Volume 9, Special Issue No. 1, February 2018

ISBN: 978-93-5300-455-2

International Journal of Advanced Research in Computer Science
(ISSN: 0976-5697)
RESEARCH PAPER

SHRI SHANKARLAL SUNDARBAI SHASUN JAIN COLLEGE FOR WOMEN
ICRACC 2018
(23rd–24th February 2018)

Available Online at www.ijarcs.info

A HYBRID ALGORITHM FOR MINING FREQUENT ITEMSETS IN TRANSACTIONAL DATABASES

Ramah Sivakumar
Research Scholar
Department of Computer Science
Bishop Heber College
Trichy-17, India, Email: rmhsvkmr@yahoo.co.in

Dr.J.G.R.Sathiaseelan
Research Advisor
Department of Computer Science
Bishop Heber College
Trichy-17, India

ABSTRACT

Frequent pattern mining is one of the most notable areas under research. Mining frequent itemsets in transactional databases paves way for business improvements. In this paper, a hybrid algorithm called CanTree is proposed, which is based on the classic Apriori and FPGrowth. The proposed algorithm has been derived by improving the existing advantages of both the algorithms and avoiding the recursive generation of conditional pattern bases and sub conditional pattern trees which is the main disadvantage in FPGrowth. The proposed algorithm has been examined by comparing the results with the existing algorithms. The parameters taken for analyzing are time, and memory space. Four different real time datasets with varied sizes from the UCI and Frequent Itemset Mining Implementations Repository (fimi) were used for the experiments. The result shows that the proposed algorithm gives betterment in the mining process of frequent itemsets than the existing algorithms.

Keywords: Frequent Itemset mining, Apriori, FPgrowth, CanTree.

1. Introduction

Itemset Mining plays an important role in the field of pattern mining. Itemset is a set of items in a particular transaction.The variations in frequent patterns are long patterns, interesting patterns, sequential patterns, graph patterns, uncertain frequent patterns spatiotemporal patterns and so on. Researches are ongoing process in this field to mine these types of patterns. Other than association rules, mining frequent patterns leads for effective classification, clustering, predictive analysis and so on. Frequent pattern mining has wider application areas such as software bug detection, network analysis, customer analysis, clustering, classification, outlier analysis, indexing, web log mining, chemical and biological applications and so on. Finding the frequent itemsets leads to form associations among the items in the transaction. Further association rules can be framed which can then be mined. Mining association rules paves way for betterment of the businesses. So the key aspect in the process is mining frequent patterns.

1.1 Proposed Work

For mining frequent patterns, different types of algorithms are presently available. Some are join-based algorithms, and some are tree based algorithms and other techniques are also available in their optimizations. Apriori is the best example for join based algorithms and FPGrowth is for tree based algorithms. Each has their own pros and cons. In this paper a new algorithm called as CanTree(CT) has been proposed, which is based on Apriori and FPGrowth algorithms. This combines both the candidate generation and tree formation techniques. The results shows the proposed algorithm's performance is better than the both existing algorithms.

The rest of the paper is organized as follows: section two briefly reviews the Apriori and FPGrowth algorithms and their optimized algorithms and describes the horizontal and vertical format of the databases. In section 3 the proposed algorithm is explained briefly and in section 4 the experimental results are shown with their interpretations and in section 5 conclusion and future work have been discussed.

2. Background

Apriori is a classic algorithm[1] was devised by Agrawal et.al in 1994 which uses join and prune technique. FPGrowth [2] is a tree based algorithm without candidate generation method. Header table and tree formation is the technique used in this algorithm. In the later phase Eclat[3] algorithm was proposed which uses the vertical database format for mining frequent patterns. Optimizations were made then on these algorithms to get the best out of them. Most of the then devised algorithms were based on these three algorithms.Xiang Cheng et al. [4] proposed an algorithm named as DP-Apriori which used transaction splitting. A support estimation technique was used in DP-Apriori which prevented information loss by transaction splitting. MSPM algorithm for patternsinmultiplebiologicalsequences was devised by Ling Chen et al.[5]. This algorithm mines frequent patterns without candidate generation. Pattern extending approach based on prefix tree was used in this approach.

Tree based algorithms are based on nodes and header tables. There are different types of data structures used for optimization of tree based algorithms. Some of them are N-list by TuongLe,etal.[6], nodelists and nodeset by Zhi-Hong Deng et al.[7].

