Implementation of Improved Association Rule Mining Algorithms for Fast Mining with Efficient Tree Structures on Large Datasets

P.Naresh, R.Suguna

Abstract- ARM is a significant area of knowledge mining which enables association rules which are essential for decision making. Frequent itemset mining has a challenge against large datasets. As going on the dataset size increases the burden and time to discover rules will increase. In this paper the ARM algorithms with tree structures like FP-tree, FIN with POC tree and PPC tree are discussed for reducing overheads and time consuming. These algorithms use highly competent data structures for mining frequent itemsets from the database. FIN uses nodeset a unique and novel data structure to extract frequent itemsets and POC tree to store frequent itemset information. These techniques are extremely helpful in the marketing fields. The proposed and implemented techniques reveal that they have improved about performance by means of time and efficiency.

Keywords: Association Rule Mining, FP-tree, POC tree, PPC tree

I. INTRODUCTION

The process of extracting required information and new patterns form datasets is termed as knowledge mining [1]. From those derived information and itemsets we can able to generate Association rules and correlations which intern disclose some relationship and linking of connection among items in transaction. This may further helpful in several areas like marketing, forecasting to make analysis and to improve business.

ARM is composed into two passes. In first pass FIM will be done and in second pass it is possible to generate Association rules [13]. The major task in ARM is to fetch the frequent items from the dataset and it is also an important task [23]. Hence the majority of the research was done in this regard.

In recent years utility mining model [5] [6] was initiated to find out some significant information from the given datasets. In these kinds of mining significant of itemset was measured by their utility.

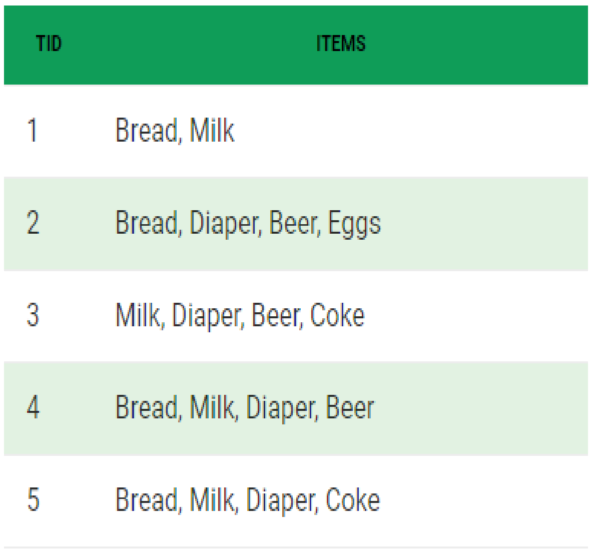
In general ARM needs the help of two qualitative measures in the process of rules generation. First one is support and second one is confidence. These may strengthen the mining process and outputs the required rules.

$$\text{Support } [X \rightarrow Y] = \frac{Transactions\ containing\ both\ X\ and\ Y}{Total\ no\ of\ transactions}$$

$$\text{Confidence } [X \rightarrow Y] = \frac{Transactions\ containing\ both\ X\ and\ Y}{Transactions\ containing\ X}$$

Support: This gives the occurrence of the item in the given dataset.

Confidence: This explains how likely **Y** is purchased when **X** is purchased.



TID	ITEMS
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Rules: Bread$\leftrightarrow$Milk
Diaper$\leftrightarrow$Beer

Fig 1: Transactions with association rules

All the fundamental association rule mining algorithms work on chronological approach [14] and they were capable until the volume of the dataset was small. As the amount of datasets started growing, their effectiveness starts decreasing. Therefore, to handle large datasets, parallel algorithms [1] were introduced. Finding out the frequent itemsets in a huge dataset [17] is a vital difficulty in knowledge mining. Now a day's these kinds of issues was increased while dealing with big datasets and large volume databases [6]. And also it was a main concern where dataset have different attribute values, missing and noisy data.

The rest of this research paper elucidates related work, implementation part of algorithms on datasets, results-discussions and conclusion-future scope of the research.

