ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, Vol 12(3), DOI: 10.17485/ijst/2019/v12i3/140766, January 2019 ISSN (Online) : 0974-5645

Determining Frequent Item Sets using Partitioning Technique for Large Transaction Database

K. Sampath Kini[1] and K. Karthik Pai[2]

[1]Department of Computer Science and Engineering, NMAMIT, Nitte, Karkala - 574110, Karnataka, India; sampath@nitte.edu.in

[2]Department of Information Science and Engineering, NMAMIT, Nitte, Karkala - 574110, Karnataka, India; karthikpai@nitte.edu.in

Abstract

Objectives: Mining of frequent item sets in transactional databases has been in widely use since years. This traditional technique involves mining of frequent itemsets on the entire set of records present in the transaction database at once. This has been causing performance issues such as out of memory, large turnaround time for the computation. The aim of the study is to propose suitable technique to overcome memory issues, reduce overall turnaround time and enable the determination of frequent item sets based on specific season or a time period. **Methods/Analysis:** Frequent item sets mining can be used for decision making in large number of real-life applications. With the growth in amount of data, quite a number of FIM (frequent itemset mining) approaches were proposed to meet the requirements of scalability. However, some existing approaches have met this requirement to some extent; they require high consumption of CPU and memory. In this study, we present another approach, named FIMUPT (frequent itemset mining using partitioning technique). The proposed technique eliminates processing of all records at once; instead it processes the records in small chunks in multiple increments. This technique makes various partitions of relevant size from large transaction database. It processes the partitions one after another instead of large transaction database. This scheme also has a provision to pass support threshold values for various partitions. Every partition of transaction database can be mapped to a season or a time period as needed. **Findings:** We observed that the proposed techniques perform very well in terms of computational time and memory usage. Size of the partition is decided based on the total number of transactions present, size of main memory and time period applicable for the entire transaction database. It consumes less memory since partition size is less compared entire transaction database. As the single partition is processed at once in the memory, it eliminates large memory needs of traditional technique [reference Wikipedia]. We have used sample dataset from the data mining library source spfm which is an open source. The size of the data sets we have used are 50MB and 100MB. Algorithms such as Apriori failed to run on 100MB datasets throwing out of memory error. **Applications:** However, with the partitioned approach successful executions were observed. Our test environment produced 15 partitions of entire transaction database. It concludes that with less memory size, we can process larger number of transaction records to perform data mining tasks. When compared with the existing approaches, experimental results tell that FIMUPT gives a performance gain of 19% on average. This technique eliminates out of memory issues seen in Apriori algorithm and its variant algorithm.

Keywords: Data Mining, Frequent Itemset, Partitioning, and Performance, Transaction Database

1. Introduction

FIM (Frequent Itemset Mining), as one of the most key research topics in data mining, is an approach to retrieve frequent items in datasets, that is widely used in the various fields[1,2]. This research paper provides new approach that can be used for determining frequent item sets from large transaction databases. Traditional technique is to scan through all records of database at once and these records will be kept in the main memory for processing. This approach causes problem when main memory cannot accommodate for the entire transaction database.

*Author for correspondence

An alternative to candidate generate-and-test based mining is pattern-growth mining, which avoids generating a large number of candidates[3] does exist. However, it still has performance issues with large data sets and does not have provision for time period based information retrieval. This new approach splits the large transaction database into multiple partitions and carries out frequent item set extraction algorithm on these partitions. Partitioning approach would help to reduce main memoryusage when frequent item sets methods are invoked by data mining algorithms. This approach also enables frequent item set retrieval on a time period basis. Experiments were conducted on a dataset extracted from large transaction database.

Since algorithm such as Apriori[4], requires that the transaction database to be residing in the memory, the objective of this paper is implemented in two stages. 1) Partitioning of large transaction database based on number of records, size of main memory and time period. 2) Perform frequent item set generation on each of these partitions. Analysis of computational time and memory usage is carried out for this approach.

