An Association Mining of Efficient by Compressed Large Database

Garima Dagaria*
Ritesh Joshi**

Abstract

Large Amount of data is being using very rapidly in the world. It to be compressed takes much more time and takes lot of effort to process these data for knowledge discovery and decision making. Data compression technique is one of good solutions to be reduce size of data that can be save more time the time of discovering useful knowledge by using appropriate methods, for example Data mining. Data mining is used to help users discover interesting and useful knowledge more easily to decision making purpose. It is more and more popular to apply the association rule mining in recent years because of its wide applications in many fields such as stock analysis, web log mining, medical diagnosis, customer market analysis and bioinformatics. In this paper the main focus in on association rule mining and data pre-process with data compression. In this paper we analysis the methods simple Apriori, Partion based Apriori and Apriori with compressed dataset. We compare these three methods on the basis of minimum support, minimum confidence, number of records and execution times

Keywords - *Association rule, Apriori Algorithem,merged transaction, quantification table.*

Introduction

Data compression is one of good solutions to be reducing size of large data that can save it time of discovering useful knowledge by using appropriate methods, for example, data mining. Data mining is used to help users discover interesting and useful knowledge more easily for decision making discovery. It is more and more popular to apply the association rule mining in recent years because of its large and wide applications in many fields such as stock analysis, web log mining, medical diagnosis, customer market analysis, and bioinformatics[4]. In this research, the main focus is on association rule mining and data pre-process with data compression. Proposed process is a knowledge discovery from compressed databases in which can be decomposed into the following two steps:

(1) Data pre-process step:

Data pre-process transforms the original database into a new data representation where several transactions are merged to become a new transaction. Eventually, it generates a new transaction database at the end of the data pre-process step.

(2) Data mining step:

It uses an Apriori-like algorithm of association rule mining to find useful information[5]. There are some problems in this approach. First, the compressed database is not reversible after the original database is transformed by the data pre-process step. It is very difficult to maintain this database in the future. Second, although some rules can be mined from the new transactions, it still needs to scan the database again to verify the result [3]. This is because the data mining step produces potentially ambiguous results. It is a serious problem to scan the database multiple times because of the high cost of re-checking the frequent itemsets.

It is even a bigger challenge to maintain the compressed database in the future. In addition, it spends too much time to check candidate itemsets in the data mining step. In this research, a more efficient approach, called Mining Merged Transactions with the Quantification Table is proposed, which can compress the original database into a smaller one and perform the data mining process without the above problems. Our approaches have the following characteristics:

(a) The compressed database can be decompressed to the original form.

(b) Reduce the process time of association rule mining by using a quantification table.

*Student, Medicaps Institute of Technology, Indore
*Assistant Professor, Medicaps Institute of Technology, Indore

(c) Reduce I/O time by using only the compressed database to do data mining.

(d) Allow incremental data mining

Previous Work

How to reduce the data space in the process becomes a challenge issue. Data compression provides a good solution which can lower the required space. Data mining has many useful applications in recent years because it can help users discover interesting knowledge in large databases. However, existing compression algorithms are not appropriate for data mining. In two different approaches were proposed to compress databases and then perform the data mining process. However, they all lack the ability to decompress the data to their original state and improve the data mining performanceSupport (X) = |T(X)| / |D|
Confidence (X→Y) = Support (X ∩ Y)/Support (X)

In support-confidence framework, if it is an interesting relation for X →Y, then X and Y must be frequent. How to define a frequent relation? There are two conditions. One condition is support (X) ≥ minsupport and support (Y) ≥ minsupport(Y). Another is Confidence (X→Y) ≥ minconfidence. Minsupport and minconfidence are user-defined thresholds [6].

The problems of mining association rules are mainly divided into two sub-problems. One is to discover the frequent itemsets and another is to generate the association rules. The first problem is more difficult than the second one.

The apriori algorithm is one of the classical algorithms in the association rule mining. It uses simple steps to discover frequent itemsets.

