMVAN Tree generation

It is explained in detail in section 4. The database considered in section 4 is after preprocessing and the result of data preprocessing is shown in section 5.1.

5.3 Reflecting hidden information of database:

All the itemsets along with their frequency count is stored in COMVAN tree in the form list, set of values attached with every nodes.

5.4 Generate frequent itemsets using FEEPAMT

Unlike FP-Tree, once the COMVAN tree is constructed, it can be mined repeatedly for frequent patterns with different support threshold without any need to rebuild the tree. perform constraint mining can be performed very easily here by using COMVAN Tree. In this FEEPAMT algorithm, initially all the itemsets are generated from the COMVAN tree with their respective support and then it is pruned to generate frequent itemsets based on the minimum threshold.

Algorithm : FEEPAMT(σ)
Input : a COMVAN Tree and required support
Output: a set of frequent itemset
// Initially control will be at root
{
First = (Root, 0 , 0)
//where Root is list of items
Add First in Openlist
While Openlist != Null
{
Take First entry of Openlist ;
Traverse in COMVAN tree till leaves to get the itemsets and its frequency count;
Let k is the level of the first item of the itemset found.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

If for any node say N, its list value for the positions >=k is not null then

New enteries will be made in Openlist;

}

// calculate actual support of the frequent itemset

While all entries not marked done

{

Pick first unmarked entry from Candidateset given by (X,Y,Z);

Search for X in the remaining enteries of Candidateset given by (A,B,C);

If X is not prefix of A but present in A then

Support of both enteries will be added to give the support of X;

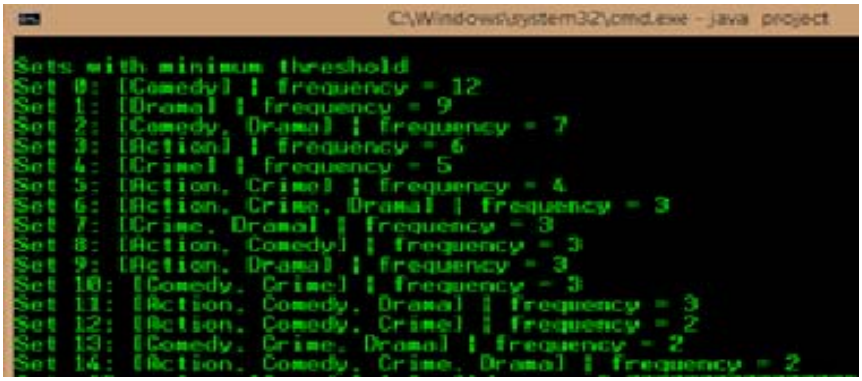
}

// Find actual Frequent Itemsets

For all enteries (X,N) in Frequencylist apply check on N with respect to σ and find the frequent itemsets.

}

Frequent itemsets generated for the example dataset for σ=10% is shown below:



5.5 Generate association rules

Association rules will get generate from the generated set of frequent itemsets and confidence of the rule is calculated by using the simple concept of mining of Association rules.

For the example dataset following association rules are generated along with their confidence:

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

6. Increment the data:

In this phase, new data can be appended in the old database through a file or by giving one transaction at a time as input. And this new transaction will get stored in the COMVAN tree without any need of rescanning the entire database.

In addition of the insertion of new transactions, this system supports deletion of some transaction as well as updation of some transaction.

During deletion, a transaction will get delete from the database, thus the respective count will get decrement by 1 as result.

During updation a transaction is getting modified, since COMVAN tree is not maintaining transaction id thus updation should delete the old association and add the new association based on the updated transaction.

7. Conclusion:

In this paper, a system is proposed for Association rule mining based on an efficient tree base data structure to support incremental data. COMVAN tree representation of the transactional database reduces number of scanning to the transactional database and hence reduces complexity drastically. COMVAN [15] representation of database is efficient with respect to time and space than many other tree based Data Structures proposed by different Researchers[8,1011,12,14]. This system starts with data preprocessing and during data processing number of items that needs to be consider, name of the items and order of the items can be decided.

References:

1. Agrawal, R., Imielinski, T. and Swami, A. “Mining Association Rules between Sets of Items in Large Database”. Proceedings of the ACM SIGMOD conference on management of data, Washington, D.C, May 26-28, 1993.
2. M. Houtsma and A. Swami. “Set-oriented mining of association rules”. Research report RJ 9567, IBM Almaden Research Center, San Jose, California, October 1993.
3. Agrawal R and Srikant, R., “Fast algorithms for mining association rules”, Proceedings of the 1994 Int. Conf. Very Large Data Bases, pp. 487-499, Santiago, Chile, September 1994.
4. Savasere A., E. Omieccinski and S. Navathe, “ An efficient algorithm for mining association rules in large database”. Proceeding of the 21st International Conference on very large database, Zurich, Switzerland, Sept 11 – 15 , pp 432-443.
5. Brin, S. R. Motwani, J. D. Ullman and S. Tsur, “ Dynamic Itemset Counting & Inplication rules for market basket data”. Proceedings of the ACM SIGMOID Int. Conf. Manage Data, vol. 6(2) 1997, 26: 255-264.
6. Toivonen H., “Sampling Large database for Association Rules”. Proceedings of the 22nd International Conference on very large database, Bombay, India, Sept 3–6, 1996 pp 134-145.
7. Siddharth Shah, N. C. Chauhan, S. D. Bhanderi, “Incremental Mining of Association Rules: A Survey”. Proceedings of the International Journal of Computer Science and Information Technologies, Vol. 3 (3) , 2012, pp 4071-4074.
8. Han J., J, Pei, Y. Yin and R. Mao”, “ Mining frequent patterns without candidate generation: A frequent pattern tree approach”. Proceedings of the Data Mining Knowledge Discovery, 2004, 8 : 52-87.
9. D. Cheung, J. Han, V. Ng, and C. Y. Wong. “Large Databases: An Incremental Updating Technique”. Proceedings of the 12th International Conference on Data Engineering, February 1996, pp 106—114.
10. D. Cheung, S. D. Lee, and B. Kao. “A General Incremental Technique for updating Discovered Association Rules”. Proceedings of the Fifth International Conference On Database Systems for Advanced Applications, April 1997, pp 185—194.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

11. Chin-Chen Chang, Yu-Chang Li and Jung-San Lee, " An Efficient Algorithm For Incremental Mining of Association Rules", Proceedings of the 15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications (RIDE-SDMA'05), 2005.
12. C. I. Ezeife and Y. Su. "Mining Incremental Association Rules with Generalized FP-Tree". Proceedings of the 15th Canadian Conference on Artificial Intelligence, May 2002.
13. C. K. Leung, Q. I. Khan and T. Hoque. "CanTree: A Tree Structure for Efficient Incremental Mining of Frequent Patterns", Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05), 2005.
14. 19W. Cheung and O.R. Zaiane , " incremental mining of frequent patterns without candidate generation or support constraint", Proceedings of IDEAS 2003, pp. 111-11.
15. Archana Gupta, Sanjeev Jain, Akhilesh Tiwari, "COMVAN: A Novel Data Structure for Storing Large Database for Incremental Association Mining", TECHNIA – International Journal of Computing Science and Communication Technologies, VOL.9 NO. 2, January. 2017 (ISSN 0974-3375), pp 1110-1113.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500