THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

Cumulative Mining of Profitable Item Sets

Neetika Verma
M.Tech. Student, Department of Computer Science and Engineering,
VIT University, Chennai, Tamil Nadu, India
Muralidhar A.
Assistant Professor (Senior), Department of Computer Science and Engineering,
VIT University, Chennai, Tamil Nadu, India

Abstract:
Association-rule mining is used to mine the relationships among the existing items in transactional database. Sometimes transactions got deleted from database then we also have to maintain those transactions so that we can find the higher profitable item sets. In this paper we are considering the Incremental database that is Dynamic database in nature. Incremental database is database that we built in backing for running the data that continually increasing with in time. Incremental database efficiently stores a time series of data typically by having some fixed timescale. In this paper, we are trying to increase the efficiency of the previous methods of finding the high profitable items. Earlier we have to rescan the database many more times for finding the high profitable item which is not an efficient approach because as data is increasing the time taken to rescan the database is also more so to overcome this we can use distributed cache.

Keywords: *Association rule mining, frequent item sets, high utility item sets, actual profit, total profit, incremental mining, map reduce distributed cache*

1. Introduction

As the rapid increase of database techniques facilitates the demand of huge data and storage for business corporations, governments and scientific organizations The high profitable item sets mining issue is one of the most important and famous problem. Earlier the traditional algorithms are not capable of finding the profit information about item sets .It may happen that the item with higher support may have less profit. Profitable extraction of items was thus considered for overcoming the drawbacks of persistent items. The profitable variable for an item can be percentage, volume and rate. The algorithms often produce very high number of candidate keys so for that we need excessive memory requirements and it also requires large amount of running time for generating huge set of candidate keys. The earlier approaches concentrate on frequent number of items but in this we are concentrating those items which gives us more profit also for example – there is one organization who sells gold but selling gold is not as frequent as selling of clothes but in previous algorithm it will give only cloth as a frequent item set. What about gold which gives us more profit. So extract gold as a higher profitable item.

In real time environment the original database is to be huge and temporal because it changes time to time because transactions are occurring simultaneously. In temporal database there are two methodologies: Valid time and Transaction time. Valid time denotes that each record is valid for that particular period of time. Transaction database denotes that for how much time that particular record should be present in database. These two methodologies should not be same for each record. There are also three distinct forms of Temporal database. Historical database stores the data related to valid time, rollback database stores the data related to transaction time, bitemporal database stores the data related both valid and transaction time. Conventional profitable extraction from the database does not matter whether transaction is deleted or not .It is insufficient to waste the discovered enlightenment from the real database. FP-GROWTH algorithm allows profitable products finding without creating candidate item set and thus increase the acceleration time for finding the higher utility item sets and frequent item sets.

Two main conditions are there upper bound and lower bound for finding the cutoff of the items .Usually we match the occurrence of each item with the support and confidence for finding frequent item sets. If the item occurrence is less than the minimum support than that item is treated as infrequent item and as per the downward closure property if that item is occurred in any other item set that is also considered as in frequent item set. Association-rule mining is used to mine the relationships among the existing items in transactional database. Items are treated as a binary variable who's values are one or zero depends upon whether that item is present in database or not.

2. Review of Related Work

In this association rule mining, high utility item sets, FP Growth and apriori algorithm is going to be explained. Association rule mining is the method of finding the frequent item sets from the database .Association rules are used to find the occurrence of the items in the data set ,so that we can decide that how much benefit we will get from that particular item and how frequently that item is being used.

Apriori Algorithm:-In apriori user defined support and confidence is given and we have to consider that item with the user defined support and confidence. Items are treated as a binary variable who's values are one or zero depends upon whether that item is present in database or not. In real time environment, many items may buy simultaneously, and each item has its own percentage, volume and rate. Profit of the product is depending upon user interest. Profitable extraction of items was thus considered for overcoming the drawbacks of persistent items. The profitable variable for an item can be percentage, volume and rate. Two main conditions are there upper bound and lower bound for finding the cutoff of the items .Usually we match the occurrence of each item with the support and confidence for finding frequent item sets. If the item occurrence is less than the minimum support than that item is treated as infrequent item and as per the downward closure property if that item is occurred in any other item set that is also considered as in frequent.

FP Growth means frequent pattern item sets this algorithm is the improved version of apriori algorithm to great extent because it reduces the number of candidate keys. It also reduces the number of scans it only needs two scans

1. it uses the FP tree data structure
2. Then mine the concurrent items from the FP tree.

