Faster Prediction of Missing Items in Shopping Carts using FUFP-DSARM

Anagha Patil
M.E. Student,
T.S.E.C,
Mumbai University,

Thirumahal R.
A.P., Dept of IT,
T.S.E.C,
Mumbai University,

ABSTRACT

Association mining techniques search for groups of frequently co-occurring items in a market-basket type of data and turn this data into rules. Previous research has focused on how to obtain list of these associations and use these “frequent item sets” for prediction purpose. This paper proposes a technique which uses partial information about the contents of the shopping carts for the prediction of what else the customer is likely to buy. Using Fast Updated Frequent Pattern Tree (FUFP-Tree) instead of Item set Trees (IT-Tree) and Frequent Pattern Tree (FP-Tree), all the rules whose antecedents contain at least one item from the incomplete shopping cart can be obtained in efficient manner. Rules are then combined and Prediction is done using Bayesian Decision Theory and DS-ARM algorithm based on the Dempster-Shafter theory of evidence combination.

Keywords

Frequent item sets, Market Baskets, Dempster-Shafter theory, IT-Tree, FP-Tree, FUFP-Tree.

1. INTRODUCTION

Association mining detects frequently co-occurring groups of items in transactional database. Association Mining systems have been developed for analyzing market baskets. The primary task of Market Basket Analysis is a modeling technique based upon the theory that if a customer buys (as registered at the checkout desk) a certain group of items, he is more (or less) likely to buy another set(s) of item(s). The intention is to use this knowledge for prediction purpose: if bread, butter, and milk often appear in the same transactions, then the presence of butter and milk in a shopping cart suggest that the customer may also buy bread. More generally, knowing which items a shopping cart contains, we want to predict other items that the customer is likely to add before proceeding to the checkout counter.
Now-a-days number of supermarkets is emerging worldwide. People enjoy buying products in supermarkets because they get each and every type of product at reasonable price. We are living in information age. The better we make use of customer purchasing behavior, better will be profit and sales to the supermarket. Knowing what products people purchase as a group can be very helpful to a retailer or to any other company. A store could use this information to place products frequently sold together into the same area, while a catalog or World Wide Web merchant could use it to determine the layout of their catalog and order form. Direct marketers could use the basket analysis results to determine what new products to offer their prior customers. Most of the methods of classification rule mining were designed for data sets with limited number of attributes and one class label. Here, we do not have a predefined class label. In fact, all items in the shopping cart become attributes and the presence/absence of the other items has to be predicted. We have a feasible rule generation algorithm and an effective method to use to this end the generated rules. For the prediction of all missing items in a shopping cart, our algorithm speeds up the computation by the use of the FUFP trees and then uses DS theoretic notions to combine the generated rules.

2. EARLIER WORK

Association rule mining (ARM) in its original form finds all the rules that satisfy the minimum support and minimum confidence constraints. Many papers tried to integrate classification and ARM. The prediction task was mentioned as early as in the pioneering association mining paper by Agrawal et al. [8], but the problem is yet to be investigated in the depth it deserves. The literature survey indicates that most authors have focused on methods to expedite the search for frequent item sets, while others have investigated such special aspects as the search for time-varying associations of localized patterns [6]. Still, some prediction-related work has been done as well. An early attempt by Bayardo and Agrawal [7] reports a method to convert frequent item sets to rules. Some papers then suggest that a selected item can be treated as a binary class (absence = 0; presence = 1) whose value can be predicted by such rules. All those methods and extensions have advantages and disadvantages in both theory and practical applications. Existing research in association mining has focused mainly on how to expedite the search for frequently co-occurring groups of items in “shopping cart” type of transactions; less attention has been paid to methods that exploit these “frequent itemsets” for prediction purposes.
The approach proposed in paper Itemset Trees for Targeted Association Querying” [2] says that If i_j is the item whose absence or presence is to be predicted, the technique can be used to generate all rules that have the form $r^{(a)} {}_{=>}$ Ij, where $r^{(a)} {}_{\epsilon}$ (I\ { i_j }) and Ij is the binary class label (i_j = present or i_j = absent). For a given item set s, the technique identifies among the rules with antecedents subsumed by those that have the highest precedence according to the reliability of the rules (low confidence and low support values).The rule is then used for the prediction of i_j. The method suffers from three shortcomings. First, it is clearly not suitable in domains with many distinct items ij. Second, the consequent is predicted based on the “testimony” of a single rule, ignoring the simple fact that rules with the same antecedent can imply different consequents—a method to combine these rules is needed. In paper presented by J. Zhang, Subasingha, K. Premaratne, Shyu, Kubat, and Hewawasam [4], a missing item is predicted in four steps. First, they use a partitioned-ARM to generate a set of association rules