Algorithm Apriori :

1) L1= {Large 1-itemsets};

2) for (k=2; Lk-1 ≠ 0; k++) do begin

3) Ck = apriori-gen (Lk-1); // New candidates

4) for all transactions t D do begin

5) Ci = subset(Ck, t); // Candidates contained in 1

6) for all candidates c Ct do

7) c.count++;

8) end;

9) Lk = {c Ck | c.count ≥minsup}

10) end

11) Answer = ←k Lk;

Lk is a set of k-itemsets. It is also called large k-itemsets. Ck is a set of candidate k-itemsets. How to discover frequent itemsets? Apriori algorithm finds out the patterns from short frequent itemsets to long frequent itemsets. It does not know how many times the process should take beforehand. It is determined by the relation of items in a transaction. The process of the algorithm is as follows[7]:

At the first step, after scanning the transaction database, it generates frequent 1-itemsets and then generates candidate 2-itemsets by means of joining frequent 1-itemsets.

At the second step, it scans the transaction database to check the count of candidate 2-itemsets. It will prune some candidate 2-itemsets if the counts of candidate 2-itemsets are less than predefined minimum support. After pruning, the remaining candidate 2-itemsets become frequent 2-itemsets which are also called large 2-itemsets. It generates candidate 3-itemsets by means of joining frequent 2-itemsets. Therefore, CK is generated by joining large (K-1)-itemsets obtained in the previous step. Large K itemsets are generated after pruning. The process will not stop until no more candidate itemset is generated.

Since most data occupy a large amount of storage space, it is beneficial to reduce the data size which makes the data mining process more efficient with the same results. Compressing the transactions of databases is one way to solve the problem [10].

Merge-Mining Algorithm

It is very effective to reduce the size of a transaction database. Their algorithm is divided into data preprocess and data mining. There are two sub-

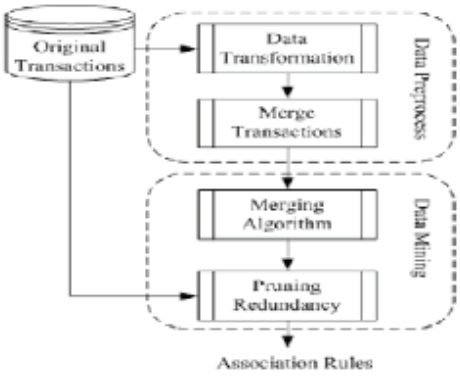


Fig 1: An overview of the Merged Transaction Algorithm

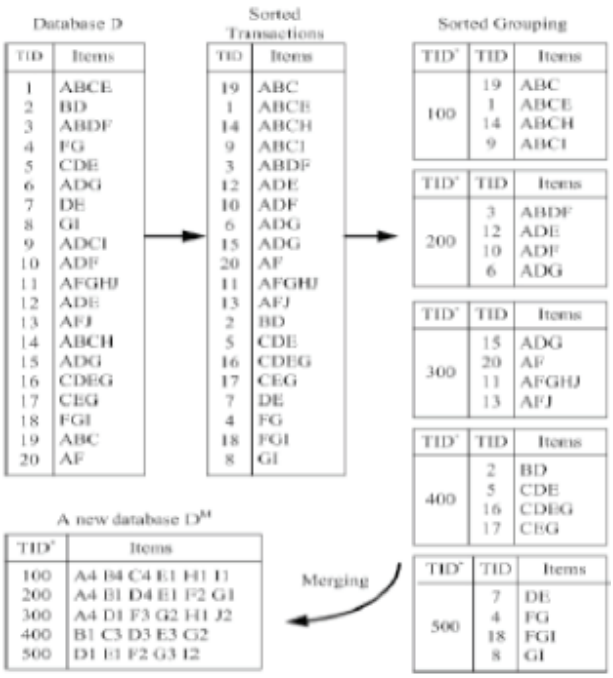


Fig 2: An example of sort grouping

processes in the data preprocess. One sub-process transforms the original database into a new data representation. It uses lexical symbols to represent raw data. Here, it's assumed that items in a transaction are sorted in lexicographic order. Another sub-process is sorting all the transactions to various groups of transactions and then merges each group into a new transaction [7].

For example, T1= {A, B, C, E} and T2 = {A, B, C, D} are two transactions. T1 and T2 are merged into a new transaction T3= {A2, B2, C2, D1, E1}. Fig.2 shows an example of the sort grouping method and Fig.3 shows an example of data mining sub-process.

