

FP-tax : Tree Structure Based Generalized Association Rule Mining

Iko Pramudiono^{*}
Institute of Industrial Science
The University of Tokyo
4-6-1 Komaba, Meguro-ku
Tokyo 153-8505, Japan

iko.pramudiono@lab.ntt.co.jp

Masaru Kitsuregawa
Institute of Industrial Science
The University of Tokyo
4-6-1 Komaba, Meguro-ku
Tokyo 153-8505, Japan

kitsure@tkl.iis.u-tokyo.ac.jp

ABSTRACT

Data mining has been widely recognized as a powerful tool to explore added value from large-scale databases. One of data mining techniques, generalized association rule mining with taxonomy, is potential to discover more useful knowledge than ordinary flat association rule mining by taking application specific information into account. We propose *pattern growth* mining paradigm based FP-tax algorithm, which employs a tree structure to compress the database. Two methods to traverse the tree structure are examined : Bottom-Up and Top-Down. Experimental results show that both methods significantly outperform classic Cumulate algorithm, in particular Top-Down FP-tax can achieve two order of magnitudes better performance than Cumulate.

Keywords

data mining, generalized association rule

1. INTRODUCTION

One popular method of data mining is association rule mining [1]. This mining that is also known as “basket data analysis” retrieves information like “90% of the customers who buy A and B also buy C” from transaction data. The association rule can be generalized [8]. In generalized association rules, application-specific knowledge in the form of taxonomies (*is-a* hierarchies) over items are used to discover more interesting rules.

There are only a few researches on algorithms to mine generalized association rules [8, 4, 10]. Some SQL queries to mine generalized association rules were proposed [6, 5]. Performance evaluations on parallel engines were also reported [5, 7].

All known algorithms are based on Apriori-like level-wise approach [2]. Recently a paradigm has become a new trend in the field of frequent pattern mining research. The paradigm is often

^{*}Currently at NTT Information Sharing Platform Laboratories, NTT Corporation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DMKD '04, June 13, 2004, Paris, France

Copyright 2004 ACM ISBN 1-58113-908-X/04/06 ...\$5.00.

called *pattern growth* or *divide-and-conquer*. The main ideas of the paradigm are the projection of database into a compact on-memory data structure and then it uses a divide-and-conquer method to extract frequent patterns from the data structure. One of the most successful pioneering algorithms is FP-growth [3]. FP-growth devises a data structure called FP-tree that collects all information required to mine frequent patterns.

Here we propose two variants of algorithm FP-tax to mine generalized association rules which based on the pattern growth paradigm. Both algorithms use a tree structure similar with FP-tree. The algorithms differ on the methods to traverse the tree. A bottom-up traversal is employed by Bottom-Up FP-tax, and a top-down traversal is employed by Top-Down FP-tax.

2. MINING GENERALIZED ASSOCIATION RULE WITH TAXONOMY

2.1 Association Rule Mining

A typical example of association rule is “if a customer buys A and B then 90% of this kind of customers buy also C ”. Here 90% is called the *confidence* of the rule. Another measure of a rule is called the *support* of the rule.

Transactions in a retail database usually consist of an identifier and a set of items or itemset. $\{A, B, C\}$ in above example is an itemset. An association rule is an implication of the form $X \implies Y$ where X and Y are itemsets. An itemset X has support s if $s\%$ of transactions contain that itemset, here we denote $s = support(X)$. The support of the rule $X \implies Y$ is $support(X \cup Y)$. The *confidence* of that rule can be written as the ratio $support(X \cup Y)/support(X)$.

The problem of mining association rules is to find all the rules that satisfy a user-specified minimum support and minimum confidence, which can be decomposed into two subproblems:

1. Find all combinations of items, called large itemsets, whose support is greater than minimum support.
2. Use the large itemsets to generate the rules.

Since the first step consumes most of processing time, development of mining algorithms has been concentrated on this step.

