International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 4 Issue 4, April-2015, pp: (62-67), Impact Factor: 1.252, Available online at: www.erpublications.com

Algorithms for Frequent Pattern Mining - An Analysis

Ms. Dhara Patel[1], Prof. Ketan Sarvakar[2]
[1]ME (CSE Student), UVPCE, Kherva, Gujarat, India
[2]Asst. Professor, UVPCE, Kherva, Gujarat, India

Abstract: Data mining refers to extracting knowledge from large amounts of data. Frequent itemsets is one of the emerging task in data mining. Frequent itemsets mining is crucial and most expensive step in association rule mining. The problem of mining frequent itemsets arises in large transactional databases where there is need to find association rules among the transactional data for the growth of business. Several algorithms have been proposed and developed to increase efficiency of mining frequent itemsets. We present a analysis of various algorithms for mining frequent itemsets that work on horizontal, vertical, projected and hybrid layout datasets.

I. Introduction

Data mining is the process of discovering interesting knowledge from large amounts of data stored in database, data warehouse, or other information repositories. Popular research area of data mining was started by the tasks of frequent item set mining and association rule induction. The huge research efforts devoted to these tasks have led to a variety of sophisticated and efficient algorithms to find frequent item sets. Among the best-known approaches are Apriori, Eclat and FP-growth [1]. Frequent pattern mining is the process of searching recurring relationships in a given dataset. Frequent patterns are patterns (i.e. itemsets) that appears in a dataset frequently. A set of items, i.e. computer and antivirus that appears frequently together in a transaction dataset is a frequent itemsets. Frequent patterns mining like Frequent itemsets find frequent itemsets from the small database and/or large database, where the database are either transactional or relational. The frequent itemset mining is the process of finding out frequent itemsets from the DB. Frequent itemsets such like 1-frequent, 2-frequent, 3-frequent. . . . . k-frequent itemsets.

Data mining refers to discovering knowledge in huge amounts of data. The idea is to seek for something called knowledge, which means regularities, rule and structure hidden in the data. Association rule mining is one of the most important data mining problems. In transactional database it depicts the purchase patterns and reflects items that are frequently associated or purchased together. The mining of association rule include two sub problems (1) Finding all frequent itemsets that appear more often than a minimum support threshold, and (2) Generate association rules using these frequent itemsets.

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. Frequent item set mining is a data analysis method, which was originally developed for market basket analysis and which aims at finding regularities in the shopping behaviour of the customers of supermarkets, mail-order companies and online shops.[1].

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 computer, how likely are they going to also buy antivirus (and what kind of antivirus) 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.

Frequent itemsets mining is vital step in mining association rules. Two major approaches towards mining frequent itemsets are the candidate generate-and-test approach and the pattern growth approach. Many different algorithms has been proposed and developed to increase the efficiency of mining frequent itemsets. Based on the transactional database layout we have horizontal layout based algorithms, vertical layout based algorithms, projected layout based algorithms and hybrid algorithms. Rest of the paper formally introduces the problem and briefly reviews the frequent itemset mining algorithms.

In this paper, section 2 discuss the problem definition of the frequent itemset mining; section 3 discuss the various algorithms for frequent itemset mining; section 4 discuss analysis and discussion; Finally section 5 concludes the paper.

II. Problem Statement

Let I = {i_1, i_2, i_3…., i_n} be a set of items and 'n' is considered the dimensionality of the problem. A set of items is called itemset. An itemset that contains k items is called k-itemset. Let D be the task relevant database which consists of transactions where each transaction T is set of items such that T ∈ I. A transaction is a pair which contains unique identifier Tid and set of items [2] and transaction T is said to contain itemset X, which is called a pattern, i.e. X ∈ T ∈ I. An association rule is an implication of the form A⇒B where A ⊂ I, B ⊂ I and A ∩ B = Ø. The rule A⇒B holds support 's' in D where 's' is the percentage of transaction in D that contain AUB that is taken to be the probability P(AUB) and confidence 'c' where 'c' is the percentage of transaction in D containing A that also contain B which is the conditional probability P(B|A). An occurrence frequency of an itemset is the number of transactions that contains the itemset. This is also known as frequency, support count or just count.

