Performance Analysis of Apriori and Partitioning Method in Frequent Itemset Generation

M.Subithra M.phil scholar,
Department of computer science
Alagappa university,karaikudi.

Dr.SS.Dhenakaran Profssor,
Department of computer science
Alagappa University,Karaikudi.

Abstract:

In Current decade, Large volume of information sharing in the world wide web. Particularly, Electronic commerce grows highly and stand on important place in the global market. Users buy all kind of products and walk around the market world through internet. For improving the business, Market analysis and consumer behavioral analysis is very important. Datamining is powerful techniques to dig the data for analysis purpose. Various algorithms achieve the optimal solution for analysis and researchers improve the existing algorithm and contribute novel methods for fine tuning the analysis process and solve the complex problems. Apriori is one of the most common techniques for finding the frequent itemset. This algorithm is used to gather the data for frequent data usage or data flow of the domain. The large amount of data split into different sets that the process is called partition algorithm. In this paper, the numerical dataset is applied in the apriori as well as partition algorithm and justify the performance of discovering the frequent itemset. For performance analysis, implement the both algorithm apriori and (PAFI) Partition Algorithm for mining Frequent Itemset. into hundred itemset data and deliver the result in term of time complexity

I.Introduction

Business sectors evaluate the market data for improving the sales. Datamining techniques help to analysis the large volume of data. Especially, The marketing industry use the mining techniques for analysis the sales information and customer behavior[8]. For market data analysis, historical data is very important for predict the future statement of business status. Massive amount of data hides the knowledge of information. The data are stored in to centralized database and acts as a large repository that is called data warehouse. The knowledge discovers from the data warehouse through datamining techniques that is leads to discover the kinds of knowledge and patterns. So many techniques avail to discover the knowledge and patterns. There are several datamining techniques such as Association, Classification, Clustering, Prediction, Sequential patterns and decision making; they are applying into dataset to dick the knowledge. Association is one the familiar technique to discover patterns and generate the association rules[4][5]. In the association mining, findin frequency itemset is important process for generating the association rules. Researchers deliver various algorithms and techniques[6][7] to retrieve frequent item set. Apriori is one of the well known and common algorithm for finding the frequent itemset. Bottom up operation proceeds the apriori algorithm,where frequent subsets extended one item at a time. It is design to operate on database containing transaction. When applying the algorithm in large dataset. time complexity and memory space becomes high. PAFI overcome the limitation of Apriori algorithm. This is to do the partition the database into transaction in matrix format and find the frequent item set. The proposed works analyze the performance of Apriori and PAFI algorithm in term of time complexity. The analyse result shows the performance of PAFI and it is better than apriori

II.Apriori Algorithm

The main objective of Apriori algorithm is to find the frequent itemset through sets of transaction from the large volume of dataset which is in datawarehouse[1][3]. Mainly , minimum support value and confident value are important parameters for generating the frequest item set. The minimum support value decides the lower bound of the frequent itemset. The confidence value measures the firmness of the frequent item set. The confident value is used to generate the association rules. For frequent itemset generation, minimum support value is set as a constant value in iteration process. In iteration process, any subset of frequent itemset must be frequent itemset. The Apriori algorithm carryout a breadth-first search in the seek space by generating candidate k+1-itemsets from frequent k itemsets. The occurance of frequent items in each transaction counts the frequent itemset.

Pseudocode of Apriori algorithm:

Pass 1

1. Generate the candidate itemsets in C_1
2. Save the frequent itemsets in L_1

Pass *k*

1. Generate the candidate itemsets in C_k from the frequent itemsets in L_{k-1}
    1. Join L_{k-1} p with L_{k-1}q, as follows:
       insert into C_k
       select $p.\text{item}_1, p.\text{item}_2, \ldots, p.\text{item}_{k-1}, q.\text{item}_{k-1}$
       from L_{k-1} p, L_{k-1}q
       where $p.\text{item}_1 = q.\text{item}_1, \ldots p.\text{item}_{k-2} = q.\text{item}_{k-2}, p.\text{item}_{k-1} < q.\text{item}_{k-1}$
    2. Generate all (k-1)-subsets from the candidate itemsets in C_k
    3. Prune all candidate itemsets from C_k where some (k-1)-subset of the candidate itemset is not in the frequent itemset L_{k-1}
2. Scan the transaction database to determine the support for each candidate itemset in C_k
3. Save the frequent itemsets in L_k

Data base table:

S.no	Items
1	clinipuls
2	pantene
3	vatika
4	clear
5	Dove
6	Sunsilk
7	hamalaya
8	Tide
9	bizzpower
10	wheelleman
11	surfexcelmatric
12	ultrapower
13	Gain
14	Ariel
15	Vico
16	pepsodent
17	Advancewhite
18	Closeup
19	Crestfloran
20	Ultrabrite
21	oral-B
22	fair&lovelly
23	Garnier
24	Humalaya
25	headensholders