(a rule set). The next step prunes the rule set (e.g., by removing redundant rules). From these, rules with the smallest distance from the observed incomplete shopping cart are selected. Finally, the items predicted by these rules are weighed by the rules' antecedents' similarity to the shopping cart. The approach in Rule Mining and Classification in a Situation Assessment Application: A Belief Theoretic Approach for Handling Data Imperfections" [3] uses a Dempster-Shafer (DS) belief theoretic approach that accommodates general data imperfections. Here, authors employ a data structure called a belief item set tree. In order to predict the missing item, the technique selects a "matching" rule set i.e. a rule is included in the matching rule set if the incoming item set is contained in rule antecedent. If no rules satisfy this condition, then, from those rules that have nonempty intersection with the item set s, rules whose antecedents are "closer" to s according to a given distance criterion (and a user-defined distance threshold) are picked. Confidence of the rule, its "entropy," and the length of its antecedent are used to assign DS theoretic parameters to the rule. Finally, the evidence contained in each rule belonging to the matching rule set is combined or "pooled" via a DS theoretic fusion technique.

The objective of our work is to propose a technique that uses partial information about the contents of shopping cart for the prediction of what else the customer is likely to buy. Using the data structure of FP trees, we obtain all rules whose antecedents contain at least one item from the incomplete shopping cart. Then, we combine these rules by uncertainty processing techniques, including the classical Bayesian decision theory and a new algorithm based on the Dempster-Shafer (DS) theory of evidence combination. This application will help to learn more about customer behaviour and to find out which products perform similarly to each other. Predicting missing items in shopping cart will help to satisfy customer, as all the products which they may be wanted to buy and are missed will be predicted. Study and implementation of efficient algorithms like Generation and Updating of FP-Trees, Rule Generation algorithm, Dempster-Shafter Theory and Bayesian Approach will be done.

3. PROBLEM STATEMENT

Given a record with itemset $s \subseteq I$, find all rules of the form $r^{(a)} \Rightarrow i_j$; $j \in [1, n]$, where s and $r^{(a)} \subseteq s$ and $i_j \not\subset s$, that exceed minimum support and minimum confidence thresholds. The consequent i_j is a single "unseen" item, i.e., $|i_j| = 1$. Also, the prediction of the consequent could be {item = present} or {item = Absent}. For each unseen item, the corresponding rule set is selected and a DS theoretic approach is used to combine the rules.

3.1 It Tree

IT Tree organizes the data in the market-basket type of database in a manner that facilitate access to stored information. An algorithm for construction of IT Tree is described in [1]. An Item set Tree, T, consists of a root and a (possibly empty) set, $\{T1 \ldots\ldots, T_k\}$, each element of which is an item set tree. Here, each item is identified by an unique integer and item set is an uninterrupted sequence of integers $\{i_1 \ldots, i_n\}$ such that $i_k < i_j$ where i_k and i_j are integers identifying the kth and jth terms respectively. An IT Tree is a partially ordered set of pairs, [item set, f] where f is frequency of occurrence of the item set the node represents. Some of the item sets in IT Tree (e.g. [1,2,4] in fig.1) are identical to at least one of the transactions contained in the original database, whereas others (e.g. [1,2]) are created during the process of tree building. The nodes which are identical to at least one transaction are flagged, indicated by black dots. This flagged IT Tree becomes the base of rule generation algorithm [1].