An itemset X is said to be frequent if its support is greater than or equal to given minimum support threshold i.e. count(X) > minsup [3]. A transaction T is said to be maximal frequent if its pattern length is greater than or equal to all other existing transactional patterns and also count of occurrence (support) in database is greater than or equal to specified minimum support threshold. Transactional database D and minimum support threshold is given, therefore the problem is to find the complete set of frequent itemsets from transactional databases, so that relation between customers behaviour and various items can be found and can be used to increase the business.

III. Pattern Mining Algorithms

(1) Apriori algorithm :

R. Aggarwal [3] first proposed the Apriori algorithm. The name of the algorithm is based on the fact that the algorithm used a prior knowledge of frequent itemset properties. It's a well known algorithm and used in most commercial products. The use of support for pruning candidate itemsets is guided by the Apriori property which states that"All nonempty subsets of a frequent itemsets must be frequent" [3]. It is also described by antimonotonic property which says if the system cannot pass the minimum support test then all its supersets will fail to pass the test [2, 3]. Therefore if the one set is infrequent then all its supersets are also frequent and vice versa.

The algorithm is find with the transactional dataset D consisting of n transactions and the minimum support count. The algorithm initially scans the database and finds the support count of each item. Upon completion of this step, the set of all frequent 1-itemsets will be known. All infrequent items whose support count is less than the minimum support is removed obtaining L_1. Next, the algorithm will iteratively generate new candidate k-itemsets using the frequent (k-1)-itemsets found in the previous iteration through join and prune step as follows:

- In the join step the candidate set is generated by self joining L_{k-1} with itself and is denoted by C_k. This step generates new candidate k-itemsets.
- In prune step C_K is the superset of L_k so members of C_K may or may not be frequent but all K-1 frequent itemsets are included in C_K. This step eliminates some of the candidate k-itemsets using the Apriori property. A scan of the database to determine the count of each candidate in C_K would result in the determination of L_k (i.e., all candidates having a count greater than or equal to the minimum support count). C_K, however, can be huge, and so this could involve grave computation. To shrink the size of C_K, the Apriori property is used as follows. Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any (k-1)-subset of candidate k-itemset is not in L_{K-1} then the candidate cannot be frequent either and so can be removed from C_K. The join and prune is repeated until no new candidate set is generated.

(2) Eclat algorithm :

Eclat (Equivalence class transformation) uses set intersection and unlike Apriori, it works on vertical layout database. Each item use intersection based approach for finding the support. In this way, the support of an itemset P can be easily computed by simply intersecting of any two subsets [4].

First the horizontal data layout is converted into vertical data layout by scanning the data set once. The support count of an itemset is length of the TID_set of the itemset. Starting with k=1 the frequent k-itemsets can be used to construct the candidate (k+1)-itemset based on Apriori property. The computation is done by intersecting TID_sets of frequent k-itemsets to compute the TID_set of the corresponding (k+1)-itemset. The process is repeated until no more frequent itemsets or candidate itemsets can be found. The main advantage of this method is that database need not be scanned to

find the support count as the TID_set length itself is the support. When the dataset contain many dense and long patterns this technique can substantially reduce the total cost of vertical format mining of frequent itemsets. However, the TID_sets can be quite long taking substantial memory space as well as computational time for intersecting the long sets. The algorithms performance is not up to the mark for small database.

(3) FP-growth algorithm :

FP-algorithm constructs a Frequent Pattern tree by compressing the transactional database and houses the itemset association information. The algorithm adopts divide and conquer strategy. It divides the compressed database into special kind of projected database sets called conditional databases. Each division locally mines the frequent pattern examining only its associated data sets.