III.Partition Algorithm

The partitioning algorithm divides the transactional dataset D into n non-overlapping partitions, D1, D2...Dn. The algorithm reduce the number of dataset process to two. During the first process, the algorithm finds all item sets in each partition. Those local frequent item sets are collected into the global candidate item sets. During the second scan, these global item sets are counted to determine if they are large across the entire dataset. The partitioning algorithm improves the performance of finding frequent item sets and also provide several advantages. Small partitions might be fit into main memory than large one. Because the size of each partition is small, the algorithm might reduce the size of candidate item sets. In addition, the algorithm require only two scans on the dataset. However, the partition algorithm reduce the size of cluster. In this method, each partitioned data is called as cluster, there is no clustering algorithm apply into the dataset. There are Many methods are available for partition the data. The choice of a particular method will depend on the type of output desired, the known performance of method with particular types of data, the hardware and software facilities available and the size of the dataset[11][10]. In general, clustering methods may be divided into two categories based on the cluster structure which they produce. The non-hierarchical methods divide a dataset of N objects into M clusters, with or without overlap. In PAFI non overlapping algorithm uses for creating partitions. These methods are sometimes divided into partitioning methods, in which the classes are mutually exclusive, and the less common clumping method, in which overlap is allowed. Each object is a member of the cluster with which it is most similar; however the threshold of similarity has to be defined. The hierarchical methods produce a set of nested clusters in which each pair of objects or clusters is progressively nested in a larger cluster until only one cluster remains. The hierarchical methods can be further divided into agglomerative or divisive methods. In agglomerative methods, the hierarchy is build up in a series of N-1 agglomerations, or Fusion, of pairs of objects, beginning with the un-clustered dataset. The less common divisive methods begin with all objects in a single cluster and at each of N-1 steps divides some clusters into two smaller clusters, until each object resides in its own cluster. The partitioning methods generally result in a set of M clusters, each object belonging to one cluster. Each cluster may be represented by a centroid or a cluster representative; this is some sort of summary description of all the objects contained in a cluster. The precise form of this description will depend on the type of the object which is being clustered. In case where real-valued data is available[9], the arithmetic mean of the attribute vectors for all objects within a cluster provides an appropriate representative; alternative types of centroid may be required in other cases, e.g., a cluster of documents can be represented by a list of those keywords that occur in some minimum number of documents within a cluster. If the number of the clusters is large, the centroids can be further clustered to produces hierarchy within a dataset.

Pseudocode of partition algorithm:

Input:
D=dataset
K= the number of centers
C=initial centroids
Output: Set of k representing a good partitioning of D database and produce the frequent pattern.
1: Select the initial data set
2: for all data point di € D do
3: assignedCenter =di.center
4: assignedPartiton= di.partition
5: for all center ci €C do
6: apply on the Item set Ii €I
7: Xn ,n=1,2,3.....N
8: A partition P of an interval I is a set of M blocks,
9: P(I)={Bm ,m € M },M={1,2...M}
10: where the blocks are sets of data cells defined by index sets Nm:: Bm ={Xn,n € Nm}
11: enter key (n) for partition.
12: Find count (item set)
13: if(count (item set) isEven())
{If key (even)
{Partition in n/2 sets
}Else
{Partition in n+/2 sets
}14: if(count(itemset) isOdd())
{If key(even
)
{Partition in n+/2 sets
}Else{
Partition in n /2 sets}
15: find frequent pattern for the local partition until all
local partition finishes
16: Combine all local partition and find the global partition.
17: Finish

Database splited table:

S.no	Items
1	Clinipuls
2	Pantene
3	Vatika
4	Clear
5	Dove
6	Sunsilk
7	Hamalaya
8	Tide
9	bizzpower
10	wheelleman
11	Surfexcelmatric
12	ultrapower
13	Gain
14	Ariel
15	Vico
16	Pepsodent
17	Advancewhite
18	Closeup
19	Crestfloran
20	Ultrabrite
21	oral-B
22	fair&lovelly
23	Garnier
24	Humalaya
25	Headensholders

Table : Partition P1

1	clinipuls
2	pantene
3	vatika
4	clear
5	Dove
6	Sunsilk
7	hamalaya
8	Tide
9	bizzpower
10	wheelleman

Table Partition P1(1)

6	Sunsilk
7	hamalaya
8	Tide
9	bizzpower
10	wheelleman

Table Partition P2

11	Surfexcelmatric
12	ultrapower
13	Gain
14	Ariel
15	Vico
16	pepsodent
17	Advancewhite
18	Closeup
19	Crestfloran
20	Ultrabrite
21	oral-B
22	fair&lovelly
23	Garnier
24	Humalaya
25	Headensholders

Table P2(1)

16	pepsodent
17	Advancewhite
18	Closeup
19	Crestfloran
20	Ultrabrite
21	oral-B

Table Partition P1(2)

11	Surfexcelmatric
12	ultrapower
13	Gain
14	Ariel
15	Vico

Table Partition P2(2)

22	fair&lovelly
23	Garnier
24	Humalaya
25	Headensholders

Table Merged Partition P1(3)

7	hamalaya
8	Tide
12	ultrapower
13	Gain

Table Merged Partition 2(3)

23	Garnier
24	Humalaya
18	Closeup
19	Crestfloran

Table Final Result.