The process called merge-mining algorithm is used to find frequent itemsets from the new transaction DM. There are two phases in this algorithm. The first phase is finding frequent itemsets. The second phase is to prune redundancy. It is possible that frequent itemsets generated in the first phase might not exist in the DM. For this reason, it needs to verify those frequent itemsets by scanning DM again[12].

Proposed Method

This section focuses on compressing related transactions and building a quantification table for pruning candidate itemsets that are impossible to become frequent itemsets. Algorithms like compress transactions to reduce the size of a transaction database. Then, they use Apriori-like algorithms to mine the compressed database. Whereas the two phase's approach of compression and data mining are used, they suffer the following problems:

(1) In the data compression phase, the original database can not be recovered to support transaction updates.

(2) In the data mining phase, a lot of candidate itemsets could be generated in a large transaction database[3].

Since both need to scan the database more than once, they have a much higher process cost. The first problem is due to the lack of rule or constraint in the process of merging transactions in the data compression phase. Therefore, the compressed database can not be decompressed to its original form In addition, they don't use user-defined threshold to filter infrequent 1-itemsets from the original database.

Another problem is that Apriori-like algorithms generate a lot of candidate itemsets and need to check the candidate itemsets by scanning the database. It is very time-consuming.

In order to provide a better performance, we limit the number of database scan to be one in the data compression phase and build a quantification table. In the data mining phase, we use the same approach of Apriori algorithm to generate candidate itemsets and reduce the number of candidate itemsets by using the quantification table. We also reduce the time of scanning the database.

We called our approach the Mining Merged Transactions with the Quantification Table (MMTQT) which has three phases:

(1) Merge related transactions to generate a compressed database

(2) Build a quantification table

(3) Discover frequent itemsets

MMTQT Approach

First, MMTQT uses the transaction relation distance to merge the relevant transactions. The definition of the transaction relation distance is defined D introduce how to build a quantification table. Then, it illustrates the process of compressing a database. How to compute support of itemsets from minimum-frequency function. Finally, it explains how to recover data from the compressed database.

Architecture

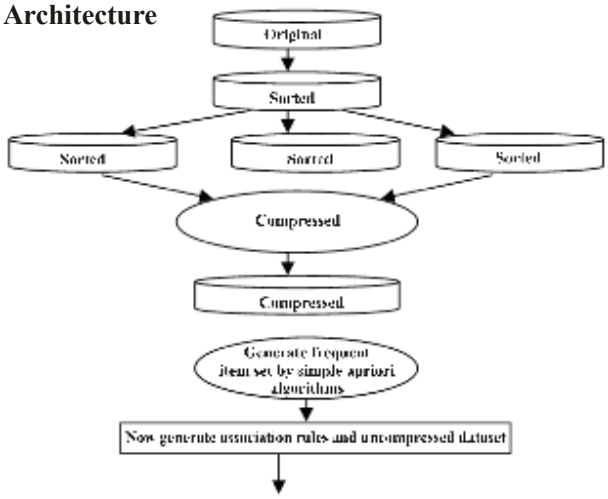


Fig 3: An example of data mining

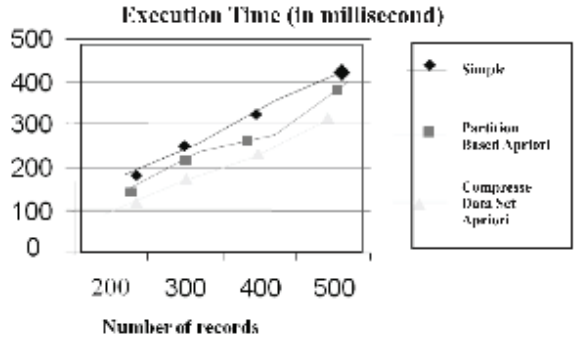
Experimental Result

M2TQT and Merged Transactions Approach were implemented in java programming language and all experiments run on a PC of Intel Pentium 4 3.0GHz processor with DDR 400MHz 4GB main memory. Synthetic datasets are generated by using the IBM dataset generator for our experiments.The dataset T4I5D10k is used to run, our algorithm, merged transactions approach and Apriori algorithm[2].

