An Empirical Study of Various Frequent Item Set Mining Techniques

Utkarsh Kumar Shrivastava
Badwani

Abstract- **Frequent item set mining has been a heart favorite theme for data mining researchers for over a decade. A large amount of literature has been dedicated to this research and tremendous progress has been made, ranging from efficient and scalable algorithms for frequent item set mining in transaction databases to numerous research frontiers, such as sequential pattern mining, structured pattern mining, correlation mining, associative classification, and frequent pattern-based clustering, as well as their broad applications. In this paper, a literature review of various latest techniques for mining frequent items from a transaction data base are presented in critical manner.**

Index Terms- Data Mining, Frequent Pattern Mining, Support, Confidence, Apriori, DIC, Partitioning.

1. INTRODUCTION

Data mining [1] is the process of extracting hidden patterns from data. As more data is gathered, with the amount of data doubling every three years, data mining is becoming an increasingly important tool to transform this data into knowledge. It is commonly used in a wide range of applications, such as marketing, fraud detection and scientific discovery. Data mining can be applied to data sets of any size, and while it can be used to uncover hidden patterns, it cannot uncover patterns which are not already present in the data set.

The discovered knowledge [2][3] can be used in many ways in corresponding applications. For example, identifying the frequently appeared sets of items in a retail database can be used to improve the decision making of merchandise placement or sales promotion. Discovering patterns of customer browsing and purchasing (from either customer records or Web traversals) may assist the modeling of user behaviors for customer retention or personalized services. Given the desired databases, whether relational, transactional, spatial, temporal, or multimedia ones, we may obtain useful information after the knowledge discovery process if appropriate mining techniques are used

A typical process of knowledge discovery in databases is illustrated in Fig. 1.

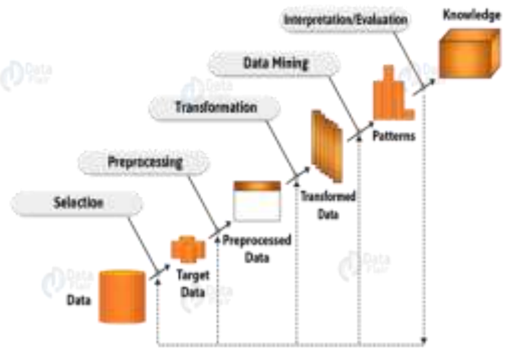

Fig. 1. The process of knowledge discovery in databases [1]

Knowledge discovery in databases is a complex process, which covers many interrelated steps. Key steps in the knowledge discovery process are:

- Data Cleaning: remove noise and inconsistent data.
- Data Integration: combine multiple data sources.
- Data Selection: select the parts of the data that are relevant for the problem.
- Data Transformation: transform the data into a suitable format.
- Data Mining: apply data mining algorithms and techniques.
- Pattern Evaluation: evaluate whether they found patterns meet the requirements
- Knowledge Presentation: present the mined knowledge to the user (e.g., Visualization).

The key step of association mining is frequent item set (pattern) mining which is to mine all item sets satisfying user specified minimum support [5]

Generally, a large number of these rules will be pruned after applying the support and confidence thresholds. Therefore most of the previous computations will be wasted. To overcome this problem and to improve the performance of the rule discovery algorithm, the association rule may be decomposed into two phases:

1. Generate the large item sets: the sets of items that have transaction support above a predetermined minimum threshold known as frequent Item sets.
2. Using the large item sets to generate the association rules for the database that has confidence above a predetermined minimum threshold.

The overall performance of mining association rules is depends primarily by the first step. The second step is easy. Once the large itemsets are identified the corresponding association rules can be derived in straightforward manner. The main consideration of the thesis is First step i.e. to find the extraction of frequent itemsets.

2. LITERATURE SURVEY

FP-Growth Algorithm

The most popular frequent itemset mining called the FP-Growth algorithm was introduced by [5]. The main aim of this algorithm was to remove the bottlenecks of the Apriori-Algorithm in generating and testing candidate set. 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. FP-growth uses a combination of the vertical and horizontal database layout to store the database in main memory. Instead of storing the cover for every item in the database, it stores the actual transactions from the database in a tree structure and every item has a linked list going through all transactions that contain that item. This new data structure is denoted by FP-tree (Frequent-Pattern tree) [4]. Essentially, all transactions are stored in a tree data structure.

Broglet's FP-Growth

Broglet implemented an efficient FP-Growth[1] algorithm using C Language. The FP-growth in his implementation preprocesses the transaction database according to [1] is as follows:

1. In an initial scan the frequencies of the items (support of single element item sets) are determined.
2. All infrequent items, that is, all items that appear in fewer transactions than a user-specified minimum number are discarded from the transactions, since, obviously, they can never be part of a frequent item set.
3. The items in each transaction are sorted, so that they are in descending order with respect to their frequency in the database.

Eclat

Eclat [11, 8, 3] algorithm is basically a depth-first search algorithm using set intersection. 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.

SaM Algorithm

The SaM (Split and Merge) algorithm established by [10] is a simplification of the already fairly simple RElim (Recursive Elimination) algorithm. While RElim represents a (conditional) database by storing one transaction list for each item (partially vertical representation), the split and merge algorithm employs only a single transaction list (purely horizontal representation), stored as an array.

This array is processed with a simple split and merge scheme, which computes a conditional database, processes this conditional database recursively, and finally eliminates the split item from the original (conditional) database.