FP algorithm is as follows:

- The algorithm first finds the frequent 1-itemsets and their supports from the horizontal database layout and arranges them in decreasing order of their support count. When the support count is same then L ordering is done. The infrequent items that do not satisfy the minimum support criteria are removed.
- FP-Growth method then constructs the FP-tree starting from creation of "null" root node. L order processing of items in each transaction is done and branch nodes are created for each transaction following the L order and accordingly supports are updated. If the same nodes are traversed in another transaction then increment the support count of the node by 1. Each item points to its occurrence in the FP-tree via chain of node-link by maintaining the header table.

(4) H-mine algorithm :

H-mine [5] algorithm is used for datasets that can fit into main memory for mining frequent patterns. Hyper structure H-struct is designed for fast mining. It has polynomial space complexity therefore more space efficient then FP-Growth. Since its space overhead can be predicted it is much faster than memory-based Apriori and FP-growth.

The major flow of algorithm is as follows.

- First scan of dataset generates 1-frequent itemset. Using Apriori property all infrequent item are eliminated and remaining items creates a frequent- item projection. The items in the frequent-item projection are alphabetically arranged to obtain F-list.

The second scan of dataset constructs H-struct [5] as follows. The two field's item-id and hyper-link is used to store frequent item every time it occurs. A header table H with item-id, support count and a hyperlink is created. Once the frequent-item projections are loaded into memory, same first item in F-list and frequent item projections are linked together as a queue and the entries in header table act as the heads of the queues. The remaining mining is performed on the H-struct without referencing any information in the original database.

For the large databases, first partition the database then mine each partition in main memory using H-struct then consolidates into global frequent pattern [5]. If the database is dense then it integrates with FP-Growth dynamically by detecting the swapping condition and constructing the FP-tree. This working ensures that it is scalable for both large and medium size databases and for both sparse and dense datasets. The advantage of using in-memory pointers is that their projected database does not need any memory, the memory required only for the set of in-memory pointers.

(5) Frequent Item Graph (FIG) algorithm :

The FIG [6] algorithm is a novel method to find all the frequent itemsets quickly. It discovers all the frequent itemsets in only one database scan. The construction of the graphical structure is done in two phases, where the first phase requires a full I/O scan of the dataset and the second phase requires only a full scan of frequent 2-itemsets. The first initial scan of the database identifies the frequent 2-itemsets with a minimum support. The goal is to generate an ordered list of frequent 2-itemsets that would be used when building the graphical structure in the second phase.

The first phase starts by arranging the entire database in the alphabetical order. During the database scan the number of occurrences of frequent 2-itemsets is determined and infrequent 2-itemsets with the support less than the support threshold are weeded out. Then the frequent 2-itemsets are ordered in the alphabetical order. Phase 2 of constructing the graphical structure requires a complete scan of the ordered frequent 2-itemsets. The ordered frequent 2-itemsets are used in constructing the graphical structure.

Advantages:

- The quick mining process that does not use candidates.
- Mining the set of all frequent itemsets in the database by scanning the entire database only once.
- I/O costs spent in mining the set of all frequent itemsets will be reduced.

Disadvantage:

- The second phase requires a full scan of frequent 2-itemsets that would be used when building the graphical structure.

(6) Frequent Itemsets Algorithm for Similar Transactions (FIAST) :

The FIAST algorithm [7] for mining frequent itemsets based on similar transactions after deleting infrequent 1-itemsets. The algorithm applies the BitTable to aggregate transactions that have similar itemsets. The aggregation significantly reduces the number of transactions in the BitTable and the time using the divide-and-conquer method to reduce the bitwise AND operation for finding itemsets. Also, a depth-first search strategy is used to generate all frequent itemsets. And overcome the limitation of memory consumption arises due to generation of large amount of datasets but results are not as much accurate.