Let the average size of the potentially large itemset be 5 for the minimum supports 5%, 10%, 15%, and 20%,

Table III: The experiment

Number of records	Time taken to execute (In millisecond) Simple Apriori Algorithm	Time taken to execute (In millisecond) Partition based Apriori algorithm	Time taken to execute (In millisecond) Compressed Transaction based Apriori
200	183	122	103
300	252	234	177
400	343	283	232
500	426	401	312



and compare our algorithm with Apriori algorithm and merged transactions approach. The performance of our algorithm is much better.

Conclusions

In this paper, our experiments lead to the following conclusions. Form the experiment it clear that compressed transaction data set methods is much better as compare to the other methods a more efficient approach, called Mining Merged Transactions with the Quantification Table is proposed, which can compress the original database into a smaller one and perform the data mining process efficiently. Our approaches have the following characteristics:

(a) The compressed database can be decompressed to the original form.

(b) Reduce the process time of association rule mining by using a quantification table.

(c) Reduce I/O time by using only the compressed database to do data mining.

(d) Allow incremental data mining.

References

1. M. C. Hung, S. Q. Weng, J. Wu, and D. L. Yang, "Efficient Mining of Association Rules Using Merged Transactions," in WSEAS Transactions on Computers, Issue 5, Vol.5, pp. 916-923, 2006.

2. M. Z. Ashrafi, D. Taniar, and K. Smith, "A Compress-Based Association Mining Algorithm for Large Dataset," in Proceedings of International Conference on Computational Science, pp. 978-987, 2003.

3. R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," in Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487-499, 1994.

4. D. Xin, J. Han, X. Yan, and H. Cheng, "Mining Compressed Frequent-Pattern Sets," in Proceedings of the 31st international conference on Very Large Data Bases, pp. 709-720, 2005.

5. G. Grahne and J. Zhu, "Fast algorithms for frequent itemset mining using FP-trees," IEEE Transactions on Knowledge and Data Engineering, Vol. 17, pp. 1347-1362, 2005.

6. M. Z. Ashrafi, D. Taniar, and K. Smith, "A Compress-Based Association Mining Algorithm for Large Dataset," in Proceedings of

International Conference on Computational Science, pp. 978-987, 2003.

7. Ashrafi and K. Smith, “Data Compression-Based Mining Algorithm for Large Dataset," in Proceedings of International Conference on Computational Science, 2003.
8. D. I. Lin and Z. M. Kedem, "Pincer-search: an efficient algorithm for discovering the maximum frequent set," IEEE Transactions on Knowledge and Data Engineering, Vol. 14, pp. 553-566, 2002.
9. E. Hullermeier, "Possibility Induction in Decision-Tree Learning," in Proceedings of the 13th European Conference on Machine Learning, pp. 173-184, 2002. Cheung, W., "Frequent Pattern Mining without Candidate Generation or Support Constraint.", Master's Thesis, University of Alberta, 2002.
10. Huang, H., Wu, X., and Relue, R. Association, Analysis with One Scan of Databases, Proceedings of the 2002 IEEE International Conference on Data Mining. 2002.
11. Zaki, M. J., Parthsarathy, S., Ogihara, M., and Li, W. New Algorithms for Fast Discovery of Association Rules. KDD, 283-286. 1997. Agarwal, R., Aggarwal, C., and Prasad, V.V.V. 2001.
12. Goulbourne, G., Coenen, F., and Leng, P. H. Computing association rule using partial totals. In Proceedings of the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases, 54-66. 2001.
13. Pei, J., Han, J., Nishio, S., Tang, S., and Yang, D. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. Proc.2001 Int.Conf.on Data Mining. 2001.
14. Orlando, S., Palmerini, P., and Perego, R. Enhancing the Apriori Algorithm for Frequent Set Counting. Proceedings of 3rd International Conference on Data Warehousing and Knowledge Discovery. 2001.
15. Grahne, G., Lakshmanan, L., and Wang, X. 2000. Efficient Mining of Constrained Correlated sets. In Proc. 2000. Int. Conf. Data Engineering (ICDE'00), San Diego, CA, pp. 512–521.