An algorithm that builds IT Tree in a single pass through the database proves some of the critical properties of itself. For example, the numbers of the nodes in IT Tree are upper bounded by twice the number of the transactions in the original database. The size of IT Tree exceeds with the size of the database. Moreover, each distinct transaction is represented by a unique IT Tree and the original transaction can be then reproduced from the IT Tree.

Example1 (An IT Tree) The flagged IT Tree of the database D ={[1,4], [2,5], [1,2,3,4,5], [1,2,4], [2,5], [2,4]} is shown in fig. 1.

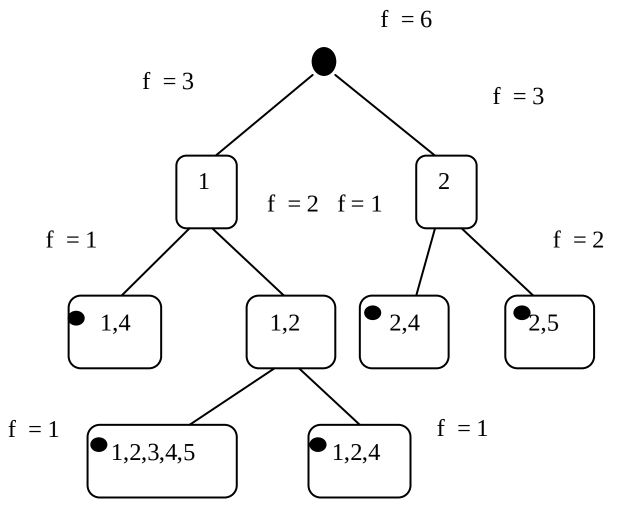

Figure 1. IT Tree

3.2 FP Tree

The FP-tree (Han et al., 2000) is used to compress a database into a tree structure storing only large items. It is condensed and complete for finding all the frequent patterns. Three steps are involved in FP-tree construction. The database is first scanned to find all items with their frequency. The items with their supports larger than a predefined minimum support are selected as large 1-itemsets (items). Next, the large items are sorted in descending frequency. At last, the database is scanned again to construct the FP-tree according to the sorted order of large items. The construction process is executed tuple by tuple, from the first transaction to the last one. After all transactions are processed, the FP-tree is completely constructed. The Header_Table is also built to help tree traversal. The Header_Table includes the sorted large items and their pointers (called frequency head) linking to their first occurrence nodes in the FP-tree. If more than one node have the same item name, they are also linked in sequence. Note that the links between nodes are single-directed from parents to children.

Example 2 (FP Tree)

Table 1 A database with five transactions

TID	Items
100	a, c, d, f, g, I, m, p
200	a, b, c, f, l, m, o
300	b, f, h, j, o
400	b, c, k, s, p
500	a, c, e, f, l, m, n, p

Table 2 All the items with their counts

Item	Frequency	Item	Frequency
a	3	j	1
b	3	k	1
c	4	l	2
d	1	m	3
e	1	n	1
f	4	o	2
g	1	p	3
h	1	s	1
I	1	w	1

Once the FP-tree is constructed from a database, a mining procedure called FP-Growth (Han et al., 2000) is executed to find all large item sets. FP-Growth derives frequent patterns directly from the FP-tree without candidate generation.

Table 3 The transactions with only sorted large items

TID	Items
100	f, c, a, m, p
200	f, c, a, b, m
300	f, b
400	c, b, p
500	f, c, a, m, p

It is a recursive process, handling the frequent items one by one and bottom- up according to the Header_Table. A conditional FP-tree is generated for each frequent item, and from the tree the large item sets with the processed item can be recursively derived. The process for generating large itemsets from the FP-tree is much faster than the Apriori algorithm. When new transactions come, the FP-Tree mining algorithm must re-process the entire updated databases to form the correct FP-tree. We will not generate conditional FP-Tree here as our main aim is prediction, conditional FP-Tree will remove the items which are below or equal to minimum count, so it is possible that no item is remaining with the incoming item(s) for prediction after applying minimum count value.