Apriori Algorithm

The first algorithm for mining all frequent item sets and strong association rules was the AIS algorithm by [6]. Shortly after that, the algorithm was improved and renamed Apriori. Apriori algorithm is, the most classical and important algorithm for mining frequent item sets. Apriori is used to find all frequent item sets 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-item sets are used to explore (k+1)-item sets

Direct Hashing and Pruning (DHP):
It is absorbed that reducing the candidate items from the database is one of the important task for increasing the efficiency. Thus a DHP technique was proposed [7] to reduce the number of candidates in the early passes C for k 1 and thus the size of database. 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 item set 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.
In this way it reduce the generation of candidate sets in the earlier stages but as the level increase the size of bucket also increase thus difficult to manage hash table as well candidate set.

Partitioning Algorithm:
Partitioning algorithm [9] is based to find the frequent elements on the basis partitioning of database in n parts. 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. This algorithm divides into two passes,

1. In the first pass whole database is divided into n number of parts.
2. Each partitioned database is loaded into main memory one by one and local frequent elements are found.
3. Combine the all locally frequent elements and make it globally candidate set.
4. Find the globally frequent elements from this candidate set.

It should be noted that if the minimum support for transactions in whole database is min_sup then the minimum support for partitioned transactions is min-sup number of transaction in that partition.
A local frequent item set may or may not be frequent with respect to the entire database thus any item set which is potentially frequent must include in any one of the frequent partition.

Sampling Algorithm:
This algorithm [10] is used to overcome the limitation of I/O overhead by not considering the whole database for checking the frequency. It is just based in the idea to pick a random sample of item set 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. In this way we try to find the frequent elements for the sample only and there is chance to miss the global frequent elements in that sample therefore lower threshold support is used instead of actual minimum support to find the frequent elements local to sample. In the best case only one pass is needed to find all frequent elements if all the elements included in sample and if elements missed in sample then second pass are needed to find the item sets missed in first pass or in sample [12].
Thus this approach is beneficial if efficiency is more important than the accuracy because this approach gives the result in very less scan or time and overcome the limitation of memory consumption arises due to generation of large amount of datasets but results are not as much accurate.

Dynamic Item set Counting (DIC):
This algorithm [4] also used to reduce the number of database scan. It is based upon the downward disclosure property in which adds the candidate item sets 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.
In this way it reduce the database scan for finding the frequent item sets by just adding the new candidate at any point of time during the run time. But it generates the large number of candidates and computing their frequencies are the bottleneck of performance while the database scans only take a small part of runtime.
Assumption: The performance of all the above algorithms relies on an implicit assumption that the

database is homogenous and thus they will not generate too many extra candidates than Apriori algorithm does. For example, if all partitions in Partition algorithm are not homogenous and nearly completely different sets of local frequent itemsets are generated from them, the performance cannot be good.

3. CONCLUSION

Frequent pattern mining is a favourite topic of many researchers across the globe. Frequent item set mining has a wide range of real world applications. It affects decision making of many industries. This paper presented a comprehensive survey of latest techniques for mining frequent patterns from a standard data set. This review will be useful for future researchers of frequent pattern mining.

REFERENCES

[1] Tan P.-N., Steinbach M., and Kumar V. “Introduction to data mining, Addison Wesley Publishers”. 2006

[2] Nizar R.Mabrouken, C.I.Ezeife. Taxonomy of Sequential Pattern Mining Algorithm”. In Proc. in ACM Computing Surveys, Vol 43, No 1, Article 3, November 2010.

[3] A.M.Said, P.P.Dominic, A.B. Abdullah. “A Comparative Study of FP-Growth Variations”. In Proc. International Journal of Computer Science and Network Security, VOL.9 No.5 may 2009.

[4] Brin.S, Motwani. R, Ullman. J.D, and S. Tsur. “Dynamic itemset counting and implication rules for market basket analysis”. In Proc. ACM-SIGMOD Int’l Conf. Management of Data (SIGMOD), May 1997, pages 255–264.

[5] C. Borgelt. “An Implementation of the FP-growth Algorithm”. Proc. Workshop Open Software for Data Mining, 1–5.ACMPress, New York, NY, USA 2005.

[6] Ling Chen, Shan Zhang,Li Tu, “An Algorithm for Mining Frequent Items on Data Stream Using Fading Factor”.33rd Annual IEEE International Computer Software and Applications Conference.172-179,2009.

[7] Cai-xia Meng, An Efficient Algorithm for Mining Frequent Patterns over High Speed Data Streams. World Congress on Software Engineering,IEEE 2009, 319-323.

[8] Varun Kumar,Rajanish Dass.Proceedings of the 43rd Hawaii International Conference on System Sciences, 2010 IEEE, 978-0-7695-3869-3.

[9] Sonali Shukla, Sushil Kumar, Bhupendra Verma,A Linear Regression-Based Frequent Itemset Forecast Algorithm for Stream Data. International Conference on Methods and Models in Computer Science, 2009.

[10] ZHOU Jun, CHEN Ming, XIONG Huan A More Accurate Space Saving Algorithm for Finding the Frequent Items.IEEE-2010.

[11] Yong-gong Ren,Zhi-dong Hu,Jian Wang. An Algorithm for Predicting Frequent Patterns over Data Streams Based on Associated Matrix. Ninth Web Information Systems and Applications Conference, 2012. 95-98.

[12] Mahmood Deypir, Mohammad Hadi Sadreddini,A New Adaptive Algorithm for Frequent Pattern Mining over Data Streams, ICCKE,2011, 230-235 FLEXChip Signal Processor (MC68175/D), Motorola, 1996.