Candidate generation and pruning is the base of Apriori algorithm. More number of candidates are generated and then they are pruned according to the given support count. There exists excessive I/O operations and repeated scanning of databases. This leads to more time and space complexity. This is the main drawback of Apriori. In FPGrowth,the recursive generation of conditional pattern bases and sub conditional pattern trees which are the main weaknessin FPGrowth. The proposed algorithm overcomes some of the limitations of both the algorithms. Both Apriori and FPgrowth uses horizontal format of the database, whereas one of the most basic algorithms Eclat uses vertical format of the database. As the proposed algorithm is based on Apriori and FPGrowth, it also mines using horizontal format of the database.

2.1Horizontal Database Layout

In horizontal data format, the data contains transaction id (tid) followed by the list of items. For example, in market basket analysis, the data formatis tid, that istransaction ID followed by the list of items purchased by the customer.

Tid/item	I1	I2	I3	I4	I5
100	1	1	1	0	0
200	1	0	1	0	0
300	0	1	0	1	1
400	1	0	0	1	1

Fig. 1 Horizontal layout of Dataset DB.

2.2 *Vertical Database layout*

Item/tid	100	200	300	400	500
I1	1	1	1	0	0
I2	1	0	1	0	0
I3	0	1	0	1	1
I4	1	0	0	1	1

Fig. 2 Vertical layout of Dataset DB.

Following are the definitions of terms which are commonly used in the paper.

- Set: Collection of elements.
- Pattern or Itemset: A set of items.
- Support: The number of occurrences of an itemset in a dataset.
- Min-support: The minimum frequency that an itemset should have to be frequent.
- Frequent pattern: An itemset whose frequency is at least min-support.
- K-itemset: an itemset containing k items
- Candidate: Any itemset that might be a frequent pattern

3. CanTree(CT)Algorithm

In this section, a new algorithm based on Apriori and the FP-Tree structure is proposed , which is called CanTree. This algorithm includes two steps. First, the data set is scanned one time to find out the frequent 1 itemsets. Function apriori-gen generates the candidate itemsets as Apriori, It generates Ck+1 from Lk in the following two step process:

Join step: By finding the union of the two frequent itemsets of size k, generate LK+1, the initial candidates of frequent itemsets of size k + 1, Mk and Nk that have the first k−1 elements in common.

LK+1 = Mk∪Nk= {iteml, . . . ,itemk−1,
itemk, itemk}
Mk= {iteml,item2, . . . , itemk−1, itemk}
Nk= {iteml,item2, . . . , itemk−1, itemk }
where, iteml<item2 <· · · <itemk<itemk.

Prune step: Check if all the itemsets of size k in Lk+1 are frequent and generate Ck+1 by removing those that do not pass this requirement from Lk+1. According to Apriori property, any subset of size k of Ck+1 that is not frequent will also be infrequent and cannot be the subset of a frequent itemset of size k+1.

Procedure treeBuild is used to compute the count of each candidate in the given candidate itemset by building the FP tree. The method constructs a FP-tree as same as FP-growth, and the next step is for each item in the header table, it finds all branches that include the item and returns the support of the candidate itemset.

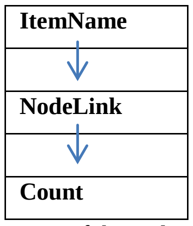

Fig.3aData structure of the node of header table

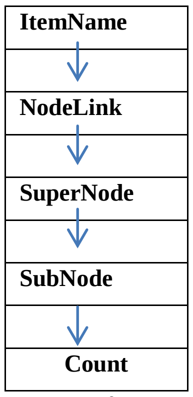

Fig.3bData structure of the node of FP-tree

Fig.3a shows the data structure of the node of header table. Its NodeLink points to the starting node in FPtree which has the same name with it. Fig.3b shows the data structure of the node of FP-tree. Its NodeLink pointsto the next node in FP-tree which has the same name with it.

Then the support of the candidate itemset is compared with the minimum support, and finally the resultant output is achieved.

The detailed CanTree algorithm is as follows.

```
Procedure :CanTree(CT)
Input :dataSet D, minimum support minsup.
Output : frequent item sets L
L1= frequent 1 item sets
for(k=2; Lk -1≠ф ; k++)
{
Ck= Apriori_Gen(Lk-1, minsup);
For each candidate c ∈ D
{
 Sup= treeBuild(c);
If(sup >minsup)
Lk = Lk U c;
 }
 }
return L = {L1UL2 U L3 U………U Ln};
```

```
Procedure : treeBuild
Input      : candidate item set c
Output     : the support of candidate item set c
Sort the items of c by decreasing order of header table;
Find the node r in the header table which has the same name with the first item of c;
s = r.NodeLink;
count = 0;
while s ≠ф
{
If the items of the itemset c except last item all appear in the prefix path of s
Count + = s.count ;
s = s. NodeLink;
 }
return count/ totalrecord ;
```

4. Experimental Results

The following are the results of the proposed CanTree(CT) algorithm in various datasets. The implementation is done using Netbeans IDE and coded using java programming language. Three realtime transactional datasets from FIMI(Frequent Itemset Mining Implementations Repository) were used for the experiments. These are also available in the machine learning UCI repository. They are Mushroom, Chess, Accident and Pumsb. Processing time is measured in milliseconds and space used is measured in megabytes.

