An Analytical Study of Various Frequent Itemset Mining Algorithms

Patel Tushar S.[1], Panchal Mayur[2], Ladumor Dhara[2], Kapadiya Jahnvi[2], Desai Piyusha[2], Prajapati Ashish[3] and Prajapati Reecha[4]
[1]Mehsana, Gujarat, INDIA
[2]LDRP Institute of Technology and Research, Gandhinagar, Gujarat, INDIA
[3]Alfa College of Engineering and Technology, Khatraj, Kalol, Gujarat, INDIA
[4]Parul Institute of Engg. and Technology, Limda, Gujarat, INDIA

Available online at: www.isca.in
Received 13th January 2013, revised 21st January 2013, accepted 30th January 2013

Abstract

Frequent pattern mining has been a focused theme in data mining research for over a decade. The research and tremendous progress has been made, ranging from efficient and scalable algorithms for frequent itemset mining in transaction databases to numerous research frontiers. The time required for generating frequent itemsets plays an important role. In this paper, study includes depth analysis of algorithms and discusses some problems of generating frequent itemsets from the algorithm. We have explored the unifying feature among the internal working of various mining algorithms. The comparative study of algorithms includes aspects like different support values.

Keywords: Frequent pattern mining, association rules, itemsets, data mining.

Introduction

In recent years the size of database has increased rapidly. This has led to a growing interest in the development of tools capable in the automatic extraction of knowledge from data. The term data mining or knowledge discovery in database has been adopted for a field of research dealing with the automatic discovery of implicit information or knowledge within the databases. The implicit information within databases, mainly the interesting association relationships among sets of objects that lead to association rules may disclose useful patterns for decision support, financial forecast, marketing policies, even medical diagnosis and many other applications.

Frequent itemsets are appearing in a data set with frequency no less than a user-specified threshold. For example, a set of items, such as milk and cereal that appear frequently together in a transaction data set is a frequent itemset.

The problem of mining frequent itemsets arose first as a subproblem of mining association rules[1]. To analyze the huge amount of data thereby exploiting the consumer behavior and make the correct decision leading to competitive edge over rivals[2]. Frequent itemsets play an essential role in many data mining tasks that try to find interesting patterns from databases such as association rules, correlations, sequences, classifiers, clusters and many more of which the mining of association rules is one of the most popular problems. Also Sequential association rule mining is one of the possible methods to analysis of data used by frequent itemsets[3].

Frequent pattern mining was first proposed by Agrawal et al. (1993) for market basket analysis in the form of association rule mining. It analyses customer buying habits by finding associations between the different items that customers place in their “shopping baskets”. For instance, if customers are buying milk, how likely are they going to also buy cereal (and what kind of cereal) on the same trip to the supermarket? Such information can lead to increased sales by helping retailers do selective marketing and arrange their shelf space. Association rules describe how often items are purchased together. For example, an association rules “milk, cereal (80%)” states that four out of five customers that bought milk also bought cereal[1].

Since the first proposal of this new data mining task and its associated efficient mining algorithms, there have been hundreds of follow-up research publications, on various kinds of extensions and applications, ranging from scalable data mining methodologies, to handling a wide diversity of data types, various extended mining tasks, and a variety of new applications.

In this paper, study includes depth analysis of algorithms and discusses some problems of generating frequent itemsets from the algorithm. We have explored the unifying feature among the internal working of various mining algorithms.

Frequent itemset mining problem: Studies of Frequent Itemset (or pattern) Mining is acknowledged in the data mining field because of its broad applications in mining association rules, correlations, and graph pattern constraint based on frequent patterns, sequential patterns, and many other data mining tasks. Efficient algorithms for mining frequent itemsets are crucial for mining association rules as well as for many other data mining tasks. The major challenge found in frequent pattern mining is a large number of result patterns. As the minimum threshold becomes lower, an exponentially large number of itemsets are generated. Therefore, pruning unimportant patterns can be done effectively in mining process and that becomes one of the main topics in frequent pattern mining. Consequently, the main aim is to optimize the process of finding patterns which should be efficient, scalable and can detect the important patterns which can be used in various ways[4].

Related work: Horizontal layout based techniques: Apriori algorithm: Apriori is used to find all frequent itemsets in a given database DB. The key idea of Apriori algorithm is to make multiple passes over the database. It employs an iterative approach known as a breadth-first search (level-wise search) through the search space, where k-itemsets are used to explore (k+1)-itemsets. The working of Apriori algorithm is fairly depends upon the Apriori property which states that" All nonempty subsets of a frequent itemsets must be frequent"[1] (table 1).