Revised Manuscript Received on December 30, 2019.
* Correspondence Author

P.Naresh, Research Scholar, Dept of CSE, Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology (Deemed to be University), Chennai.
Dr.R.Suguna*, Professor, Dept of CSE, Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology (Deemed to be University), Chennai.

II. RELATED WORK

Han et.al invented the FP-growth algorithm for mining. FP-growth [5] follows a depth first approach. It maintains a tree structure which had left node and right node, every node represents an item related to a transaction. Every node has node count with respect to all repeated items of as dataset. In this data was compressed in tree format and this algorithm needed two dataset scans for mining. No candidate key generation was needed for this. FP-Growth [11] is much faster than other algorithms. Optimizations techniques are require more to reduce time. Zaki proposed a depth first approach like FP for mining process.

FP-growth Algorithm:

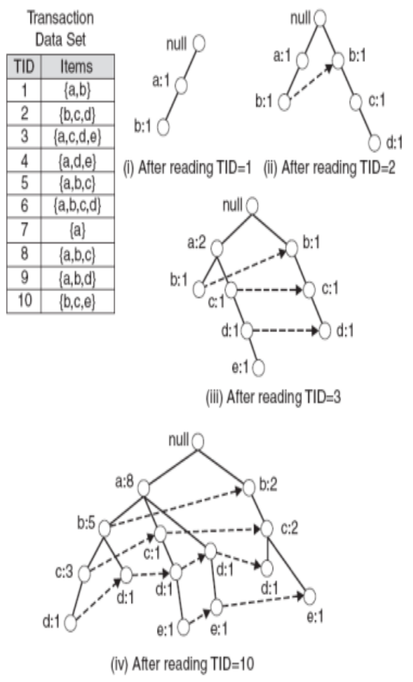
Constructed FP-Tree in 2 passes

Pass 1:

Scan the database for frequent itemsets.
Sort frequent items.

Pass 2:

Read transactions and do mapping.
Extract frequent items



TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

Fig 2: FP-tree Mining

FIM [18] was mainly anticipated for analysis of customer interests depending on their transactions.

In recent years so many frameworks were proposed for representing an itemset [15]. Every transaction items were read from dataset and a list of nodes were maintained as the prefix tree [8]. This item set representation has to maintain a) adequate information about the items b) sum counts of items in all transactions; it gives the support of each item c) the node list of (k+1) item was obtained by intersection of two itemsets. These may improve the mining process efficiently as they use node-lists.

FIN:

Zhi-Hong Deng proposed a fast FIM using Nodesets [2] in the year 2014. A nodeset is datastructure used in mining which inturn generates a tree structure. It also do the pruning to stay away from repetition.

PPC-Tree:

PPC tree consists of a root node with null values and child's for that root. Every node except root node has pre-order [2], post-order [2], item name, count and child's list. Here pre-order responsible for storing order rank of the node in the tree, post-order stores rank of node, item name is the unique one for which the particular item belongs, count represents a integer value of children count of a particular node and children list maintains all child's of a tree.

PrePost algorithm works as follows:

- Build a PPC tree
- Iidentifying itemsets with 1-frequency
- Construct the N-list for frequent-itemset according to PPC tree
- Scan the PPC tree to find frequent 2-itemsets
- Mine every frequent k-itemsets from dataset

Definition of N-list:

While considering of every node N in the PPC-tree, the Pre and Post code of N, the N-list [7] of the frequent item is the sequence of every Pre and post codes of nodes adding them in the PPC-tree. The Pre and Post codes are arranged in an ascending order of their pre-order values.

Transactional dataset

The below given dataset contains five transactions and respective items and they are ordered according to their supports.