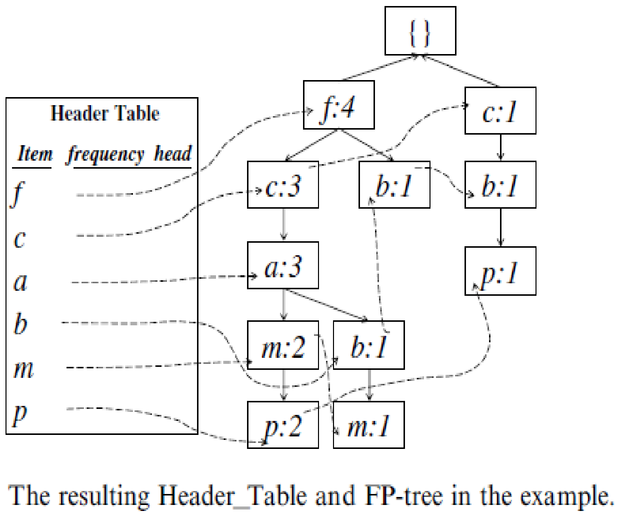

Figure 2. FP Tree

3.3 FUFP Tree

A fast updated FP-tree (FUFP-tree) makes the tree update easier. It is similar to the FP-tree structure except that the links between parent nodes and their child nodes are bi-directional. Besides, the counts of the sorted frequent items are also kept in the Header_Table of the FP-tree algorithm. Bi- directional linking and storing the counts in the Header_Table will help fasten the maintenance process. An incremental FUFP-tree maintenance algorithm processes newly inserted transactions. It first partitions items into four parts according to whether they are large or small in the original database and in the new transactions. Each part is then processed in its own way. The Header_Table and the FUFP-tree are correspondingly updated whenever necessary.. An FUFP-tree must be built in advance from the original database before new transactions come. The FUFP tree construction algorithm is the same as the FP-tree algorithm (Han et al., 2000) except that the links between parent nodes and their child nodes are bi-directional. Bi-directional linking will help fasten the process of item deletion in the maintenance process.
When new transactions are added, the incremental maintenance algorithm [5] processes them to maintain the FUFP-tree. It first partitions items into four parts according to whether they are large or small in the original database and in the new transactions. Each part is then processed in its own way. The Header_Table and the FUFP-tree are correspondingly updated whenever necessary.The entire FUFP-tree can be re-constructed in a batch way when a sufficiently large number of transactions are inserted is shown in [5].

Example 3 (FUFP Tree)

Table 4 A database with ten transactions

TID	Items
1	a, b, c, d, e, g, h
2	a, b, f, g
3	b, d, e, f, g
4	a, b, f, h
5	a, b, f, i
6	a, c, d, e, g, h
7	a, b, h, i
8	b, c, d, f, g
9	a, b, f, h
10	a, b, g, h