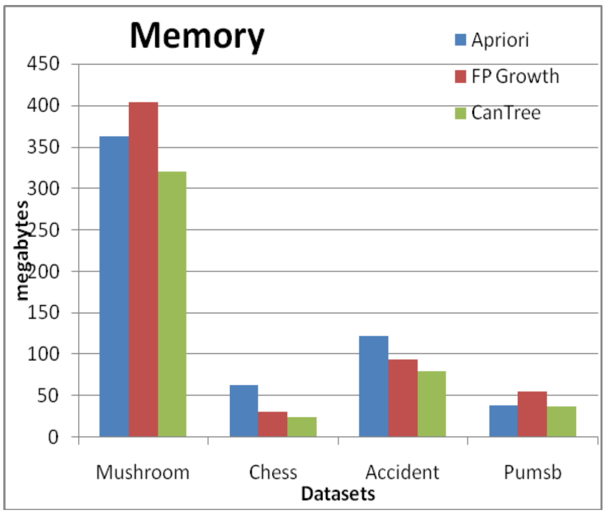

Fig.4 **Memory space consumption of Algorithms**

The Fig.4 shows the results of memory consumed by the algorithms in various datasets. The datasets are of varied sizes and number of transactions and frequent itemsets are different. It is observed that Apriori algorithm consumed more space than the other two. FPGrowth algorithm took less space than Apriori and the proposed CanTree utilized space lesser than both the algorithms in all the four datasets.