2.2 Generalized Association Rule with Taxonomy

In most cases, items can be classified according to some kind of “is a” hierarchies. [8] For example “Sushi is a Japanese Food” and

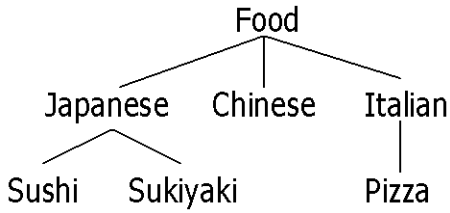


Figure 1: Taxonomy example

Table 1: Example of TAXONOMY table

DESC	ANC
Sushi	Japanese Food
Sushi	Food
Sukiyaki	Japanese Food
Sukiyaki	Food
Pizza	Italian Food
Pizza	Food
Japanese Food	Food
Chinese Food	Food
Italian Food	Food

also “Sushi is a Food” can be expressed as taxonomy as showed in figure 1. Here we categorize sushi as descendant and Japanese food and food are its ancestors. This tree can be implemented as a taxonomy table such as shown in the same figure. By including taxonomy as application specific knowledge more interesting rules can be discovered.

Since support counting for each itemset must also includes the combinations of all ancestors for each item, generally this kind of mining requires significantly more time.

Previous works on generalized association rule mining put great efforts to prune the combinations of items which are unlikely to become frequent [8, 4, 10]. Cumulate algorithm is based on Apriori-like algorithm with three optimizations such as filtering the ancestors added to the transactions, pre-computing the ancestors and pruning itemsets containing an item and its ancestors [8].

3. TREE BASED GENERALIZED ASSOCIATION RULE MINING ALGORITHMS

Our algorithm employs a tree structure called FP-tree [3], but the FP-tree also includes the taxonomy information. We propose two methods to traverse the FP-tree in order to extract the frequent patterns which will be used to generate the generalized association rules.

3.1 Construction of the tree structure

The construction of the FP-tree requires two scans of the transaction database. The first scan accumulates the support of each item and then selects items that satisfy minimum support, i.e. frequent 1-itemsets. The supports of the ancestors of each item are also accumulated. Those items are sorted in frequency descending order to form *F-list*. The second scan constructs the FP-tree.

The pseudo code for the construction of the FP-tree is given in Figure 2. First, the ancestors of each item in the transaction is added. Then the transactions are reordered according to the *F-list*, while non-frequent items are stripped off. Lastly, the re-ordered transactions are inserted into the FP-tree. In the function *insert_fptree*, if the node corresponding to the items in transaction

```

construct_fptree(database D, flist FList)
input : database D, F-list FList;
output : FP-tree FPtree;
{
1:while not eof(D) do
2: tranline = read_trans(D);
3: begin
4: add all ancestors of each item in tranline
5: removing any duplicates in tranline
6: end
7: o_trans = get_ordered_trans(Flist, tranline);
8: insert_fptree(FPtree, o_trans);
9:end while
}
  
```

Figure 2: Pseudo code of FP-tree construction

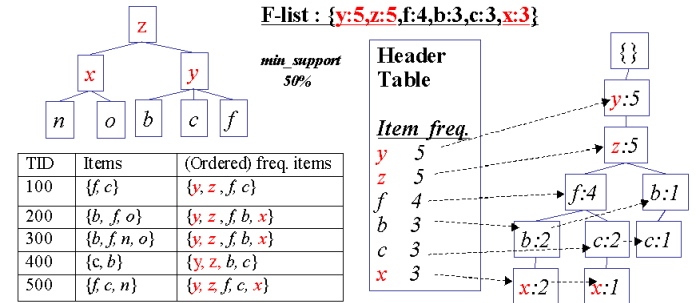


Figure 3: Example of FP-tree construction

exists the count of the node is increased, otherwise a new node is generated and the count is set to one. The same order of the items plays important role for the compression of the database since common prefixes can be shared among many transactions.

The FP-tree also has a frequent-item header table that holds the head of the node-links, which connect nodes of same item in FP-tree. The node-links facilitate item traversal during the mining of frequent patterns.