2. Methodology

In this section, description about methodology applied and algorithms used are captured. Methodology for determining frequent item set involves various underlying sub tasks. These tasks include making sub partitions of large transaction database, iterating through all partitions created one after another, perform frequent item set retrieval on each partition and finally combine the results obtained from processing each partition. The steps followed in our approach are depicted in the flow (Figure 1).

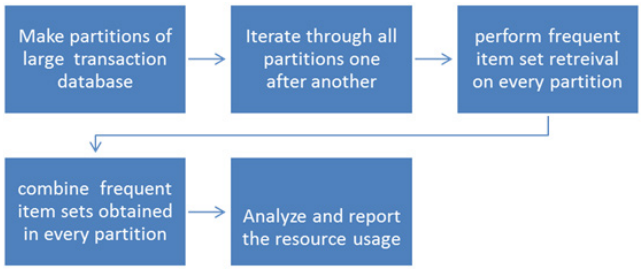

Figure 1. **FIMUPT** - partitioning based approach for frequent items set retrieval.

Algorithm: FIMUPT

Algorithm 1: P-DB

This algorithm is used for creating Q number of partitions from the entire transaction database.

Input: D, a transactional database; m, size of main memory; r, size of the record;

Output: P, set of partitions; P contains partition details in terms of start and end transaction date, which represents time period of a given partition

N ← |D| /* no: records in entire transaction db */
S ← N * r /* total size of the transaction db */
Q ← S/m /* Q represents total number of partitions */
X ← m/r /* no of records in partition */
For i = 1 to Q
sdate ← minimum of transaction date among X no of records in ith partition
edate ← maximum of transaction date among X no of records in ith partition
/* sdate and edate represents both start and end date of the corresponding partition */
add {sdate,edate} to P
Return P;

Algorithm 2: FI-PTDB

This algorithm is used for invoking frequent item set mining algorithm on each partition in iterations. The function find Fi can be existing algorithms such as Apriori, FP-Growth[5].

Input: D, a transactional database; P, set of partitions; ttable, a threshold table;

Output: PIs, a set of complete frequent items sets

For each partition Pq ∈ P do
Lq ←findFI(Pq, ttable[Pq]);
PIs ← PIs ∪ Lq;
Return PIs;

Frequent Item Set Retrieval based on Specific Time Period or a Season

Assuming a scenario of large supermarket tracks sales data for each item: each item is assigned a unique number. The retail market has a transaction database where every transaction consists of set of itemsets that were purchased and a timestamp to capture the transaction date.

Let us assume that transactions consist of itemsets as below:

Itemsets	Transaction Date	Season
{1,2,3,4}	12/Jun/2018	Rainy
{1,2,4,6}	13/Jun/2108	Rainy
{1,2,5}	13/Jul/2108	Rainy
{1,3,4}	14/Sep/2018	Winter
{2,3}	14/Sep/2018	Winter
{3,4}	2/Mar/2018	Summer
{1,4,7}	4/Apr/2018	Summer

Let us assume item number **1** represents item name umbrellaand item number **2** represents item name Rain Coat (used mainly during rainy season). We would apply regular Apriori technique to retrieve the frequent item sets of the above database. Let us assume that frequent item set is an item set if it is existed in at least 3 transactions: the value 3 here, is the *support threshold*. We would see the result as below.

Itemno	Min.Support
{1}	5
{2}	4
{3}	4
{4}	5

The itemsets{1},{2},{3},{4} with size 1 have a support threshold of minimum 3. With this all items are determined as frequent from entire database transactions. Current algorithms do not have provision to know information about frequent itemsets during specific season e.g. rainy season.

With the partitioning technique based on time period explained in the algorithm: **FI-PTDB** above, it is possible retrieve frequent itemsets during specific time period or a season. So when we apply Apriori using partitioning. ie find FI(3,Rainy) for the period Rainy and threshold value as 3, we would see the below result.

Item	Min.Support
{1}	3
{2}	3

From the above result, item number 1 and item number 2 are determined as frequent itemset during rainy season. This way partitioning technique enables information extraction during specific time period, mainly useful for retail enterprises.