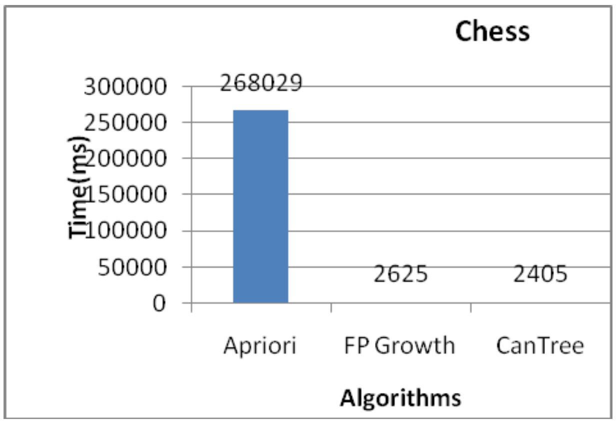

Fig.5Time consumption ofchess dataset

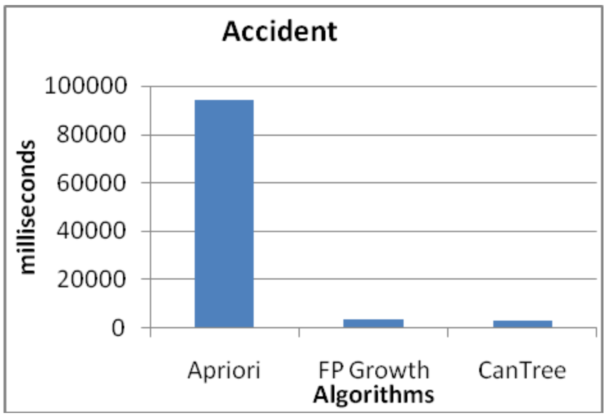

Fig.6Time consumption ofAccident dataset

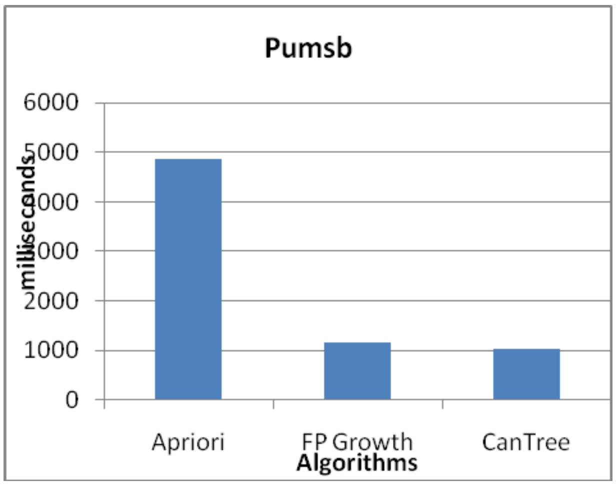

Fig.7Time consumption ofpumsb dataset

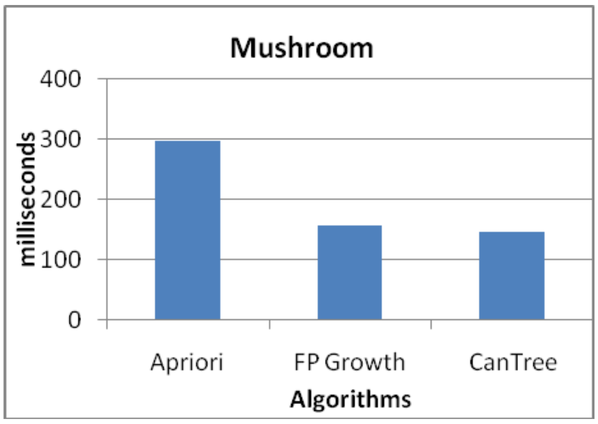

Fig.8Time consumption of mushroom dataset

Fig.5,6,7,8shows the Time taken to mine frequent itemsets by the algorithms in various datasets. It is observed that Apriori took more time to process than the other two in all the datasets. FPGrowth algorithm used less time than Apriori and little bit more than the proposed CanTree algorithm. The proposed algorithm processed in lesser time than the existing Apriori and FPGrowth algorithms.

5. Conclusion and Future Work

The experiments were conducted using different support thresholds. The datasets were realtime datasets from UCI(UC Irwin machine Learning Repository) and FIMI(Frequent ItemsetMining Dataset Repository). The proposed algorithm scans the database twice and the mining process is done. It is observed that CanTree(CT) performs better than the existing Apriori and FPGrowth algorithms both in reducing processing time and space used. Still it needs to generate candidate itemsets. In future the algorithm can be enhanced by taking efforts to further reduce the amount of memory space and processing time. The proposed algorithm still needs to generate candidates for processing that can be avoided in the algorithm's enhancements, and implement the same in data streams.

References

1. M.S. Chen, J. Han, P.S. Yu, "Data mining: an overview from a database perspective", *IEEE Transactions on Knowledge and Data Engineering*, 1996, 8, pp. 866-883.
2. J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publisher, San Francisco, CA, USA, 2001.
3. M. J. Zaki, S. Parthasarathy, M.Ogihara, W. Li et al., "New algorithms for fast discovery of association rules," in KDD, vol. 97, 1997, pp. 283–286.
4. Xiang Cheng, Sen Su, Shengzhi Xu, Zhengyi Li, ―DP-Apriori: A differentially private frequent itemset mining algorithm based on transaction splitting‖ Computers & Security, Volume 50, Pages 74-90, May 2015.
5. Ling Chen, Wei Liu, ―Frequent patterns mining in multiple biological sequences‖, Elsevier, Computers in Biology and Medicine, Volume 43, Issue 10, Pages 1444-1452, 1 October 2013.
6. Tuong Le, Bay Vo, ―An N-list-based algorithm for mining frequent closed patterns‖, Expert Systems with Applications, Volume 42, Issue 19, Pages 6648-6657, 1 November 2015.
7. Zhi-Hong Deng, Sheng-Long Lv, ―Fast mining frequent itemsets using Nodesets‖, Expert Systems with Applications, Volume 41, Issue 10, Pages 4505-4512, August 2014
8. Manjit Kaur, Urvashi Garg, Sarbjit Kaur,"Advanced eclat algorithm for frequent itemsets generation", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 9 (2015) pp. 23263-23279 © Research IndiaPublications www.ripublication.com
9. Qihua Lan, Defu Zhang, Bo Wu,"A New Algorithm For Frequent Itemsets Mining Based On Apriori And FP-Tree",978-0-7695-3571-5/09 $25.00 © 2009IEEE,DOI 10.1109/GCIS.2009.387
10. Neha Dwivedi, Srinivasa Rao Satti,"Vertical-format Based Frequent Pattern Mining - A Hybrid Approach", Journal of Intelligent Computing Volume 6 Number 4 December 2015
11. RamahSivakumar, J.G.R.Sathiaseelan,"A Performance based Empirical Study of the Frequent Itemset Mining Algorithms",International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI-2017), 978-1-5386-0814-2/17/$31.00 Â©2017 IEEE