Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 5, Issue. 1, January 2016, pg.271 – 275

Comparative Study on Algorithms of Frequent Itemset Mining

Ms. Ramya.S.Bhat[1], Mrs. A.Rafega Beham[2]

[1]M.Tech Scholar, Department of Information Science and Engineering, New Horizon College Of Engineering, Bangalore, India

[2]Asst.Professor, Department of Information Science and Engineering, New Horizon College Of Engineering, Bangalore, India

[1] bhatramyas@gmail.com, [2] rafeeka.rifa@gmail.com

Abstract- In this paper, we discuss about the frequent itemset mining which is a type of data mining that focuses on looking at sequences of events and which searches for recurring relationships in a given data set and about the algorithms used for the frequent itemset mining. The comparative study between the algorithms namely Apriori algorithm and FP growth algorithm which gives the details about the efficiency about the each algorithm and thereby helps to decide the most optimum and efficient algorithm. No doubt that Apriori algorithm successfully finds the frequent elements from the database, but FP-growth an efficient mining method of frequent patterns with respect to the large database finding the large pattern.

Keywords- Data Mining, Frequent Itemset, Apriori algorithm, FP Growth algorithm.

I.INTRODUCTION

Data mining, *the mining of hidden predictive information from large databases*, is the most powerful technology which helps companies focus on the most important information in their data warehouses. Data mining tools predict future trends and behaviours, allowing businesses to make proactive, knowledge-driven decisions. The automated, prospective analyses offered by data mining goes beyond the analyses of history events provided by tools typical of decision support systems. Data mining tools can answer business questions but which were too time consuming to resolve.

Data mining techniques are the result of a long process of research and product development. This evolution began when business data was first stored on computers, continued with improvements in data access, and more recently, generated technologies that allow users to navigate through their data in real time. Data mining is done in many patterns, one among them is the Frequent Itemset mining.

Frequent itemset mining is a type of data mining that focuses on looking at sequences of events.Frequent patterns are patterns (such as itemsets, subsequences, or substructures) that appear in a data set frequently. For example, a set of items, such as milk and egg, that appear frequently together in a transaction data set is a frequent itemset. A subsequence, such as buying first a Laptop, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (frequent) sequential pattern. A substructure can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a (frequent) structured pattern.

Frequent pattern mining searches for recurring relationships in a given data set. Frequent pattern mining is used for the discovery of interesting associations and correlations between itemsets in transactional and relational database. With large amount of data continuously being collected and stored, many organizations are becoming interested in mining such patterns from the large databases. The discovery of interesting correlation relationships among huge amounts of business transaction records can help in many business decision-making processes, such as catalog design, cross-marketing, and customer shopping behaviour analysis. A typical example of frequent itemset mining is market basket analysis. This process analyzes customer buying habits by recognizing associations between the different items that customers place in their "shopping baskets" .The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. The Boolean vectors can be analyzed for buying patterns that portray items that are most frequently associated or bought together. These patterns can be represented in the form of association rules. For example, the information that customers who purchase computers also tend to buy antivirus software which is represented in Association Rule as below:

computer ⇒ antivirus software [support = 3%,confidence = 50%]

Here, in this association rule support and confidence are two measures of rule interestingness. They respectively reflect the usefulness and certain type of discovered rules. A support of 3% for Association Rule means that 3% of all the transactions under analysis show that computer and antivirus software are purchased together. A confidence of 50% means that 50% of the customers who purchased a computer also bought the antivirus software. Some of the Algorithms used for analysing the frequent itemset pattern are Apriori algorithm and the F-P Growth Algorithm. In this paper we conduct a comparative study between the two algorithms and find the efficient among the two.

II.RELATED WORK

Apriori is an algorithm proposed by R.Agrawal and R.Srikantin 1994 for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties. Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore(k+1)-itemsets. First, the set of frequent1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted L1.Next,L1 is used to find L2, the set of frequent2-itemsets,which is used to find L3,and soon, until no more frequent k-itemsets can be found. The finding of each Lk requires one full scan of the database. The Apriori algorithm is illustrated below:

Algorithm: **Apriori**, Find frequent itemsets using an iterative level-wise approach based on candidate generation. Input:D, a database of transactions; min sup, the minimum support count threshold.

Output: L, frequent itemsets in D. Method:

(1) L1 = find frequent 1-itemsets(D);

(2) for (k = 2;Lk−1 6= φ;k++) {

(3) Ck = apriori gen(Lk−1);

(4) for each transaction t ∈ D { // scan D for counts

(5) Ct = subset(Ck, t); // get the subsets of t that are candidates

(6) for each candidate c ∈Ct

(7) c.count++;

(8) }

(9) Lk = {c ∈Ck|c.count ≥ min sup}

(10) }

(11) return L = ∪kLk;

procedure apriori gen(Lk−1:frequent (k−1)-itemsets)

(1) for each itemset l1 ∈ Lk−1

(2) for each itemset l2 ∈ Lk−1

(3) if (l1[1] = l2[1])∧(l1[2] = l2[2])∧...∧(l1[k−2] = l2[k−2])∧(l1[k−1] < l2[k−1]) then {

(4) c = l1 on l2; // join step: generate candidates

(5) if has infrequent subset(c, Lk−1) then

(6) delete c; // prune step: remove unfruitful candidate

(7) else add c toCk;

(8) }

(9) returnCk;

procedure has frequent subset(c: candidate k-itemset; Lk−1: frequent (k−1)-itemsets); // use prior knowledge

(1) for each (k−1)-subset s of c

(2) if s 6∈ Lk−1 then

(3) return TRUE;

(4) return FALSE;

To increase the efficiency of this algorithm, joining and pruning method is used.