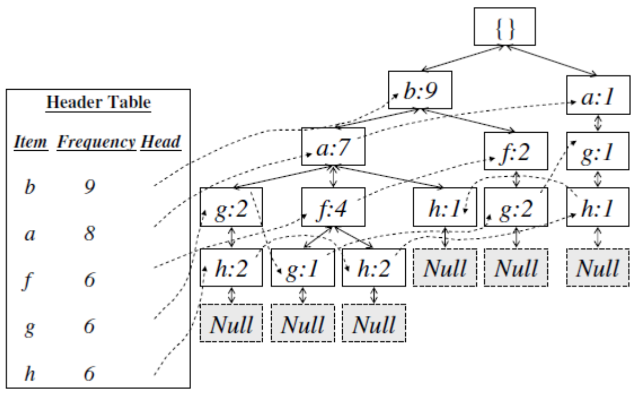

Figure 3. FUFP-Tree

4. RULE GENERATION MECHANISM

To generate association rules from any Tree structure we will use rule Generation Algorithm explained in [1]. The algorithm takes an incoming item set as the input, compares with nodes in Tree and returns a graph that defines the association rules entailed by the given incoming item set. The graph consists of two lists: the antecedents list $R^{(a)}$ and the consequents list $R^{(c)}$. Each node, $r_i^{(a)}$, in the antecedents list keeps the corresponding frequency count $f(r_i^{(a)})_j$. As shown in Fig.4, a line, $l_{i,j}$, between the two lists links an antecedent $r_i^{(a)}$ with a consequent Ij. The cardinality of the link, $f(l_{i,j})$, represents the support count of the rule $r_i^{(a)}$ => Ij. The frequency counts denoted by $f_o(\bullet)$ are used in the process of building the graph. If the incoming itemset is s and if Ti represents a transaction in the database, then $f_o(r^{(a)})$ records the number of times s ∩ Ti = $r^{(a)}$. Thus, $f_o(r^{(a)})$ records the number of times where s ∩ Ti = $r^{(a)}$ and I_j Є Ti. All the frequency counts are initialized to zero at the beginning of the algorithm and updated as we traverse the Tree.

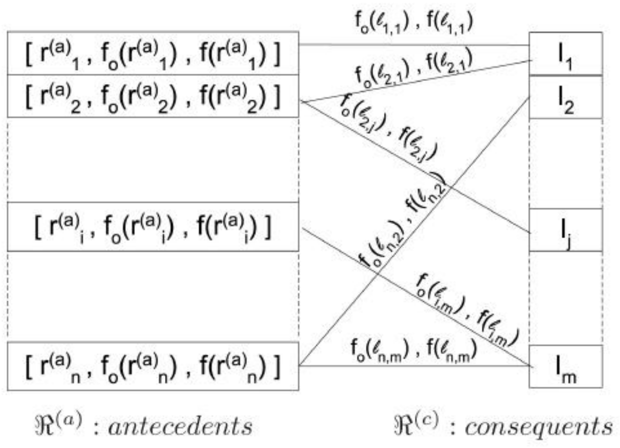
$[r^{(a)}_1, f_o(r^{(a)}_1), f(r^{(a)}_1)]$
$[r^{(a)}_2, f_o(r^{(a)}_2), f(r^{(a)}_2)]$
$[r^{(a)}_i, f_o(r^{(a)}_i), f(r^{(a)}_i)]$
$[r^{(a)}_n, f_o(r^{(a)}_n), f(r^{(a)}_n)]$
$f_o(\ell_{1,1}), f(\ell_{1,1})$
$f_o(\ell_{2,1}), f(\ell_{2,1})$
$f_o(\ell_{2,j}), f(\ell_{2,j})$
$f_o(\ell_{n,2}), f(\ell_{n,2})$
$f_o(\ell_{i,m}), f(\ell_{i,m})$
$f_o(\ell_{n,m}), f(\ell_{n,m})$
I_1
I_2
I_j
I_m
$\Re^{(a)}$: antecedents
$\Re^{(c)}$: consequents

Figure 4. Rule Graph

5. EMPLOYING DEMPSTER-SHAFTER THEORY

Many rules with equal antecedents differ in their consequents—some of these consequents contain i_j, others do not. To combine them DS-ARM[1] is used. We select only rules that exceed the minimum support and the minimum confidence in the rule combination step. In addition, if two rules with the same consequent have overlapping antecedents such that the antecedent of one rule is a subset of the antecedent of the other rule (e.g.,[a => c], [a, b => c]), we only consider the rule with the higher confidence.We need to calculate Basic Belief Assignment and d(discounting factor) as given in [1], to keep the rules as independent as possible. Finally we will have Table of Pruned Rule set and combined BBA. We can then remove the overlapping rules while keeping the highest confidence rule. If two overlapping rules have the same confidence, the rule with the lower support is dropped. Rules are then combined using the Dempster's rule of combination[4]. Pignistic probability can be used to pick the most probable event. According to the values in the table and using the pignistic approximation, the predictor may predict that which item will be added to the cart.