There are two steps in the FIAST algorithm as follows:

1. Creating the ItemTable and the BitTable: The ItemTable and the BitTable are constructed by a single scan of the database and aggregated the transactions that have similar itemsets.
2. Mining Frequent Itemsets: All frequent itemsets are mined by the FIAST algorithm.

The FIAST algorithm is a pattern growth approach without candidate generation. The algorithm shows the following profits:

- Minimize I/O: The FIAST algorithm minimizes I/O resources by scanning database only once for mining frequent itemsets.
- Save space: The FIAST algorithm saves space by storing items in the set of bits regardless of the size of the datasets.
- Reduce time: The FIAST algorithm reduces time by using bitwise AND operation for fast finding itemsets, and using a divide-and-conquer method to reduce finding tasks into smaller ones.
- Appropriate for similar transactions: The FIAST algorithm appropriates for mining frequent itemsets that have a number of similar transactions after deleting infrequent 1-itemsets, and works especially well for dense datasets.

(7) Indexed Limited Level Tree (ILLT) Algorithm :

ILLT [8] algorithm is to reduce the repeated database scans and to reduce the cost of computation required for generating frequent itemsets. Proposed ILLT algorithm is a three level tree structure algorithm composed of two steps. First step is construction of ILLTree structures. Second step is mining the tree structures for finding frequent itemsets. Frequent itemsets for any given support levels can be discovered quickly from the ILLTree structures. The mining process might take in some cases less than one full scan of the data structure for discovering frequent itemsets. First step of ILLT algorithm is construction of tree data structures. The levels of the tree are limited to three with an index node so it is named as Index Limited Level Tree (ILLT). The compact tree constructed in the first step is done by doing only one scan of given transactional database. From the resultant ILLTree it is easy to find frequent itemsets for different support levels. Scanning the database again is not needed at any stage. The tree structures store the contents of the transactions in their nodes.

Second step is the frequent itemset generation. Association rule mining starts by defining the support level σ. Based on the given support, the algorithm finds all frequent itemsets that occur more than σ. ILLTree scans this tree structures constructed in the first step to find the frequent itemsets. To find any frequent k-itemsets, the tree with the label k is searched. Given support level is compared with the occurrence number in the third level node. If the occurrence value matches the given support level, then the itemsets are frequent ones. Frequent itemsets can be achieved for any support levels at any time from this tree structure without scanning the database again thus reducing the computational cost and time.

IV. Analysis and Discussion

In our experiments we choose wine dataset with different properties, to prove the efficiency of the algorithm. In the wine dataset, 178 number of records and 14 number of columns. Table 1 shows the dataset from the UCI repository of machine learning databases.

Table 1: The characteristics of Dataset

Itemset	Number of Records	Number of Columns
Wine.data.txt	178	14

As a result, the performance of various algorithms shown in Table 2. The run time is the time to mine the frequent itemsets. The result is also shown in Figure 1.

Table 2: Total execution time using Wine dataset

Support	Total Execution Time in Seconds			
	H-mine	FIG	FIAST	ILLT
30	5.64	5.45	5.40	5.19
40	4.82	4.71	4.10	3.81
50	2.73	2.66	2.39	1.77