ID	Items	Ordered frequent items
1	a, f, g	a, f
2	a, b, c, e	b, c, e, a
3	b, c, e, i	b, c, e
4	b, c, e, h	b, c, e
5	b, c, d, e, f	b, c, e, f

The PPC-tree generated having root node and child nodes depending on frequency and support. The generated below tree contains a null root node with value, key pair as (0, 7) and left two children nodes and right side five children nodes

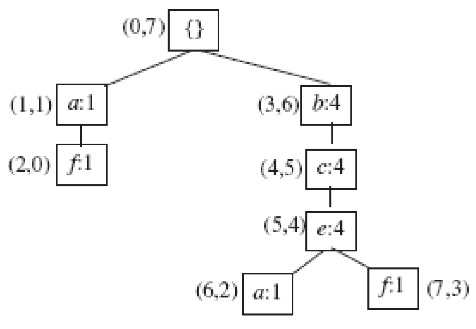


Fig 3: PPC –Tree

POC Tree:

POC Tree algorithm works as follows:

- Build a POC tree
- Iidentifying itemsets with 1-frequency
- Construct the Nodeset for frequent-itemset according to POC tree
- Scan the POC tree to find frequent 2-itemsets
- Mine every frequent k-itemsets from dataset

Definition of Nodeset

Each and every node N in POC tree, has a pair of value one is pre order and the other is the count and it is represented as N-info of N. Nodeset [9] of the frequent item i is the series with every N-info's about the nodes within the POC tree.

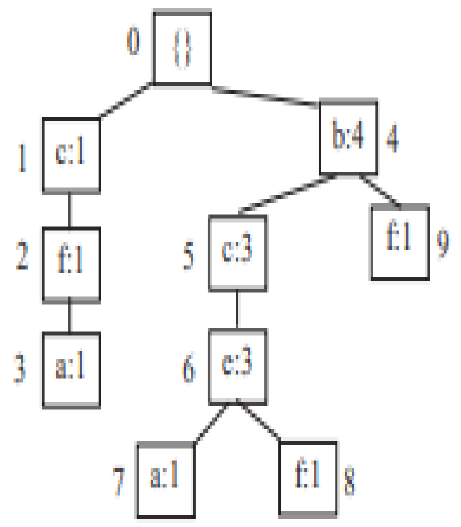


Fig 4: POC Tree

In the above POC, there is a root node present with null value also had its six right side child's and three left side Childs

III. IMPLEMENTATION

FP-growth [5] is a tree based frequent generation approach which does not contain any candidate key. It was very efficient algorithm by means of memory and efficiency. A FP-tree [18] [19] had dense structure which represents data items in the tree format. In this every transaction from the dataset was read and it mapped with a path in FP-tree [11] [12]. This process continues until all the transactions were read from the dataset. Different transactions which have common items allow the tree to stay behind squashed to avoid overlapping.

FP-Growth algorithm

To construct FP-tree requires the following passes

Pass 1:

- Scan data and find support for each item and Discard infrequent items.
- Sort frequent items in decreasing order based on their support.

Pass 2:

- Nodes correspond to items and have a count value
- FP-Growth reads 1 transaction at a time and maps it to a path
- Fixed order is used, so paths can overlap when transactions share items
- In this case, counters are incremented
- Pointers are maintained between nodes containing the same item, creating singly linked lists
- Frequent itemsets extracted from the FP-Tree.

FIN with POC and PPC

Despite the fact that node list and n list consists novel structures for mining process. These lists have to encode every node of PPC or POC tree which requires more space and time for frequent items generation [21][22]. To overcome these drawbacks a nodeset structure was proposed to improve the time-space tradeoffs. The nodeset proves that it was better by means of speed and performance.

FIN algorithm generates [2] all frequent itemsets which are present in the given dataset. Fin algorithm first scan the given dataset and produce a tree and then it visit each node in the tree and calculates its support after that it moves to next node and do the same method until all nodes in the tree were traversed. Finally it displays all the generated FI's along with their support. From those frequent items we can generate association rules for analysis.