6. COMPARISION

IT Tree, FP Tree and FUFP Tree can be compared on the basis of four parameters.

Table 5 Comparing IT Tree, FP Tree and FUFP Tree

Parameter	**Structure**	**Size**	**Memory Requirement**	**Update**
IT Tree	Simple	Grows with large no. of transactions	High	Not Easy
FP Tree	Complex	Always less than no. of transactions	Low	Not Easy
FUFP Tree	Complex	Always less than no. of transactions	Low	Easy

To insert a single market basket to an existing IT Tree, algorithm in [1] must visit less than existing nodes since Flagged IT Tree is used. But with large number of market baskets, size of IT Tree grows rapidly. In case of IT Trees, every new transaction is stored individually as a new node (exception: if transaction is repeated only frequency is updated) plus new node is created for ancestor, so number of total nodes may be high than number of transactions for distinct transactions. Hence memory requirement is high. FP-Trees are more complex but give frequent patterns more effectively. Since in FP-Trees transaction is not stored individually, size of FP Tree is always less than total number of transactions even if number of transactions increases rapidly. Hence memory requirement is low. FUFP Trees are similar to that of FP Tree structure except that the links between parent nodes and their child nodes are bi-directional. So, it makes update easier than FP-Tree.

7. CONCLUSION

This paper focuses on the task of association mining i.e. provided incomplete shopping cart, can we predict which other items this cart may contain. Here the idea is to convert the database into a tree structure which will help to generate rules. This tree structure can be IT-Tree or FP-Tree or FUFP-Tree and performance depends on number of items, number of transactions. When presented an incomplete list of transactions, DS-ARM finds all high-support, high-confidence rules that have as antecedent a subset of s. then it combines the consequents of all these rules and creates a set of items most likely to complete the shopping cart.

8. REFERENCES

[1] K. Wickramaratna, M. Kubat, K. Premarathnes, “Predicting Missing Items in Shopping Carts,” *IEEE Trans. On Knowledge and Data Engineering*, Vol. 21, No. 7, July 2009.

[2] M. Kubat, A. Hafez, V.V. Raghavan, J.R. Lekkala, and W.K. Chen, “Itemset Trees for Targeted Association Querying,” *IEEE Trans. On Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1522-1534, Nov./Dec. 2003.

[3] J. Zhang, Subasingha, K. Premaratne, Shyu, M. Kubat, and Hewawasam, “A novel belief theoretic association rule mining based classifier for handling class label ambiguities,” in *Proc. FDM Workshop, IEEE ICDM*, Brighton, U.K., Nov. 2004, pp. 213–222.

[4] D. H. G. Resconi, G. J. Klir, and Y. Pan, “On the computation of the uncertainty measure for the Dempster–Shafer theory,” *Int. J. Gen. Syst.*, vol. 25, no. 2, pp. 153–163, 1996.

[5] Tzung-Pei-Hong, Chun-Wei Lin, Yu-Lung Wu, “Incrementlly Fast Updated Pattern Trees,” *Expert Systems with applications* , vol. 34, pp- 2424-2435, 2008.

[6] A. Rozsypal and M. Kubat, “Association Mining in Time-Varying Domains,” *Intelligent Data Analysis*, vol. 9, pp. 273-288, 2005.

[7] C.C. Aggarwal, C. Procopius, and P.S. Yu, “Finding Localized Associations in Market Basket Data,” *IEEE Trans. On Knowledge and Data Engineering*, vol. 14, no. 1, pp. 51-62, Jan./Feb. 2002.

[8] R. Agarwal, T. Imielinski, and A. Swami, “Mining Association Rules between Sets of Items in Large Databases,” Proc. ACM SIGMOD, pp. 207-216, 1993.