Table–1
Apriori algorithm parameters

Storage Structure	**Array based**
Technique	Use Apriori property and join and prune menthod
Memory utilization	Due to large amount of candidate are produced so require large memory space
Databases	Suitable for sparse datasets as well as dense datasets
Time	Execution time is more as time wasted in producing candidates at every time

Direct Hashing and Pruning (DHP): A DHP technique was proposed to reduce the number of candidates in the early passes C_k for $k > 1$ and thus the size of database[5]. In this method, support is counted by mapping the items from the candidate list into the buckets which is divided according to support known as Hash table structure. As the new itemset is encountered if item exist earlier then increase the bucket count else insert into new bucket. Thus in the end the bucket whose support count is less the minimum support is removed from the candidate set (table 2).

Table–2
Direct Hashing and Pruning algorithm parameters

Storage Structure	**Array based**
Technique	Use hashing technique for fining frequent itemsets
Memory utilization	Require less space at earlier passes but more in later stages
Databases	Suitable for medium databases
Time	Execution time is small for small databases

Partitioning algorithm: Partitioning algorithm is based to find the frequent elements on the basis partitioning of database in n parts[6]. It overcomes the memory problem for large database which do not fit into main memory because small parts of database easily fit into main memory (table 3).

Table–3
Partitioning algorithm parameters

Storage Structure	**Array based**
Technique	Partition the database for finding local frequent item first
Memory utilization	Each partition is easily occupy in main memory
Databases	Suitable for large databases
Time	Execution time is more because of finding locally frequent then globally frequent

Dynamic Itemset Counting (DIC): This algorithm also used to reduce the number of database scan[7]. It is based upon the downward disclosure property in which adds the candidate itemsets at different point of time during the scan. In this dynamic blocks are formed from the database marked by start points and unlike the previous techniques of Apriori it dynamically changes the sets of candidates during the database scan. Unlike the Apriori it cannot start the next level scan at the end of first level scan, it start the scan by starting label attached to each dynamic partition of candidate sets (table 4).

Table–4
Dynamic Itemset Counting algorithm parameters

Storage Structure	**Array based**
Technique	Based upon dynamic insertion of candidate items
Memory utilization	Require different amount of memory at different point of time
Databases	Suitable for medium and low databases
Time	Execution time is small because dynamic itemset are added according to situation

Table–5
Sampling algorithm parameters

Storage Structure	**Array based**
Technique	Pick any random sample for checking frequency of whole database at lower threshold support
Memory utilization	Very less amount of memory is needed
Databases	Suitable for any kind of dataset but mostly not give accurate results
Time	Execution time is very much small

Sampling algorithm: This algorithm is used to overcome the limitation of I/O overhead by not considering the whole database for checking the frequency[8]. It is just based in the idea to pick a random sample of itemset R from the database instead of whole database D. The sample is picked in such a way that whole sample is accommodated in the main memory (table 5).

Vertical layout based technique: Eclat algorithm: Eclat algorithm is basically a depth-first search algorithm using set intersection[9]. It uses a vertical database layout i.e. instead of explicitly listing all transactions; each item is stored together with its cover (also called tidlist) and uses the intersection based approach to compute the support of an itemset. In this way, the support of an itemset X can be easily computed by simply intersecting the covers of any two subsets Y, Z ⊆ X, such that Y U Z = X. It states that, when the database is stored in the vertical layout, the support of a set can be counted much easier by simply intersecting the covers of two of its subsets that together give the set itself (table 6).

Table–6
Eclat algorithm parameters

Storage Structure	**Array based**
Technique	Use intersection of transaction ids list for generating candidate itemsets
Memory utilization	Require less amount of memory compare to apriori if itemsets are small in number
Databases	Suitable for medium and dense datasets but not suitable for small datasets
Time	Execution time is small then apriori algorithm

Projected database based techniques

FP-Growth algorithm: The most popular frequent itemset mining called the FP-Growth algorithm[10]. The problem of Apriori algorithm was dealt with, by introducing a novel, compact data structure, called frequent pattern tree, or FP-tree then based on this structure an FP-tree-based pattern fragment growth method was developed. Essentially, all transactions are stored in a tree data structure (table 7).

Table–7
FP-Growth algorithm parameters

Storage Structure	**Tree based**
Technique	It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy the minimum support
Memory utilization	Due to compact structure and no candidates generation require less memory
Databases	Suitable for large and medium datasets
Time	Execution time is large due to complex compact data structure

H-mine algorithm: H-mine algorithm is the improvement over FP-tree algorithm as in H-mine projected database is created using in-memory pointers[11]. H-mine uses an H-struct new data structure for mining purpose known as hyperlinked structure. It is used upon the dynamic adjustment of pointers which helps to maintain the processed projected tree in main memory therefore H-mine proposed for frequent pattern data mining for data sets that can fit into main memory (table 8).

