A Graph-Based Approach for Mining Closed Large Itemsets

Lee-Wen Huang

Dept. of Computer Science and Engineering
National Sun Yat-Sen University

huanglw@gmail.com

Ye-In Chang

Dept. of Computer Science and Engineering
National Sun Yat-Sen University

changyi@cse.nsysu.edu.tw

Abstract

Data Mining means a process of nontrivial extraction of implicit, previously and potentially useful information from data in databases. Mining closed large itemsets is a further work of mining association rules, which aims to find the set of necessary subsets of large itemsets that could be representative of all large itemsets. In this paper, we design a graph-based approach, considering the character of data, to mine the closed large itemsets efficiently. Two features of market basket analysis are considered – the number of items is large; the number of associated items for each item is small. Combining the cut-point method and the clique concept, the new algorithm can find the closed large itemsets efficiently. The simulation results show that the new algorithm outperforms the FP-Growth algorithm in the execution time and the space of storage.

Keywords: Association rule, Closed large itemsets, Data mining, Graphic-based mining

1. Introduction

In recent years, with the computers and information industries growing more and more rapidly, varies data around us become more complexity and huge. Commercial behavior, scientific statistics, natural phenomena and DNA projects are some examples to produce lots of data every day. In the past, we may put these data in drawers or databases to extract some information efficiently. However, as the amount of data grows, it becomes very difficult to determine the useful data. Generally speaking, there are four kinds of problems that we have met (Agrawal and Srikant, 1995; Han et al., 2007; Lucchese et al., 2006; Srikant and Agrawal, 1996; Zaki, 1998). First, there are too much information for us to digest. Second, it is difficult to recognize the reality of the information. Third, it is hard to guarantee the security of these information. Finally, we cannot deal with different forms of information easily because of much data. In order to solve these four problems, people start to think how to find useful knowledge without overwhelming by information flooding. Although recent database management systems can provide quick insertion, deletion, query and statistic, they cannot detect the relations and rules between data. So, there comes a research named *data mining* which is used to help making decisions and information retrieval.

Data mining is one kind of analysis techniques which are data driven. This is different from Online Analytical Processing (OLAP) as well as the ad-hoc query and reporting approach, both of which are hypothesis driven (Kamath, 2001). A hypothesis is formulated by the query types. That is, if a pattern cannot be formulated, we cannot use the hypothesis to find this pattern. However, since data mining is data driven, it is not important whether we know the rules or query types, the expected patterns will be discovered in the process of data mining.

There are many methods for data mining, such as *association rule mining, sequential pattern mining, attribute oriented induction, data classification, data clustering, pattern-based similarity search, and data cube* (Lee, 2007). The tasks of finding association rule in large databases will bring lots of benefits for many applications, such as bioinformatics, market-basket analysis, customer relationship management (Kamath, 2001), etc. There are several Apriori-based algorithms, such as AprioriAll (Agrawal and Srikant, 1995) and GSP (Srikant and Agrawal, 1996), have been proposed. However, their search space is extremely large. For example, with m attributes, there are $O(m^k)$ potentially frequent sequences of length k. With millions of objects in the database, the problem of I/O

minimization becomes paramount. Moreover, these algorithms are iterative in nature. Therefore, they need to scan the whole database many times, which is obviously very expensive in I/O cost (Zaki, 1998).

A large itemset X is closed if there is no large itemset X´ such that (1) X $\subset$X´ and (2) $\forall$ transaction T, X $\in$T $\rightarrow$ X´ $\in$T. For example, if we have the set of frequent itemsets S is {(x):3,(y):3, (z):2, (x, y): 3, (x, z):2, (y, z):2,(x, y, z): 2}, then the related set of closed frequent itemsets C = {(x, y): 3, (x, y, z): 2}. All large itemsets can be obtained from closed large itemsets without losing support information.

However, the large itemsets may not always correlate with each other (Huang and Chang, 2009). Ohsawa and Yada incorporate full consideration of two distinct features of market basket analysis into the algorithm design (Ohsawa and Yada, 2009). The first feature is that the number of items can be large; the second feature is that the number of associated items for each item is relatively small. In this paper, we propose a new graphic-based algorithm, the GCFP algorithm, to discover the closed large itemsets. We conduct several experiments using different synthetic datasets. These simulation results show that the proposed algorithm outperforms the FP-Growth algorithm in all datasets. The main reason is that our algorithm could generate the densely structure than the FP-Growth algorithm.

