Analysis of various frequent item mining methods

HKUST 19/20 Spring term (Comp4331)

Prepared by: Yu, Kijun (20319196)

Abstract

This report is based on the procedure of different frequent item mining with given transaction data and minimum support. The methods used in this assignment were Apriori algorithm, Hash Tree, and Frequent Pattern Growth. 3 separate methods were compiled in program respectively and the test was run for several times to get the result for comparison. These tests were carried out by spyder3 in Windows 10, the CPU used was Intel® Core™ i7-8700 @ 3.20GHz. Also, the closed itemset and maximal item set were mined based on the result of the apriori algorithm.

Key words: apriori, hash tree, fp-growth.

I. Introduction.

Different algorithms approach the dataset in non-identical manner. This difference of mining mechanism creates time difference in mining. Apriori would take the longest time over all since it repeatedly scans through whole transaction data set, where as other two types of algorithm scan a smaller number of times. For example. Fp growth algorithm scans the entire table only 2 times to build the frequent itemset in descending order of frequency and to build the tree. After first scan the non-frequent items were removed from transactions to created even smaller dataset. This would reduce the time for comparison as there are a smaller number of items in each transaction.

II. Apriori Algorithm

This method repeatedly scans the transaction table to check each items' frequency. The first set of items with length 1 can be easily created by checking each item's frequency in transaction. This item set can be also utilized for other mechanism as well. After extracting the frequency of items, infrequent items are removed from dataset. Supersets of item sets are created with k + 1 length. This created itemset first undergoes checking for infrequent subsets (pruning), this pruning of itemset is not required for length 1 and 2. In the pruning section, each subset of items is scanned though the previous item set with smaller length and if there is an infrequent subset, the item set is removed. This is due to the fact that a superset cannot have greater number of support than its own subsets. This makes sure that superset with infrequent subset will have infrequent support, hence removed.

After pruning the initial set of k items are made. Then these table are scanned through the whole transaction table again to check the frequency of each itemset. In this procedure again non-frequent set are removed to mine only frequent item sets. These steps are repeated until no more frequent set can be created or the length of single itemset reaches maximum length of single transaction.

With the given dataset and minimum support = 400, the apriori coding took nearly 11 to 12 minutes to be processed in the provided environment several results are averaged to calculate the time taken for processing data with apriori algorithm. Average value of the processing time was 11 minutes 27 seconds. This was with respective to repeated iteration through entire transaction table. There were 88162 transactions in the given dataset. Since single item set had almost 300 entries. On approx. calculation there was more than 26 million comparisons are done for single itemset. Figure 1 shows the result of apriori algorithm.

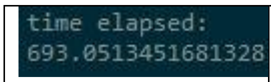
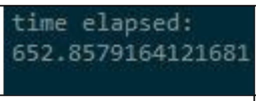
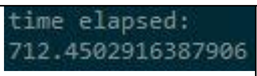
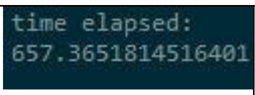
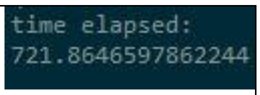
 time elapsed: 693.0513451681328	 time elapsed: 652.8579164121681	 time elapsed: 712.4502916387906	 time elapsed: 657.3651814516401	 time elapsed: 721.8646597862244
11min 33sec	10min 52sec	11min 52sec	10min 57sec	12min 1sec

Figure 1: Apriori algorithm time

III. Hash tree Algorithm

With base of apriori algorithm for creating L1 table (table of single entries with frequency above min) the code was slightly implemented for hash tree algorithm. Two Class items were created in implementing the program for hash tree algorithm. A tree with several member function to generate the tree and tree node in trees to link items together as child of each node. The hash tree is built with each of entries in frequent item set with length k, then the transaction entries are split into k length of subsets to provide support in the hash tree. By doing so transactions are only compared once for each hash tree with length k and this resulted in reduction of time in very big margin. It only took 10 and several seconds to process the transaction and get frequent item sets. After adding the support to the tree, the tree is only visited once to provided itemset with support above pre-mentioned minimum support. These steps are repeated for longer numbered entries until no more hash tree has itemset with support above minimum

Figure 2 shows the processing time of hash tree algorithm implemented from apriori algorithm. The average value of taken time was 12.8 seconds

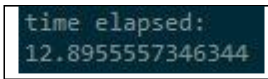
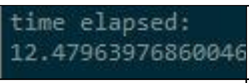
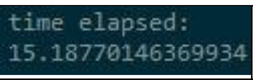
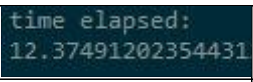
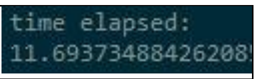
 time elapsed: 12.8955557346344	 time elapsed: 12.47963976860046	 time elapsed: 15.18770146369934	 time elapsed: 12.37491202354431	 time elapsed: 11.69373488426208
12.8seconds	12.4seconds	15.1seconds	12.3seconds	11.6seconds

Figure 2: Hash tree algorithm time

IV. Fp tree algorithm

Fp tree algorithm takes different approach from hash tree algorithm. The single item sets are initially mined and arranged in frequency order, and the transaction is rebuilt with only frequent item sets with order of descending frequency. Hence the process entirely checks the transaction table entries only 2 times.

There are several advantages of this algorithm. First is of course a smaller number of table search compared to another implementation. In this algorithm, pairing of items are not required to make the process faster. The database is stored in a compact version of memory as same transaction share same nodes and even shorter transaction share the nodes. Hence, it is efficient and scalable for mining both long and short frequent patterns.

However, implementing FP Tree algorithm is more cumbersome and difficult to build than Apriori. And due to its complexity, it may be expensive. Also, when the database is large, the algorithm may not fit in the shared memory.   The average time taken for this algorithm was 3min 26.2secs

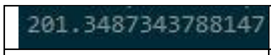
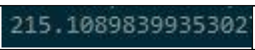
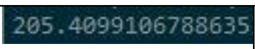
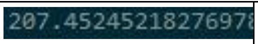
 201.3487343788147	 215.108983993530	 205.4099106788635	 207.4524521827697	203.222758293151
3min 21sec	3min 35sec	3min 25sec	3min 27sec	3min 23sec

Figure 3: Fp Tree algorithm time

V. Closed and maximal item set

Closed item set and maximal item set are type of frequent item set. The mined values from apriori algorithm was further processed to mine closed and maximal item sets. The mining of these two sets take less than a second because it is mined from already produced frequent set. Hence one would simply say the time for getting closed and maximal itemset is equal to the time taken for whole process of mining itself. To implement this function for mining closed and maximal items a single class was implemented to keep list of maximal items and closed items, since maximal items are subset of closed items both could be processed at same time with small conditions in implementing the function.

Maximal item are items with no direct superset, closed are items with no support same as its own in its supersets, so in both case the items are compared to its superset and processed to mine them altogether.

VI. Conclusion

Apriori algorithm is longest in all algorithm as it is based on comparing the frequent items with transactions one by one, the hash tree algorithm takes least time since it processes items by index and hash keys. Lesser comparison is required for hash tree algorithm. For the fp tree algorithm, the transactions are processed less time but the algorithm itself is complex compared to the others. And this makes the algorithm very expensive as to keep tract of all the headers and arrange transaction or even travel tree in reverse manner. Hence, depending on the data set, if one needs to mine the frequent items the algorithms are needed to be chosen accordingly to process it accurately and fast.