High utility item sets are those items which satisfy the minimum cutoff or support. Suppose if minimum cutoff is 40% for each item then the item must be greater than or equal to that minimum threshold. For finding these high utility item sets there are so many algorithms but the efficient one is which reduce the number of scans and number of candidate keys. In this paper we are using Map reduce for concatenating the profit and number of frequencies so that we get the high utility item sets. If we use map reduce the number of scans reduced as we are using the incremental database so updates are incrementally happen. so to reduce the database scan, we are using distributed architecture. In this data is divided to different clusters and then operations perform individually by every node by doing this time of scanning reduced. In this one scan is done by mapper part and other scan is done by reducer part.After that FP Growth part comes in the first part we just find the utility items After that we have to find high utility item sets we are finding high utility item sets by using FP Growth Algorithm which will reduce the number of candidate keys. As our data is updating incrementally we have to use the framework in which we can store the data and perform operation fastly.So in this we are using Mahout.

3. Proposed Work

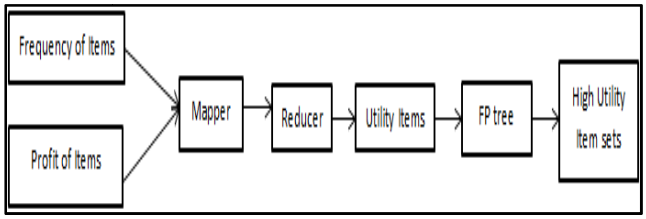

Figure 1

In this we are trying to improve the efficiency of the previous algorithm. By using Map Reduce we can reduce the number of rescans. Map will take a set of data and convert into another form it basically convert the individual item and convert into tuple in the form of (key, value) pair here we are taking item as a key and pass its number of occurrences and its profit or we can say price as a value. After that Reducer part will come in this we take the value (number of times one item is coming, Profit) and write the logic for concatenate these. After getting the result that value we considered as a utility itemsets. We take one user defined threshold value that we also called as support after getting the utility itemsets we have to find the high utility itemsets for finding high utility itemsets we have to match that utility with threshold if the utility is less than the threshold value than prune that item and if not store and generate the candidate set and again check the same. But then again the main task is how to reduce the generation of candidate Keys. So to generate less number of candidate keys we are using FP Tree here. It first generates the items in Descending order then generate tree without generating the candidate keys again and again. In this we are taking the input of original database, profit table, minimum cutoff. It will increase the efficiency and scalability also. We are using Mahout which is an algorithm library on hadoop .It is a scalable Machine learning it also uses the Map reduce paradigm. In mahout we can analyze the item in group and then we can analyze which Item is close to appear together. In this we are mapping the two input files as we are taking two tables we want both table available at the same time so that we can easily do mapping between two tables. For that we are using distributed cache after that we can multiply the number of occurrences of the item and profit related to that item.

4. Example

In this example, elaborate the proposed algorithm .The original Database which consist of items and transaction respectively.

Minimum cutoff given by user for example 30% .The original table contains all the items with the number of occurrences Table 1. Minimum cutoff checks the probability of the item that how frequently it will occur in future. The profit table is given for each item Table after multiply of these two we get the utility of the item sets.

ID	C0	CI	C2	C3
1	1	3	9	1
2	8	7	0	8
3	9	9	6	1
4	9	0	1	8
5	4	9	9	3
6	1	4	4	4
7	0	8	2	0
8	1	6	3	2
9	8	2	1	4

Table 1

Item	Profit
C0	10
C1	3
C2	6
C3	12

Table 2

Suppose these are the input table in which every item is having their number of occurrences in every transaction and every particular item is its own profit related to every transaction. So for finding the Utility of Items we have to do mapping between every item related to its profit. This Mapping will be done by using hash mapping after doing that we got all the Utility items after that we take the user threshold value and compare that value with all the value that we got as a result of mapper the values that is less than user threshold value are going to be pruned after that we get the utility item sets which we will take as the input of mahout in which we are using the FP tree approach.

This is calculated for every single item Now for calculating the actual profit for high weighted profit item sets we have to find the frequent item sets containing two or more candidate keys for that we can apply an efficient algorithm. From Table 1 consider first three transactions as our input then according to the proposed hashing algorithm initially the support is calculated for each item set and stored in dynamic list array. Take the support threshold(S) from the user to find the interesting patterns .If the threshold(S) is less than the threshold of each item set than that item is not a frequent item set. Else the generated item set is frequent..