3. Results of Experiments

Experiments were conducted on dataset obtained from the source spmf - an open source data mining library[6]. We ran the algorithms on datasets by both partitioning and without partitioning approach. Size of the dataset was around 50MB with 5898255 transactions. Recorded computational time for all the runs is shown via graph (Figure 1). Memory usage for all the runs is shown via graph (Figure 2 and 3). Comparison of features of FIMUPT with other existing algorithms is shown in Table 1. Experiments were conducted on following system environment.

- Processor type : Intel(R) Pentium(R) CPU N3540 @2.16GHZ
- RAM size : 4GB
- System : windows 8.1, 64 bit, x-64 based processor
- Java/JVM: 1.8

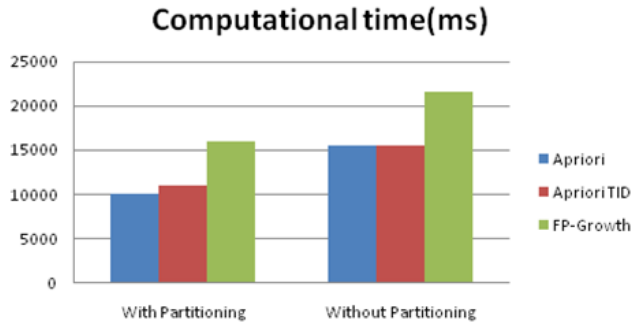

Figure 2. Turnaround time determined across different approaches of frequent item retrieval.

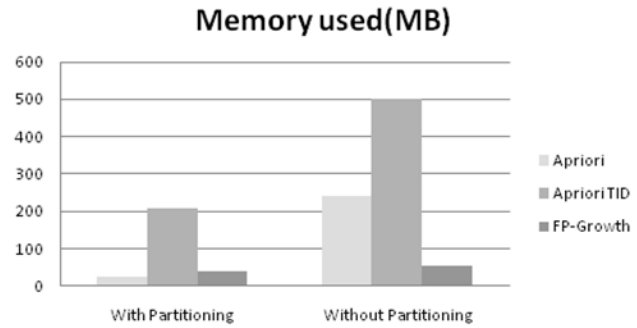

Figure 3. Memory usage determined across different approaches of frequent item retrieval.

Table 1. Comparison of features of FIMUPT with other existing algorithms

Comparison of features of FIMUPT with other existing algorithms				
Algorithm/features	**Computational time**	**Memory Usage**	**Scalability**	**Mining based on time period**
Apriori	High	High	low	No
Apriori -TID	High	Out of memory	low	No
FP-Growth	High	low	Medium	No
FIMUPT	Medium	optimal	High	Yes

4. Conclusion

We have come to following conclusions from the experimental findings.

- Most of the existing techniques on frequent itemsets retrieval will get into performance issues such as out of memory or large computational time on large datasets.
- The partitioning based technique discussed in this paper overcomes resource consumption problems.
- The partitioning based technique also enables season or time period based frequent item sets retrieval.
- This technique cannot be applied if the transaction record does not have transaction date captured for each transaction.

5. References

1. Ishita R, Rathod A. Frequent itemset mining in data mining: A survey. International Journal of Computer Applications. 2016; 139(9):15-8.
2. Abbar MAJ, Deekshatulu BL, Chandra P. A novel algorithm for utility-frequent itemset mining in market basket analysis. Proceedings of the 2015 Innovations in Bio-inspired Computing and Applications. 2015; p. 337-45.
3. Pei J, Han J, Wang W. Constraint-based sequential pattern mining: The pattern-growth methods. Journal of Intelligent Information Systems. 2007; 28(2):133-60. https://doi.org/10.1007/s10844-006-0006-z
4. Apriori Algorithm. Available from: https://en.wikipedia.org/wiki/Apriori_algorithm. Date accessed: 10/10/2018.
5. Han J, Pei J, Yin Y, Mao R. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. Data Mining and Knowledge Discovery. 2004; 8(1):53-87. https://doi.org/10.1023/B:DAMI.0000005258.31418.83
6. An open source data mining library. Available from: http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php. Date accessed: 08/01/2019.