The rest of the paper is organized as follow. Section 2 gives a survey of several well-known data mining algorithms for association rules. In Section 3, we present a new graph-based algorithm for mining closed large itemsets. In Section 4, we give a comparison of the performance of the FP-Growth algorithm and our proposed algorithm. In Section 5, we give a summary and point out some future research directions.

2. Related work

2.1 **Apriori**

Agrawal and Srikant proposed the Apriori algorithm (Agrawal and Srikant, 1994) for mining association rules. This algorithm generates the candidate itemsets to be counted in an iteration by using only the itemsets found large in the previous iteration. This reduces the number of candidate itemsets. However, for each candidate itemset, it needs to count its appearances in all transactions. In this algorithm, each iteration requires one pass over the database and it wastes a lot of time.

2.2 **FP**-Growth

Unlike Apriori, Han proposed the FP-Growth algorithm (Han et al, 2007) for mining association rules. First, a data structure, called the FP-Tree, is constructed, which is an extended prefix-tree structure storing crucial, quantitative information about large itemsets. Only L1 will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of sharing nodes than less frequently occurring ones. Second, an FP-Tree-based pattern fragment growth mining method, is developed, which starts from L1, examines only its conditional pattern base, constructs its conditional FP-Tree, and performs mining recursively with such a tree.

2.3 The Graphic Representation

Huang and Chang proposed the GAR algorithm (Huang and Chang, 2009). We give a simple example to illustrate how to denote large itemsets by using the graphic representation. An example is given in Figure 1. In the first iteration, we scan all the transactions to count the number of occurrences for 1-itemsets and 2-itemsets. Assuming that the minimum transaction support is two, the set of large 1-itemsets, L1, and 2-itemsets, L2, can then be determined. Then, we let each item be a node in the graph, and draw a line from A to B if {A,B} is a large 2-itemsets. As observed in (Agrawal and Srikant, 1994), any subset of a large itemset must be a large itemset by itself. That is, {A,B,C} in L3 implies {A,B} in L2, {A,C} in L2 and {B,C} in L2. Based on this observation, if {A,B,C} in L3, the degree of node A must be greater than 2 or equal to 2. So, we can discard the nodes whose degree is less than 2. For example, the degree of node I is 0, and the degree of node K is 1. These nodes cannot appear in L3.

Finally, we can get the large itemsets, {A,B,D}, {B,C,D} and {F,G,H}. The total number of nodes in the GFP-Forest is 10, whereas the total number of nodes in the FP-tree is 27 which represents a reduction ratio of 2.7.

TID	Items
1	ABDE
2	AJL
3	DEFKL
4	BCDJ
5	CKL
6	ABCD
7	DEI
8	BCDFHG
9	JK
10	FGHIJ

Figure1 Database D

3. The Graph-Based Approach

In this section, we propose the GCFP algorithm. It represents the large itemsets as a graph, which constructs a graph based on L2. Based on the property of the graph, we partition the graph into different sub-graphs, which results in the process time of mining association rules can be reduced.

3.1 The Algorithm

GCFP Algorithm is shown as Figure 2. We use the same example, which is given in Figure 1, to go through the algorithm. First, we generate L1 and L2 by scanning database once. We count 1-itemsets in an array. In the same time, we count 2-itemsets in a sparse matrix, in which only the lower triangular part is used. Then, we check whether the minimum support requirement is met.

```
Procedure GCFP;
Begin
    GenerateL1L2();
    ConstructGraph();
    CutGraph();
    FindClique();
    ConstructGCFP-Forest();
End;
```

Figure 2 The GCFP algorithm

Second, based on L2, we call procedure ConstructGraph to construct the graph and assign the different group number to the disconnected graphs. Next, we use the cut-point property of the graph to divide a graph. A connected undirected graph is biconnected if there are no vertices whose removal disconnects the rest of the graph. If a graph is not biconnected, the vertices whose removal would disconnect the graph are known as cut points. Depth-first search provides a linear-time algorithm to find all cut points in a connected graph. First, starting at any vertex, we perform a depth-first search and number the nodes as they are visited. For each vertex V, we call this preorder number V.Num. Then, for every vertex V in the depth-first search spanning tree, we compute the lowest-numbered vertex, which we call V.Low, that is reachable from V by taking zero or more tree edges and then possibly one back edge.