6. Conclusion and Future Work

In this paper we proposed a method which is used for finding the high utility item sets efficiently by usingmaphashing. Map will take all the items as the key, value pair. Then reducer will take these items for finding the high utility item sets. According to the experimental the proposed approach is scalable Hence increases the efficiency. Simple apriori algorithm is used for finding the frequent item sets but it generates many number of candidate Keys in every pass which results in accumulating the implication. Although it is not suitable for finding the high utility item sets using apriori algorithm as complexity accumulated the efficiency will be reduced. Thus there is one of the technique to elevate the performance map reduce using distributed cache. Map reduce using hashing spawn the large item sets sufficiently and greatly. This hashing technique generates the item sets for each transaction, and put them into a table. Distributed cache can easily handle the incremental data which is updating in every second and Hash mapping is used for finding the utility item sets. Our database requires less scan and less time to read the database. It will also generate less number of candidate keys. How to improve the efficiency by reducing the number of candidate keys is still an issue which will be done in future and how to increase the efficiency of tree data structure by reducing the computation step by step will also be takecare. How we can remove the use of user threshold value criteria because its user defined and to decide that what to take the threshold value is a concern.

8. References

i. Agrawal, R., & Srikant, R., (1994). Fast algorithms for mining association rules in large databases. In The 20th international conference on very large data bases (pp.487–499).

ii. Agrawal, R., Imielinski, T., & Swami, A., (1993a). Mining association rules between sets of items in large databases. In International conference on management of data (pp. 207–216).

iii. Agrawal, R., Imielinski, T., & Swami, A. (1993b). Database mining: a performance perspective. IEEE Transactions on Knowledge and Data Engineering, 5, 914–925.

iv. Berzal, F., Cubero, J. C., Marín, N., & Serrano, J.-M. (2001). TBAR: An efficient method for association rule mining in relational databases. Data and Knowledge Engineering, 37, 47–64.

v. Brin, S., Motwani, R., Ullman, J. D., & Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. SIGMOD Record, 26, 255–264.
vi. Chan, R., Yang, Q., & Shen, Y. D., 2003. Mining high utility itemsets. In The 3rd IEEE international conference on data mining (pp. 19–26).
vii. Chen, M. S., Han, J., & Yu, P. S. (1996). Data mining: an overview from a database perspective. IEEE Transactions on Knowledge and Data Engineering, 8, 866–883.
viii. Cheung, D. W., Jiawei, H., Ng, V. T., & Wong, C. Y., 1996. Maintenance of discovered association rules in large databases: an incremental updating technique. In The12th international conference on data engineering (pp. 106–114). Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Mining and Knowledge Discovery, 8, 53–87.
ix. Hong, T. P., Lin, C. W., & Wu, Y. L. (2008). Incrementally fast updated frequent pattern trees. Expert Systems with Applications, 34, 2424–2435.
x. IBM Quest Data Mining Project (1996). Quest Synthetic Data Generation Code. http://www.almaden.ibm.com/cs/quest/syndata.html.
xi. Li, Y. C., Yeh, J. S., & Chang, C. C. (2005a). Direct candidates generation: a novel algorithm for discovering complete share-frequen itemsets. Lecture Notes in Computer Science, 551–560.
xii. Li, Y. C., Yeh, J. S., & Chang, C. C. (2005b). A fast algorithm for mining share-frequent item sets. Lecture Notes in Computer Science, 417–428.
xiii. Liu, Y., Liao, W. K., & Choudhary, A., 2005a. A fast high utility itemsets mining algorithm. In The 1st international workshop on utility-based data mining (pp. 90–99).
xiv. W. Wang, J. Yang, and P. Yu: Efficient Mining of Weighted Association Rules (WAR). 6th KDD (2000)
xv. Feng Tao, Fionn Murtagh, and Mohsen Farid: Weighted Association Rule Mining using Weighted Support and Significance Framework. 9th KDD (2003)
xvi. S. Lu, H. Hu, and F. Li: Mining weighted association rules. Intelligent Data Analysis, 5(3)(2001), 211-225
xvii. B. Barber and H.J.Hamilton: Extracting share frequent itemsets with infrequent subsets. Data Mining and Knowledge Discovery, 7(2) (2003), 153-185
xviii. Raymond Chan, Qiang Yang, Yi-Dong Shen: Mining high utility Itemsets. ICDM (2003)
xix. IBMdatagenerator,http://www.almaden.ibm.com/software/quest/Resources/index.shtml