An example of the construction of taxonomy concious FP-tree is given in Figure 3. The two level taxonomy with ancestors *x, y, z* is depicted on the upper left of the figure, and the transaction database is given below it. The minimum support is set to three (50%). After the first pass, the *F-list* is determined. Notice that ancestor *x* is also included in the *F-list* although its descendants are infrequent items.

After the transactions are reordered, they are inserted one by one into the FP-tree. The final FP-tree is depicted on the right side of the Figure 3.

3.2 Bottom-up traversal method

The pseudo code for the recursive function Bottom-Up FP-tax (BU-FPtax) is given in Figure 4. Inputs to the BU-FPtax algorithm are the FP-tree, the minimum support, and a list of ancestors which have been investigated so far. To find all frequent patterns whose support are higher than minimum support, BU-FPtax adopts the same methodology with FP-growth called *conditional-search*, which looks for all patterns with the same suffix once at a time [3].

BU-FPtax traverses nodes in the FP-tree starting from the least frequent item in *F-list*. While visiting each node, BU-FPtax also collects the prefix-path of the node, which is the set of items on the path from the suffix node to the root of the tree. BU-FPtax also stores the count on the node as the count of the prefix path. The prefix paths form the so-called *conditional pattern base* of that

```

procedure BU-FPtax(FPtree, X, anclist);
input : FP-tree Tree, itemset X, itemset anclist;
{
1:for each item y (bottom-up order)
   in the header of FPtree do
2: if(y is in anclist) then continue;
   //Filtering 1
3: if(ancestors of y is in Y) then continue;
   //Filtering 2
4: generate pattern Y = y U X with
   support = y.support;
5: begin
6:   add ancestors of y to anclist;
7: end
8: cond_pbase = construct_cond_pbase(Tree,y);
9: Y-FList = sort_cond_pbase(cond_pbase);
10: Y-Tree = construct_fptree(cond_pbase,Y-FList);
11: if (Y-Tree is not NULL) then
12:   BU-FPtax(Y-Tree, Y, anclist);
13: end if
14:end for
}

```

Figure 4: Pseudo code for Bottom-Up FP-tax

item.

The conditional pattern base is a small database of frequent patterns that co-occur with the item. Then BU-FPtax creates small FP-tree from the conditional pattern base called *conditional FP-tree*. During each iteration, a new frequent itemset is generated by adding the suffix to the itemset from the previous iteration. BU-FPtax also maintains a list of ancestors *anclist* for the items in the current itemset.

Here we propose two kind of filtering methods utilizing the characteristics of ancestor-descendant relation to reduce the search space. The theoretical foundation for the optimizations is given in the Cumulate paper [8], which states the following observations :

1. We do not need to count any itemset which contains both an item and its ancestor because the support is contained in the ancestor's support.
2. Pruning of such itemset is sufficient to ensure that we never generate itemsets in subsequent iterations which contain both an item and its ancestor.

Two kinds of filtering methods are needed to completely remove itemsets that contain both an item and its ancestors :

1. Pruning itemset whose item is already in the anclist

The anclist contains all ancestors examined so far for the current suffix. Thus it is sufficient to check the membership of the anclist against the item to make sure that no ancestors of the item in the itemset.
2. Pruning items whose ancestors is already included in the itemset

The second filtering is needed in order to prune the descendants of the items in the itemset. The first filtering only checks the ancestors of the item. It is also obvious that direct descendants of items in the itemset is not needed.

When BU-FPtax encounters such conditions, the iteration is stopped and BU-FPtax processes the next item.