The FIN algorithm as follows

```
Algorithm 2: FIN Algorithm
Input: A transaction database DB and a minimum support $\xi$.
Output: F, the set of all frequent itemsets.
(1)      $F \leftarrow \emptyset$;
(2)      Call Algorithm 1 to construct the POC-tree and find $F_1$,
       the set of all frequent 1-itemset;
(3)      $F_2 \leftarrow \emptyset$;
(4)      Scan the POC-tree by the pre-order traversal do
(5)        $N \leftarrow$ currently visiting Node;
(6)        $i_y \leftarrow$ the item registered in $N$;
(7)        For each ancestor of $N$, $N_a$, do
(8)          $i_x \leftarrow$ the item registered in $N_a$;
(9)          If $i_x i_y \in F_2$, then
(10)           $i_x i_y.support \leftarrow i_x i_y.support + N.account$;
(11)         Else
(12)           $i_x i_y.support \leftarrow N.account$;
(13)           $F_2 \leftarrow F_2 \cup \{i_x i_y\}$;
(14)         Endif
(15)       Endfor
(16)     For each itemset, $P$, in $F_2$ do
(17)       If $P.support < \xi \times |DB|$, then
(18)         $F_2 \leftarrow F_2 - \{P\}$;
(19)       Else
(20)         $P.\ Nodeset \leftarrow \emptyset$;
(21)       Endif
(22)     Endfor
(23)   Scan the POC-tree by the pre-order traversal do
(24)     $Nd \leftarrow$ currently visiting Node;
(25)     $i_y \leftarrow$ the item registered in $Nd$;
(26)     For each ancestor of $Nd$, $Nd_a$, do
(27)       $i_x \leftarrow$ the item registered in $Nd_a$;
(28)       If $i_x i_y \in F_2$, then
(29)         $i_x i_y.Nodeset \leftarrow i_x i_y.Nodeset \cup Nd.N\_info$;
(30)       Endif
(31)     Endfor
(32)     $F \leftarrow F \cup F_1$;
(33)     For each frequent itemset, $i_s i_t$, in $F_2$ do
(34)     Create the root of a tree, $R_{st}$, and label it by $i_s i_t$;
(35)       Constructing_Pattern_Tree($R_{st}$, $\{i \mid i \in F_1, i \succ i_s\}$, $\emptyset$);
(36)     Endfor
(37)     Return F;
```

POC Tree is generated depending on pre-ordering of visiting of all items present in each transaction in the given database.

POC Tree Algorithm

Input : A transaction database DB and a minimum support n.

Output A POC-tree

POC-tree Construction

- Create the root of a POC-tree
- For each transaction Trans in DB
- Select the frequent items in Trans and sort out them according to the order of F1.
- Let the sorted frequent-item

  list in Trans be [p | P], where p is the first element and P is the remaining list.
- Call insert tree ([p | P],Tr).

  The function insert tree ([p | P], Tr)
- If Tr has a child N such that N.item-name = p.item-name,
- then increase N's count by 1

  else
- create a new node N, with its count initialized to 1

  and add it to Tr's children-list.
- If P is nonempty,
- call insert tree(P, N) recursively.
- End if
- End if

PPC Tree is generated depending on pre-ordering and post ordering of visiting all items present in each transaction in the given database.

PPC Tree Algorithm

Input : A transaction database DB and a minimum support s.

Output: A PPC-tree and F1 (the set of frequent 1-itemsets).

PPC-tree construction

- Create the root of a PPC-tree, T r , and label it as "null".
- for each transaction T rans in DB do
- Select the frequent items in T rans and sort out them according to the order of

  F1.
- Call insert tree([p | P],Tr)

  end
- Scan PPC-tree to generate the pre-order and the post-order of each node.
- Function insert tree([p | P],Tr)
- if Tr has a child N such that N.item-name = p.item-name then
- increase N's count by 1

  else
- create a new node N, with its count initialized to 1

  and add it to T rs children-list
- if P is nonempty then
- call insert tree(P,N) recursively.
- end if
- end if

After tree has been constructed then association rules generated depending on their support values.

IV. RESULTS AND DISCUSSIONS

In this paper experiments were made by considering consistent datasets [16] [20] taken from the site "https://archive.ics.uci.edu/ml/index.php"

Dataset	Min Sup	No of Instances	No of Attributes	Size
mushroom	5	8124	22	356 KB
connect	40	67557	42	5829 KB

Execution time was considered for Analysis of various algorithms with varying parameters. The unit of measurement for the execution time is Seconds.