24	Humalaya
8	Tide
13	Gain
19	Crestfloran

P1,P2 are initial partitions which is split from original dataset, p1(1), p1(2) are partition table which is split number initial partition P1. Partition2(3) elements are incorporated from P1(1),P2(2). Final result merges all partitions and create overll table result. .

In this paper, the entire database is splitted into non overlapping partitions of various sizes, each partition represents as a cluster[2]. Cluster loads in the memory and computing large dataset and the Each cluster is considered one at a time by loading the first cluster into memory and calculating large itemsets and the ensuing support counts. Number of partitions fit in to memory space based on the side of partitions. The partition dataset reduce the consumption of memory space. Then the second cluster is considered as same process followed in the first cluster and the cumulative support count is calculated for the cumulative large itemsets. These steps are continued for the all set of clusters and finally we have the whole large itemsets and the corresponding cumulative support counts. This approach reduces the consumption of memory space since it considers only a small cluster at a time and hence it is scalable for any large size of the database. For discovering the large itemsets it is sufficient to go through the transactions into the partitioned itemset alone. There is no need to iterate the entire database again. Hence it decreases the redundant database inspection and increase the efficiency. Memory space management directs to improve the performance of formulating the frequent item set. So the time complexity is decrease when compare to apriori algorithm. So the performance of PAFI is efficient while handling the large amount of data than apriori

IV. Experimental Results

Apriori and PAFI are implemented in Java platform which build with Windows 7 operation system. Initially the dataset perform the preprocess that is called binary conversion. If the value is present in the itemset , set the binary value as 1. If the value is not present in the itemset, set the binary value as 0. Minimum support value is very important parameter for generating the frequent itemset. The number of frequent itemset generation varies from the minimum support value.

Table 1. Apriori algorithm with Min Support value 0.5

No of Itemset	Time complexity in mili seconds
25	383
50	1128
100	3323

Table 2. PAFI algorithm with Min Support value 0.5

No of Itemset	Time complexity in milli seconds
25	231
50	923
100	1761

On the comparsion of time complexity, the apriori algorithm consume more time of finding the frequent itemset than partition algorithm for finding frequent item set.

V. Conclusion

In association mining techniques, finding frequent itemset is basic operation for generating the association rule. Researchers have approached various methods to mine the frequent itemset. In this paper. Apriori algorithms and compared in terms of time complexity.The efficiency of PAFI is better than Apriori when compared to iteration process, time complexity and utilization of memory space.

Reference

[1] Agrawal R, Imielinski T, Swami A, "Mining association rules between sets of items in large databases". In: Proc. of the l993ACM on Management of Data, Washington, D.C, May 1993. 207-216.

[2] D.Kerana Hanirex, Dr.M.A.Dorai Rangaswamy:" Efficient algorithm for mining frequent item sets using clustering techniques." In International Journal on Computer Science and Engineering Vol. 3 No. 3 Mar 2011. 1028-1032.

[3]Margatet H. Dunham. Data Mining, Introductory and Advanced Topics: Upper Saddle River, New Jersey: Pearson Education Inc.,2003.

[4] Tong Qiang, Zhou Yuanchun, Wu Kaichao, Yan Baoping, " A quantitative association rules mining algorithm". Computer engineering. 2007, 33(10):34-35.

[5] Wael A. AlZoubi, Azuraliza Abu Bakar, Khairuddin Omar," Scalable and Efficient Method for Mining Association Rules", International Conference on Electrical Engineering and Informatics 2009.

[6] Wael Ahmad AlZoubi, Khairuddin Omar, zuraliza Abu Bakar "An Efficient Mining of Transactional Data Using Graph-based Technique",3rd Conference on Data Mining and Optimization (DMO) 2011, Selangor, Malay.

[7] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," in Proceedings of the 20th VLDB Conference, 1994, pp. 487−499.

[8] Arun K Pujari. Data Mining Techniques (Edition 5):Hyderabad, India: Universities Press (India) Private Limited, 2003.

[9] Margatet H. Dunham. Data Mining, Introductory and Advanced Topics: Upper Saddle River, New Jersey: Pearson Education Inc., 2003.

[10] Jiawei Han. Data Mining, concepts and Techniques: San Francisco, CA: Morgan Kaufmann Publishers.,2004.

[11] Akhilesh Tiwari, Rajendra K. Gupta, and Dev Prakash Agrawal "Cluster Based Partition Approach for Mining Frequent Itemsets" In Proceedings of the IJCSNS International Journal of computer Science and Network Security, VOL.9 No.6, June 2009.