The process is recursively iterated until no conditional pattern base can be generated and all frequent patterns that contain the item are discovered.

```

procedure TD-FPtax(H, X, anclist);
input : header_table H, itemset X,
      itemset anclist;
{
1:for each item y (top-down order) in H do
2: if(y.support >= minsupp) then
3:   if(y is in anclist) then continue;
   //Filtering 1
4:   if(ancestors of y is in Y) then continue;
   //Filtering 2
5:   generate pattern Y = y U X with
   support = y.support;
6:   begin
7:     add ancestors of y to anclist;
8:   end
9:   H_y = create_new_header_table(H, y);
10:  reconnect_node_links(H_y);
11:  if(H_y is not NULL) then
12:    BU-FPtax(H_y, Y, anclist);
13:  end if
14: end if
15:end for
}

```

Figure 5: Pseudo code for Top-Down FP-tax

3.3 Top-down traversal method

The top-down algorithm Top-Down FP-tax is inspired by algorithm Top-Down FP-growth [9]. The advantage of this method is no need to construct the conditional pattern bases and the subtrees. The filtering methods similar with Bottom-Up FP-tax are also employed to avoid the unnecessary traversals of items which occur with their ancestors.

The recursive function TD-FPtax is depicted in Figure 5. TD-FPtax eliminates the generation of conditional pattern bases by directly reconnecting the node links of the FP-tree. During the reconnection, TD-FPtax also modifies the count information in the FP-tree nodes. For each new suffix extension y , a new header table H_y is created. The header table contains items prior to the item y in the previous header table. Then following the node-links of y , the prefix paths to the root node are traversed. During the traversal, the node links of the FP-tree nodes are reconnected to header table H_y . The counts in header table and in the FP-tree node are also modified relative to its co-occurrence with y . The process is recursively iterated.

4. PERFORMANCE EVALUATION

The algorithms of Bottom-Up FP-tax, Top-Down FP-tax and Cumulate is implemented in C. They are executed sequentially on a Sun Fire 4800 with four 900MHz CPUs. The total main memory is 16GB and the operating system is Solaris 8.

The synthetic transaction data generator developed at IBM Almaden is used to prepare the datasets [2]. Here we report the experiment results on two datasets called R50F20T25I20 and R250F5T10I4 with parameters described in table 2.

The execution time of each algorithm with several minimum supports are given in Figure 6 and Figure 7. Notice that logarithmic scale is used for the execution time. The experiment results clearly show that our proposed algorithms can outperform Cumulate significantly. In particular, Top-Down FP-tax can deliver two order of magnitudes better performance even with low minimum support for denser R50F20T25I20 dataset. When the minimum support was set to 2%, the execution of Cumulate and Bottom-Up FP-tax were aborted because they did not finish after 24 hours.

	R50F20T25I20	R250F5T10I4
Size of transaction data	145MB	61MB
Number of transactions	1000000	1000000
Avg. transaction length	25	10
Avg. pattern length	20	4
Size of taxonomy	1MB	1MB
Number of items	100000	100000
Number of roots	50	250
Average fanout	20	5

Table 2: Dataset parameters

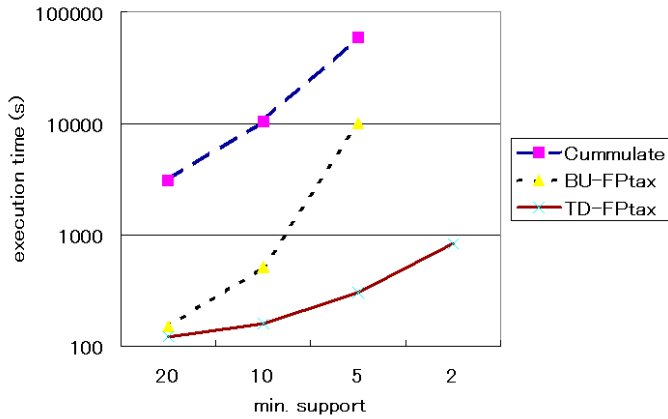


Figure 6: Performance comparison(R50F20T25I20)