Table–8
H-mine algorithm parameters

Storage Structure	**Tree based**
Technique	It uses the hyperlink pointers to store the partitioned projected database in main memory
Memory utilization	Memory is utilized according to needs and partitions of projected database
Databases	Suitable for sparse and dense datasets
Time	Execution time is large then FP-tree and others because of partition the database

Results and Discussion

Data set: The dataset was obtained from the UCI repository of machine learning databases[12]. The characteristics of adult dataset selected for the experiment (table 9).

Table–9
The characteristics of adult dataset

File name	**Number of Records**	**Number of Columns**
adult.D14.N48842.C2.num	48842	14

Result analysis: A detailed study to assess the performance of Apriori, Eclat and FP-Growth algorithms. The performance metrics is the total execution time taken and the number of frequent itemsets generated using an adult datasets. For this comparison also same dataset were selected as for the experiments with minimum support ranging from 30% to 70%.

Table–10
Total execution time using Adult dataset

Support	**Total Execution Time in Seconds**				
	Apriori	**Eclat**	**FP-Growth**	**Relim**	**SaM**
30	9.85	0.54	0.56	0.49	0.47
40	6.72	0.49	0.5	0.44	0.44
50	4.51	0.45	0.49	0.42	0.41
60	2.69	0.44	0.48	0.4	0.4
70	1.7	0.4	0.42	0.39	0.37

The total execution time of Apriori, Eclat, FP-Growth, Relim and SaM algorithms with different support threshold using an adult data set. The execution time is decreased when the support threshold increased. The SaM algorithm is better than the

Apriori and near the Relim. The difference between execution time of these algorithms decreases with increasing a support threshold. The Apriori takes more time as that compared to other algorithms (table 10).

Conclusion

Finally, we want to discuss performance are realized and which approaches mobile model scenarios. Summary of the in-depth analysis of few algorithms is done which made a significant contribution to the search of improving the efficiency of frequent itemset mining algorithms. By analytical study of the classical frequent itemset mining algorithms like Apriori, DHP, Partitioning, Sampling, DIC, Eclat, FP-growth, H-mine and find out the strength and weaknesses of these algorithms were analyzed.

A comparison framework has developed to allow the flexible comparison of Apriori, Eclat and FP-growth algorithms. Using this framework this paper presented the comparative performance study of these algorithms such as, Apriori, Eclat and FP-growth.

The execution time is decreased when the support threshold increased. The Eclat algorithm is better than the Apriori and near the FP-Growth.

References

1. Agrawal R., Imielienski T., and Swami A., Mining Association Rules between Sets of Items in Large Databases, *Proc. Conf. on Management of Data*, 207–216 **(1993)**

2. Raorane A.A., Kulkarni R.V. and Jitkar B.D., Association Rule – Extracting Knowledge Using Market Basket Analysis, *Res. J. Recent Sci.,* **1(2)**, 19-27 **(2012)**

3. Shrivastava Neeraj and Lodhi Singh Swati, Overview of Non-redundant Association Rule Mining, *Res. J. Recent Sci.,* **1(2)**, 108-112 **(2012)**

4. Pramod S., and Vyas O.P., Survey on Frequent Item set Mining Algorithms, *In Proc. International Journal of Computer Applications (0975-8887)*, **1(15),** 86–91 **(2010)**

5. Park. J.S, Chen M.S., Yu P.S., An effective hash-based algorithm for mining association rules, *In Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD),* 175–186 **(1995)**

6. Savasere E. Omiecinski and Navathe S., An efficient algorithm for mining association rules in large databases, *In Proc. Int'l Conf. Very Large Data Bases (VLDB),* 432–443 **(1995)**

7. Brin. S., Motwani. R., Ullman. J.D. and Tsur. S., Dynamic itemset counting and implication rules for market basket analysis, *In Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD),* 255–264 **(1997)**

8. C Toivonen. H., Sampling large databases for association rules, *In Proc. Int'l Conf. Very Large Data Bases (VLDB),* 134–145 **(1996)**

9. Borgelt C., Efficient Implementations of Apriori and Eclat, *In Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations* **(2003)**

10. Han J., Pei H., and Yin. Y., Mining Frequent Patterns without Candidate Generation, *In Proc. Conf. on the Management of Data* **(2000)**

11. Pei. J, Han. J, Lu. H, Nishio. S. Tang. S. and Yang. D., H-mine: Hyper-structure mining of frequent patterns in large databases, *In Proc. Int'l Conf. Data Mining* **(2001)**

12. Blake C.L. and Merz. C.J., UCI Repository of Machine Learning Databases, *Dept. of Information and Computer Science, University of California at Irvine, CA, USA* **(1998)**