Then, we call procedure CutGraph to divide the graphs to sub-graphs. Based on groups, we transform the database D to database D', as shown in Figure 3. Then, we find the maximal cliques and

construct the GCFP-Forest using the similar structure of FP-tree. Finally, we can get the closed large itemsets, {A,B,D}, {B,C,D} {F,G,H}, {B,D}, {D,E}, {D,F}, {K,L}, {A}, {C}, {D}, {E}, {F}, {I}, {J}, {K}, and {L}.

TID	Items	
1	ABDE	DBA
2	AJL	A
3	DEFKL	D,F
4	BCDJ	DBC
5	CKL	C
6	ABCD	DBCA
7	DEI	D
8	BCDFHG	DBC, FGH
9	JK	(null)
10	FGHIJ	FGH

Figure3 Database D'

4. Performance

In this Section, we study the performance of the proposed algorithm. In this paper, we only make a comparison of our proposed algorithm with the FP-growth algorithm. Our experiments were performed on a Pentium Dual-Core personal computer with one CPU clock rate of 2.7 GHz, 2GB of main memory, running Windows XP Professional, and coded in Java. The data was stored on a local 500G IDE 3.5" drive.

4.1 Generation of Synthetic Data

We generated synthetic transactions to evaluate the performance of the algorithms over a large range of data characteristics. The synthetic data is said to simulate a customer buying pattern in a retail environment. The parameters used in the generation of the synthetic data are shown in Table 1. The length of a transaction is determined by a Poisson distribution with mean \mu equal to |T|. The size of a transaction is between 1 and |MT|. The transaction is repeatedly assigned items from a set of potentially maximal large itemsets F, until the length of the transaction does not exceed the generated length.

Table 1: Parameters

$\|D\|$	Number of transactions
$\|T\|$	Average size of transactions
$\|MT\|$	Maximum size of the transactions
$\|I\|$	Average size of maximal potentially large itemsets
$\|MI\|$	Maximum size of the potentially large itemsets
$\|L\|$	Number of maximal potentially large itemsets
N	Number of items

The length of an itemset in F is determined according to a Poisson distribution with mean u equal to |I|. The size of each potentially large itemset is between 1 and |MI|. Items in the first itemset are chosen randomly from the set of items. To model the phenomenon that large itemsets often have common items, some fraction of items in subsequent itemsets are chosen from the previous itemset generated. We use an exponentially distributed random variable with mean equal to the correlation level to decide this fraction for each itemset. The remaining items are picked at random. In the datasets used in the experiments, the correlation level was set to 0.5. Each itemset in F has an associated

weight that determines the probability that this itemset will be picked. The weight is picked from an exponential distribution with mean equal to 1. The weights are normalized such that the sum of all weights equals 1.

For example, suppose the number of large itemsets is 5. According to the exponential distribution with mean equal to 1, the probabilities for those 5 itemsets with ID equal to 1, 2, 3, 4 and 5 are 0.43, 0.26, 0.16, 0.1 and 0.05, respectively, after the normalization process. These probabilities are then accumulated such that each size falls in a range. For each transaction, we generate a random real number which is between 0 and 1 to determine the ID of the potentially large itemset. To model the phenomenon that all the items in a large itemset are not always bought together, we assign each itemset in F a corruption level c. When adding an itemset to a transaction, we keep dropping an item from the itemset as long as a uniformly distributed random number (between 0 and 1) is less than c. The corruption level for an itemset is fixed and is obtained from a normal distribution with mean = 0.5 and variance = 0.1. Each transaction is stored in a file system with the form of <transaction identifier, item>.

Some different data sets were used for performance comparison. Table 2 shows the names and parameter settings for each data set. For all data sets, N was set to 1,000 and |L| was set to 2,000.

Table 2: Parameters values for synthetic dataset

Case	Name	|T|	|MT|	|I|	|MI|	|D|
1	T5.I2.D20K	5	10	2	4	20K
2	T10.I6.D50K	10	15	6	10	50K

4.2 Simulation Results

In our simulation result, the performance measures which we concern about are the CPU execution time of the computation and the sensitivity to parameters.