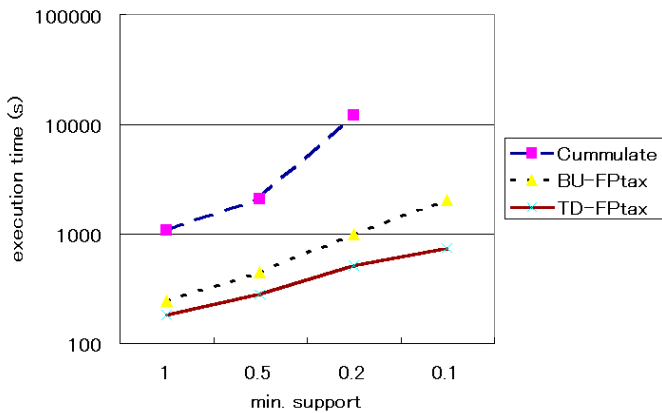


Figure 7: Performance comparison(R250F5T10I4)

When the minimum support is low, the FP-tree is becoming more bushy so that Bottom-Up FP-tax needs more time to traverse the tree. However since the ancestors are likely more frequent, they tend to dominate the upper part of the FP-tree. Thus Top-Down FP-tax is expected to benefit more from the proposed filtering methods.

Here we only perform comparison with Cumulate. However other algorithms such as Prutax are reported to deliver at most one order of magnitudes faster than Cumulate [4].

5. SUMMARY AND CONCLUSION

Generalized association rule mining with taxonomy is one of the complicated mining task that requires long processing time. As far as authors know only Apriori based algorithms are available to perform data mining on generalized association rule.

We proposed two FP-tree based algorithms namely Bottom-Up FP-tax and Top-Down FP-tax. The algorithms employ two kinds of filter to remove itemsets containing item and its ancestors. The distribution of ancestors in the FP-tree is utilized better by the Top-Down FP-tax so that it can record more than 200 times better performance than Cumulate algorithm.

In frequent pattern mining research, it is well known that some characteristics of the datasets such as the denseness can affect the choice of mining algorithm. We are going to investigate the efficiency of the proposed algorithms against more datasets. In particular, we are interested in how certain characteristics such as the distribution of the ancestors in the FP-tree can affect the performance.

6. REFERENCES

- [1] R. Agrawal, T. Imielinski, A. Swami. "Mining Association Rules between Sets of Items in Large Databases". In *Proc. of the ACM SIGMOD Conf. on Management of Data*, 1993.
- [2] R. Agrawal, R. Srikant. "Fast Algorithms for Mining Association Rules". In *Proc. of the VLDB Conference*, 1994.
- [3] J. Han, J. Pei and Y. Yin "Mining Frequent Pattern without Candidate Generation". In *Proc. of the ACM SIGMOD Conf. on Management of Data*, 2000.
- [4] J. Hipp, A. Myka, R. Wirth, and U. Guntzer. "A New Algorithm for Faster Mining of Generalized Association Rules" In *Proc. of second PKDD Conference*, 1998.
- [5] P. Iko, T. Shintani, T. Tamura, and M. Kitsuregawa. "Mining Generalized Association Rule using Parallel RDB Engine on PC Cluster". In *Proc. of First Int. Conf. on Data Warehousing and Knowledge Discovery (DAWAK)*, 1999.
- [6] S. Sarawagi, S. Thomas. "Mining Generalized Association Rules and Sequential Patterns Using SQL Queries". In *Proc. of KDD Conference*, 1998.
- [7] T. Shintani, M. Kitsuregawa "Parallel Mining Algorithms for Generalized Association Rules with Classification Hierarchy." In *Proc. of the ACM SIGMOD Conf. on Management of Data*, 1998.
- [8] R. Srikant, R. Agrawal. "Mining Generalized Association Rules". In *Proc. of VLDB Conference*, 1995.
- [9] K. Wang, L. Tang, J. Han, and J. Liu. "Top down FP-Growth for association rule mining". In *Proc. of the 6th Pacific Area Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2002
- [10] I. Weber "On Pruning Strategies for Discovery of Generalized and Quantitative Association Rules". In *Proc. of Knowledge Discovery and Data Mining Workshop, (Pricai'98)*, 1998.