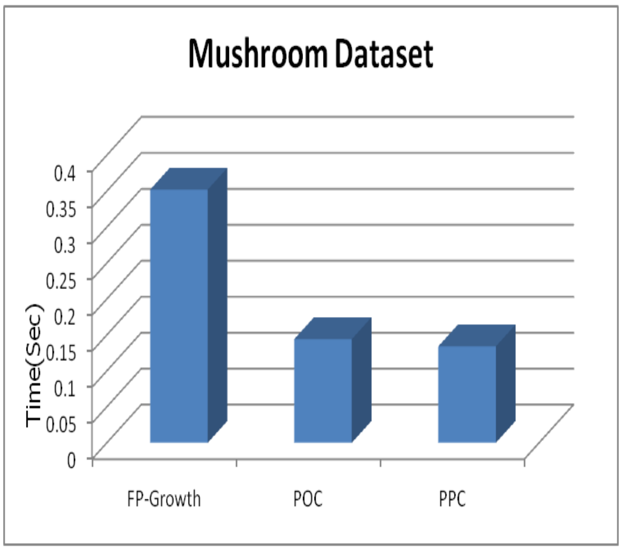


Fig 5: Time comparison of algorithms for Mushroom (small) Dataset

For Mushroom dataset Fp-growth took 0.34 sec, Fin-POC tooks 0.14 sec and Fin-PPc tooks 0.11 Sec. For Small Datasets FIN algorithm took less time and it shows better performance.

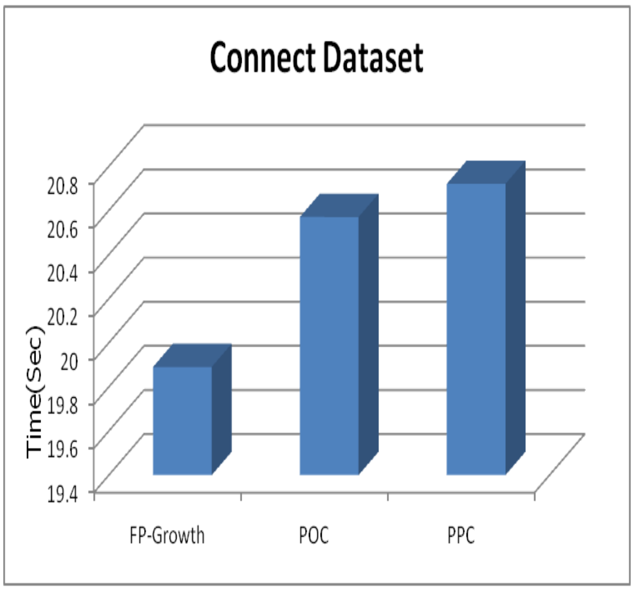


Fig 6: Time comparison of algorithms for Connect (Large) Dataset

For connect dataset Fp-growth took 19.8 sec, Fin-POC tooks 20.4 sec and Fin-PPc tooks 20.6 Sec. For large datasets FIN algorithm took somewhat more computing time than the Fp-growth.

V. CONCLUSIONS AND FUTURE SCOPE

This research paper explores and entails the frequent itemset mining algorithms for rules generation are implemented and analyzed with the appropriate datasets. All algorithms have their bests and worst's depending on the dataset size and types of data being present. FP-tree, FIN with POC tree and PPC tree were implemented in python and results are tabulated. Compared to all FIN algorithm took less time and more efficient.

In future the research extends to mine frequent itemsets fastly and also dynamically generates rules against record insertions for large datasets.

