Research Article

An Association of Efficient Mining by Compressed Database

Anjani Pandey¹, Gayatri Singh²

*Corresponding author:
Anjani Pandey

¹Dept of Computer Engineering V.I.T.S, Satna (M.P)
²Dept of CSE (Software System) V.I.T.S, Satna (M.P)

Abstract

Data mining can be viewed as a result of the natural evolution of information technology. The spread of computing has led to an explosion in the volume of data to be stored on hard disks and sent over the Internet. This growth has led to a need for data compression, that is, the ability to reduce the amount of storage or Internet bandwidth required to handle the data. This paper analyses the various data mining approaches which is used to compress the original database into a smaller one and perform the data mining process for compressed transaction such as M2TQT, PINCER-SEARCH algorithm, APRIORI & ID3 algorithm, AIS & SETM, CT-Apriori algorithm, CBMine, CT-ITL algorithm, FIUT-Tree. Among the various techniques M2TQT uses the relationship of transactions to merge related transactions and builds a quantification table to prune the candidate item sets which are impossible to become frequent in order to improve the performance of mining association rules. Thus M2TQT is observed to perform better than existing approaches.

Keywords: Quantification table, Association Mining, Merge Transitions.

Introduction

The current parallel and distributed algorithms are based on the serial algorithm Apriori. An excellent survey given in classifies the algorithms by load-balancing strategy, data parallelism and task parallelism. The two paradigms differ in whether the candidate set is distributed across the processors or not. In the data parallelism paradigm, each node counts the same set of candidates. may or may not be partitioned in either paradigm theoretically. In practice for more efficient I/O it is usually assumed the database is partitioned and distributed across the processors. In the data parallelism paradigm, the database is distributed across the processors. Each processor is responsible for computing the local support counts of all the candidates, which are the support counts in its database partition. All processors then compute the global support counts of the candidates, which are the total support counts of the candidates in the whole database, by exchanging the local support counts (Global Reduction). Subsequently, large item sets are computed by each processor independently.

Merging Process

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.
Compressing the transactions of databases is one way to solve the problem. Figure 1.2 shows overview of the Merged Transaction Algorithm [2]. It is very effective to reduce the size of a transaction database. Their algorithm is divided into data preprocess and data mining.

**Transaction Relation Distance**

Based on the relation distance between transactions one can merge transactions with closer relationship to generate a better compressed database. Here the transaction relation and transaction relation distance are defined as follows:

Definition

1. Transaction Relation: The relation between two different transactions T1 and T2 is that T1 is either a subset or a superset of T2.

2. Transaction Relation Distance: Distance is the number of different items between two transactions.

**A Quantification Table**

To reduce the number of candidate item sets to be generated, additional information is required to help prune non-frequent item sets.

**Table 1.1: An Example Database**

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>ABCDE</td>
</tr>
<tr>
<td>200</td>
<td>CDE</td>
</tr>
<tr>
<td>300</td>
<td>ACD</td>
</tr>
</tbody>
</table>

For instance, after reading the transaction {ABCDE} of TID 100, it knows the transaction length \( n \) is 5. For the prefix-item A, the counters under L5 to L1 are all increased by one from the initial value of zero. That is, A1 appears in each Li where \( i = 5 \) to 1. For the prefix-item B, the counters under L4 to L1 are all increased by one as well. That is, B1 appears in each Li where \( i = 4 \) to 1. The same process is performed for items C, D, and E. Similarly, after reading TID 200 {CDE}, the table has C2 in L3, L2, and L1; D2 in L2 and L1; E2 in L1. Finally, with the last transaction {ACD}, it will increase the counters by one from A1 to A2 in L3, L2, and L1; C2 to C3 in L2 and L1; D2 to D3 in L1. Table 2 shows the result of building the quantification table. With this table, we can easily prune the candidate item sets whose counters are smaller than the minimum support.

**Conclusion**

Simplify the description items in each transaction are presented in a lexicographical order. Algorithms like compress transactions to reduce the size of a transaction database. They use Apriori-like algorithms to mine the compressed database.
References


[12]. Beil F, Ester M, Xu X. Frequent Term-Based Text Clustering, ACM SIGKDD, 2002


[15]. Pei J, Han J, Nishio S, Tang S, Yang D. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. Proc.2001 Int.Conf.on Data Mining.