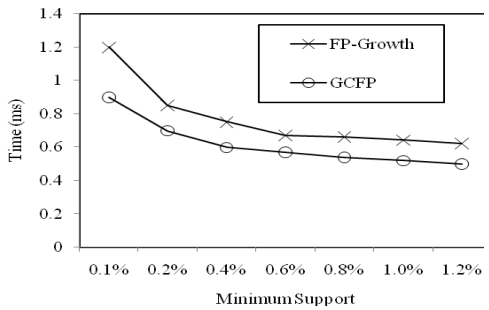

Figure 4 The relationship between the execution time (seconds) and the minimal support (Case1)

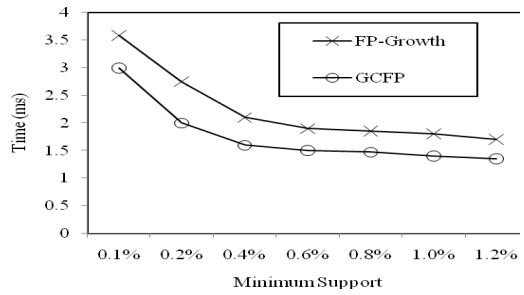

Figure 5 The relationship between the execution time (seconds) and the minimal support (Case2)

From the above comparisons, we can see that our algorithm is always faster than the FP-Growth algorithm.

5. Conclusion

In this paper, we have proposed a graphic-based algorithm for mining large itemsets. In this Section, we give a summary of this paper and point out some future research directions. We have presented the proposed graphic-based algorithm for mining closed large itemsets. We have made a comparison of execution time between the proposed algorithm and the FP-Growth algorithm by simulation. We have conducted several experiments using different synthetic datasets. The simulation results have shown that our algorithm could provide better performance than the FP-Growth algorithm in terms of the processing time.

Acknowledgement

This research was supported in part by the National Science Council of Republic of China under Grant No. NSC-93-2213-E-110-003. The authors also like to thank ``Aim for Top University Plan" project of NSYSU and Ministry of Education, Taiwan, for partially supporting the research.

Reference

Agrawal, R. and Srikant, R. (1994) 'Fast Algorithms for Mining Association Rules', *Proc. of the 20th Int. Conf. on Very Large Data Bases*, pp. 487--499.

Agrawal, R. and Srikant, R. (1995) 'Mining Sequential Patterns', *Proc. of the 11th Int. Conf. on Data Engineering*, pp. 3--14.

Han, J., Cheng, H., Xin, D., and Yan, X. (2007) 'Frequent Pattern Mining: Current Status and Future Directions', *Data Mining and Knowledge Discovery*, Vol. 15, No. 1, pp. 55--86.

Huang, L. W. and Chang, Y. I. (2009) 'An Efficient Graph-Based Approach to Mining Association Rules for Large Databases', *International Journal of Intelligent Information and Database Systems*, Vol. 3, No. 3, pp. 274-259.

Kamath, C. (2001) 'The Role of Parallel and Distributed Processing in Data Mining', Tech. Rep. UCRL-JC-142468, *Newsletter of the IEEE Technical Committee on Distributed Processing*.

Lee, W. P. (2007) 'Introduction to Data Mining'. http://www.datamining.org.tw/ dl/checkload1.asp? data_no_68.

Lucchese, C., Orlando, S., and Perego, R. (2006) 'Fast and Memory Efficient Mining of Frequent Closed Itemsets', *IEEE Trans. on Knowledge and Data Engineering*, Vol. 18, No. 1, pp. 21--36.

Ohsawa, Y. and Yada, K. (2009) 'Data Mining for Design and Marketing', Chapman & Hall/CRC Data Mining and Knowledge Discovery Series.

Srikant, R. and Agrawal, R. (1996) 'Mining Sequential Patterns: Generalizations and Performance Improvements', *Proc. of the 5th Int. Conf. on Extending Database Technology*, pp. 3--17.

Zaki, M. J. (1998) 'fficient Enumeration of Frequent Sequences', *Proc. of the 7th Int. Conf. on Information and Knowledge Management*, pp. 68--75.