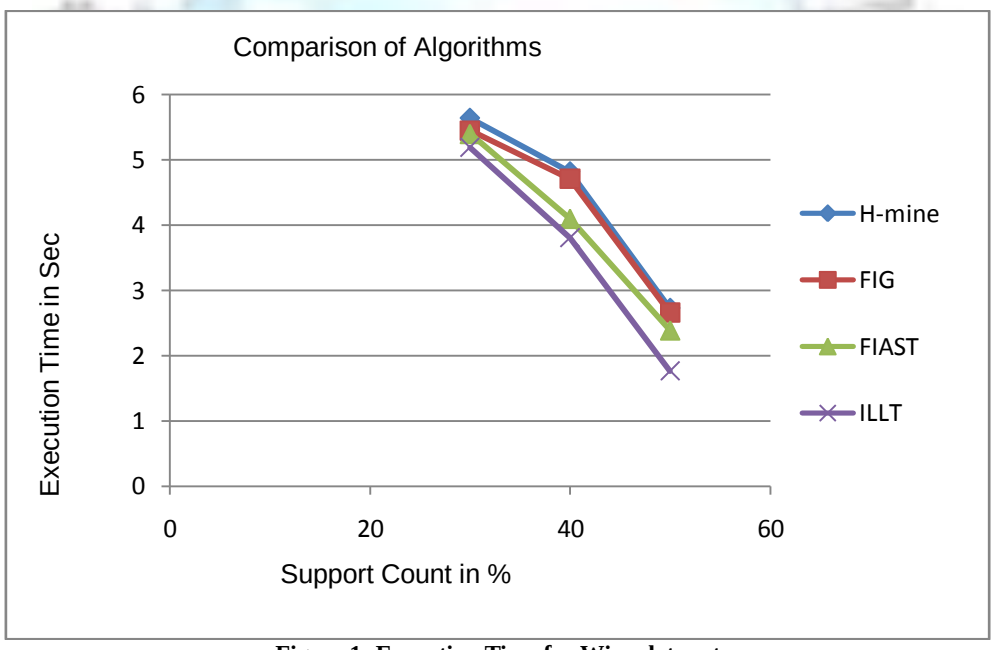

Figure 1: Execution Time for Wine data set

Conclusion

Frequent pattern mining problem has been studied extensively with alternative problem formulations, as well as new variants of existing algorithms. The efficiency with respect to run time of enumerating the complete set of frequent patterns has attracted most research. But we turn our attention from run time efficiency to address issues related to the interpretation and practicality of frequent itemset mining results. Although run time efficiency is also an important aspect of practicality, in real life application settings, coming up with a few high quality patterns is likely to be more valuable than simply enumerating a massive number of patterns in a very short time. Efficiency of mining task is no longer a bottleneck but there is still an urgent need for methods that derive compact, yet high quality results with good application properties.

References

[1]. Christian Borgelt, "Simple Algorithms for Frequent Item Set Mining."European Center for Soft Computing, Asturias, Spain.
[2]. Agrawal.R., Srikant.R. September 1994. Fast algorithms for mining association rules. In Proc. Int'l Conf. Very Large Data Bases (VLDB), pp.487–499.
[3]. Aggrawal.R; Imielinski.t; Swami.A. 1993. Mining Association Rules between Sets of Items in Large Databases. ACM SIGMOD Conference. Washington DC, USA.
[4]. C.Borgelt. 2003.Efficient Implementations of Apriori and Eclat. In Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations, CEUR Workshop Proceedings 90, Aachen, Germany.
[5]. Pei.J., Han.J., Lu.H., Nishio.S., Tang. S.,Yang. D. November 2001. H-mine: Hyper-structure mining of frequent patterns in large databases. In Proc. Int'l Conf. Data Mining (ICDM).
[6]. Kumar, A.V.S.; Wahidabanu, R.S.D.; , "A Frequent Item Graph Approach for Discovering Frequent Itemsets," Advanced Computer Theory and Engineering, 2008. ICACTE '08. International Conference on , vol., no., pp.952-956, 20-22 Dec. 2008
[7]. Duemong, F.; Preechaveerakul, L.; Vanichayobon, S.; , "FIAST: A Novel Algorithm for Mining Frequent Itemsets," Future Computer and Communication, 2009. ICFCC 2009. International Conference on , vol., no., pp.140-144, 3-5 April 2009
[8]. Venkateswari, S.; Suresh, R.M.; , "An efficient for discovery of frequent itemsets," Signal and Image Processing (ICSIP), 2010 International Conference on, vol., no., pp.531-533, 15-17 Dec. 2010.