REFERENCES

1. Ms. Vinaya Sawant, Dr. Ketan Shah, "Performance Evaluation of Distributed Association Rule Mining Algorithms", 7th International Conference on Communication, Computing and Virtualization, pp. 127 – 134, 2016.
2. M.Ranjanisindu, Dr.S.Gunasekaran, Ms.N.R.Deepa, "Frequent Pattern Mining Algorithms PrePost, FIN, H-Mine – A Survey", International Journal of Innovative Research in Science, Engineering and Technology, pp.797-803, 2019.
3. L. Luna, et al., "Optimization of quality measures in association rule mining: an empirical study," International Journal of Computational Intelligence Systems. vol. 12, no. 1, pp. 59–78, 2018.
4. J. Rajni, M. D. Borah. "A novel approach for mining frequent patterns from incremental data," International Journal of Data Mining, Modelling and Management 8.3, 2016, pp. 244–264.
5. Gosta Grahne, IEEE, and Jianfei Zhu, IEEE, "Fast Algorithms for Frequent Itemset Mining Using FP-Trees", IEEE Transactions On Knowledge and Data Engineering, VOL. 17, NO. 10, pp.1347-1362 OCTOBER 2005.
6. Qiu, Ping, Long Zhao, and Xiangjun Dong, "NegI-NSP: Negative sequential pattern mining based on loose constraints," IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society. IEEE, pp. 3419–3425, 2017.
7. Zhi-Hong Deng , Sheng-Long Lv, "PrePost+: An efficient N-lists-based algorithm for mining frequent itemsets via Children–Parent Equivalence pruning", Expert Systems with Applications 42 (2015) 5424–5432.
8. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases", Research Report RJ 9839, IBM Almaden Research Center, San Jose, California, June 1994.
9. Zhi-Hong Deng, Sheng-Long Lv, "Fast mining frequent itemsets using Nodesets", Expert Syst. Appl. 41(10): 4505-4512 (2014).
10. Lichun Li, Rongxing Lu, Kim-Kwang Raymond Choo, Anwitaman Datta, and Jun Shao. (2016), "Privacy-Preserving-Outsourced Association Rule Mining on Vertically Partitioned Databases", IEEE Transactions on Information Forensics and Security. 11 (8), p1-15.
11. Kanwal Garg, Deepak Kumar, "Comparing the Performance of Frequent Pattern Mining Algorithms", (IJCA) International Journal of Computer Applications (0975 – 8887) Vol. 69, No.25, May 2013.
12. Agarwal, R.C., Agarwal, C.C., and Prasad, V.V.V., "A tree projection algorithm for generation of frequent item sets", Journal of Parallel and Distributed Computing, vol.61, pp.350–371, 2001.
13. Goswami, D.N., et., al., "An Algorithm for Frequent Pattern Mining Based On Apriori ", (IJCSE) International Journal on Computer Science and Engineering, Vol.02, No.04, Pp. 942-947, 2010.
14. mining frequent itemsets algorithms. Int J Mach Learn Cybern, pp 1–11.
15. UCI: mushroom data set. http://www.cs.sfu.ca/wangk/ucidata/dataset/mushroom.
16. Goethals B, Le Page W, Mampaey M (2010) Mining interesting sets and rules in relational databases. In: Proceedings of the ACM symposium on applied computing, ACM, pp 997–1001.
17. Duan L, Street WN (2014) Speeding up maximal fully-correlated itemsets search in large databases. Int J Mach Learn Cybern,pp 1–11.
18. C. Lucchese, S. Orlando, and R. Perego, "Fast and Memory Efficient Mining of Frequent Closed Itemsets," IEEE Trans.Knowledge and Data Eng., vol. 18, no. 1, pp. 21-36, Jan. 2006.

AUTHOR'S PROFILE

P.Naresh, received his B.Tech and M.Tech in Computer Science and Engineering from Jawaharlal Nehru Technological University, Anantapur, Andhra Pradesh, during 2011 and 2013 respectively. Currently pursuing and science engineering at Vel Tech Rangarajan Dr Sagunthala R&D Institute of Science and Technology (Deemed to be university), Chennai, Tamil Nadu. His interesting research areas are Data Mming Big data and cloud computing

Dr. R. Suguna, received B.E. and Science M.Tech in Computer Science Engineering from Thiagarajar College of Engineering and Indian Institute of Technology Madras during the year 1989 and 2004 respectively. She completed doctorate from Anna University in 2011. She is working as professor in Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and ITechnology (Deemed to be university) in CSE Dept. She has 22 years of teaching experience and 6 years in industry. She has published 50+ research papers in reputed joumals and conferences. She has delivered invited talks and chaired the sessions in intemational conferences. Her research interest includes image processing, data mining and machine leaming Cunently 10 research scholars are under her supervision. She has organized workshops in machine leaming and deep leaming in anous institutions. She is an active member of IEEE and CSI.