The steps to mine the frequent elements are as follows:

• **Generate and test**: In this first find the 1-itemset frequent elements L1 by scanning the database and removing all those elements from C which cannot satisfy the minimum support criteria.

• **Join step**: To attain the next level elements Ck join the previous frequent elements by self join i.e. Lk-1*Lk-1 known as Cartesian product of Lk-1 . i.e. This step generates new candidate k-itemsets based on joining Lk-1 with itself which is found in the previous iteration. Let Ck denote candidate k-itemset and Lk be the frequent k-itemset.

• **Prune step**: 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 Ck would result in the determination of Lk (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to Lk). Step 2 and 3 is repeated until no new candidate set is generated. As we have seen, in many cases the Apriori candidate generate-and-test method gradually reduces the size of candidate sets, leading to good performance gain.

Limitations:

However, it can suffer from two nontrivial costs:

(1) It may need to generate a huge number of candidate sets. For example, if there are 104 frequent 1-itemsets, the Apriori algorithm will need to generate more than 107 candidate 2-itemsets. Moreover, to discover a frequent pattern of size 100, such as {a1,..., a100}, it has to generate at least $2100 - 1 \approx 1030$ candidates in total.

(2) Itmayneedtorepeatedlyscanthedatabaseandcheckalargesetofcandidatesbypattern matching. It is costly to go over each transaction in the database to determine the support of the candidate itemsets.

Another method that mines the complete set of frequent itemsets without candidate generation is called frequent-pattern growth, or simply FP-growth, which adopts a divide-and-conquer strategy. First, it compresses the database representing frequent items into a frequent-pattern tree, which retains the itemset association information. It then divides the compressed database into a set of conditional databases, each associated with one frequent item or pattern fragment and mines each such database separately. F-P Growth algorithm is illustrated below:

Algorithm: FP growth. Mine frequent itemsets using an FP-tree by pattern fragment growth.

Input: D, a transaction database;

min sup, the minimum support count threshold.

Output: The complete set of frequent patterns. Method:

1. The FP-tree is constructed in the following steps:

(a) Scan the transaction database D once. Collect F, the set of frequent items, and their support counts. Sort F in support count descending order as L, the list of frequent items.

(b) Create the root of an FP-tree, and label it as "null." For each transaction Trans in D do the following. Select and sort the frequent items in Trans according to the order of L. 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], T), which is performed as follows. If T has a child N such that N.item-name=p.item-name, then increment N's count by 1; else create a new node N, and let its count be 1, its parent link be linked to T, and its node-link to the nodes with the same item-name via the node-link structure. If P is nonempty, call insert tree(P, N) recursively.

2. The FP-tree is mined by calling FP growth(FP tree, null), which is implemented as follows. procedure FP growth(Tree, α)

(1) if Tree contains a single path P then

(2) for each combination (denoted as β) of the nodes in the path P

(3) generate pattern $\beta \cup \alpha$ with support count = minimum support count of nodes in β;

(4) else for each ai in the header of Tree {

(5) generate pattern $\beta = ai \cup \alpha$ with support count = ai.support count;

(6) construct β's conditional pattern base and then β's conditional FP tree Treeβ;

(7) if Treeβ 6= / 0 then

(8) call FP growth(Treeβ, β); }

III.CONCLUSION

In this paper, after the comparative study between the two algorithms it is easy to decide that F-P Growth algorithm is more efficient compared to an Apriori algorithm. It is no doubt that Apriori algorithm successfully finds the frequent elements from the database. But as the dimensionality of the database increase with the number of items then more search space is needed and I/O cost will increase. Number of database scan is increased thus candidate generation will increase, which in turn results in increase in computational cost. FP-

tree a novel data structure storing compressed, crucial information about frequent patterns is compact yet complete for frequent pattern mining. FP-growth an efficient mining method of frequent patterns in large database using a highly compact FP-tree, divide-and-conquer method. However, both Apriori and FP-Growth are aiming to find out complete set of patterns but, FP-Growth is more efficient than Apriori in concern to long patterns.

REFERENCES

[1] Liwu, ZOU, Guangwei, REN, "The data mining algorithm analysis for personalized service," Fourth International Conference on Multimedia Information Networking and Security, 2012.

[2] Jun TAN, Yingyong BU and Bo YANG, "An Efficient Frequent Pattern Mining Algorithm", Sixth International Conference on Fuzzy Systems and Knowledge Discovery, 2009.

[3] Wei Zhang, Hongzhi Liao, Na Zhao, "Research on the FP Growth Algorithm about Association Rule Mining", International Seminar on Business and Information Management, 2008.

[4] S.P Latha, DR. N.Ramaraj. "Algorithm for Efficient Data Mining". In Proc. Int' Conf. on IEEE International Computational Intelligence and Multimedia Applications, 2007.

[5] Dongme Sun, Shaohua Teng, Wei Zhang, Haibin Zhu. "An Algorithm to Improve the Effectiveness of Apriori". In Proc. Int'l Conf. on 6th IEEE International Conf. on Cognitive Informatics (ICCI07), 2007.

[6] Daniel Hunyadi, "Performance comparison of Apriori and FP-Growth algorithms in generating association rules", Proceedings of the European Computing Conference, 2006.

[7] By Jiawei Han, Micheline Kamber, "Data mining Concepts and Techniques" Morgan Kaufmann Publishers, 2006.

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

[9] Han.J, Pei.J, and Yin. Y. "Mining frequent patterns without candidate generation". In Proc. ACM-SIGMOD International Conf. Management of Data (SIGMOD), 2000.

[10] R. Agrawal, Imielinski.t, Swami.A. "Mining Association Rules between Sets of Items in Large Databases". In Proc. International Conf. of the ACM SIGMOD Conference Washington DC, USA, 1993.

