ISSN:0975-9646

A Sequential Approach for Surmising Missing Items in the Shopping Cart

Arshiya Subhani, Prof. P.Pradeep kumar
Department CSE, Vivekananda Institute of Technology and Science, Karimnagar, A.P, India

Abstract: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. This paper contributes to the latter task by proposing a technique that uses partial information about the contents of a shopping cart for the prediction of what else the customer is likely to buy. Using the recently proposed data structure of itemset trees (IT-trees), we obtain, in a computationally efficient manner, 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.

Key Words: Frequent itemsets, itemset trees (IT-Tresss), uncertainty processing, Dempster-Shafer theory.

1. INTRODUCTION

The primary task of association mining is to detect frequently co-occurring groups of items in transactional databases. The intention is to use this knowledge for prediction purposes: if bread, butter, and milk often appear in the same ransactions, then the presence of butter and milk in a shopping cart suggests 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.

This paradigm can be exploited in diverse applications For example, in the domain discussed in [1], each "shopping cart" contained a set of hyperlinks pointing to a Web page [1]; in medical applications, the shopping cart may contain a patient's symptoms, results of lab tests, and diagnoses; in a financial domain, the cart may contain companies held in the same portfolio; and Bollmann-Sdorra et al. [2] proposed a framework that employs frequent itemsets in the field of information retrieval.The prediction task was mentioned as early as in the pioneering association mining paper by Agrawal et al. [3], but the problem is yet to be investigated in the depth it deserves. The literature survey in [4] indicates that most authors have focused on methods to expedite the search for frequent itemsets, while others have investigated such special aspects as the search for time-varying associations [5], [6] or the identification of localized patterns [7]. Still, some prediction-related work has been done as well.

In our work, we wanted to make the next logical step by allowing any item to be treated as a class label—its value is to be predicted based on the presence or absence of other items. Put another way, knowing a subset of the shopping cart's contents, we want to "guess" (predict) the rest.It is important to understand that allowing any item to be treated as a class label presents serious challenges as compared with the case of just a single class label. The number of different items can be very high, perhaps hundreds, or thousand, or even more. To generate association rules for each of them separately would give rise to great many rules with two obvious consequences: first, the memory space occupied by these rules can be many times larger than the original database .second, identifying the most relevant rules and combining their sometimes conflicting predictions may easily incur prohibitive computational costs. In our work, we sought to solve both of these problems by developing a technique that answers user's queries (for shopping cart completion) in a way that is acceptable not only in terms of accuracy, but also in terms of time and space complexity.

2. PROBLEM STATEMENT

Let $I=\{i_1,\ldots\ldots i_n\}$be a set of distinct items and let a database consist of transactions T1,. . . ,TN such that Ti ⊑I; 8i. An itemset, X, is a group of items, i.e., X _ I. The support of itemset X is the number, or the percentage, of transactions that subsume X. An itemset that satisfies a user-specified minimum support value is referred to as a frequent itemset or a high support itemset.

An association rule has the form $r^{(a)} \rightarrow r^{(c)}$, where $r^{(a)}$ and $r^{(c)}$ are itemsets. The former, $r^{(a)}$, is the rule's antecedent and the latter, $r^{(c)}$, its consequent. The rule reads: if all items from $r^{(a)}$ are present in a transaction, then all items from $r^{(c)}$ are also present in the same transaction..The probabilistic confidence in the rule $r^{(a)} \rightarrow r^{(c)}$ can be defined with the help of supports (relative frequencies) of the antecedent and consequent as the percentage of transactions that contain $r^{(c)}$ among those transactions that contain $r^{(a)}$:

$$Conf = support(r^{(a)} \mathbf{U} r^{(c)})/support(r^{(a)}). \quad -(1)$$

Let s be a given itemset. An algorithm developed in [4] generates, in a computationally feasible manner, all rules $s \rightarrow l$,that satisfy the user-supplied minimum support and confidence values Θs and Θc, respectively. Of course, if no frequent item set subsumes s, no rules are generated. However, we are also interested in rules with antecedents that are subsumed by s. Furthermore, we need to be aware of the circumstance that the presence of an item might suggest the absence of other items. With all these issues in mind, we narrow down the space of association rules by the following guidelines:

1. For a given itemset s, rule antecedents should be subsumed by s.
2. The rule consequent is limited to any single "unseen" item (presence or absence of the unseen item).

3.THE PROPOSED APPROACH

Association rule mining (ARM) in its original form finds all the rules that satisfy the minimum support and minimum confidence constraints. Many later papers tried to integrate classification and ARM. The goal was to build a classifier using so-called class association rules. In classification rule mining, there is one and only one predetermined target, the class label. Most of the time, classification rule mining is applied to databases in a "table" format, with a predefined set of attributes and a class label. Attributes usually take a value out of a finite set of values (although missing values are often permitted).

Some papers, such as [8] and [9], demonstrated encouraging results by incorporating DS theoretic notions with class association rules. But most of these methods were designed for data sets with limited number of attributes (or data sets with small number of distinct items) and one class label. In our task, 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. What is needed is 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 itemset trees (IT-trees) and then uses DS theoretic notions to combine the generated rules. The flowchart in Fig. 1 shows an outline of our proposed system.

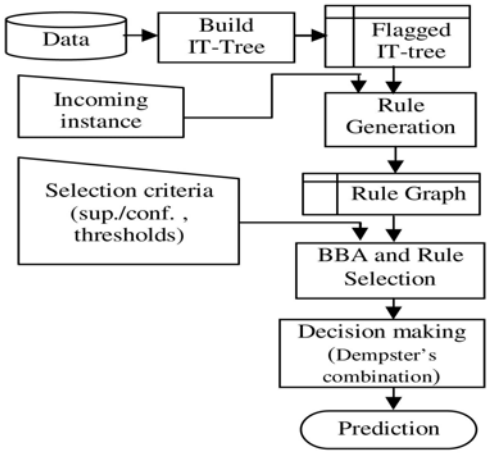

Fig. 1. An overview of our proposed system.

3.1 Itemset Tree (IT-Tree)

Let us briefly summarize the technique of IT-trees as developed in [4]. Here, items are identified by an uninterrupted sequence of integers. Let D denote a set of itemsets and let M be the noumber of distinct items encountered in D. Each item is identified with an integer from [1,M] so that items in an itemset, $s=[a_1,a_2,\ldots a_p]$, can be ordered; $a_i<a_j$ for i<j,where a_i and a_j are integers identified i^{th} and j^{th} items respectively.

Definition 1
(ancestor,largest common ancestor,child):
Let the symbols s,c,and l denote itemsets.

1. s is an ancestor of c and write $s \sqsubseteq c$ iff $s=[a_1,a_2,\ldots a_m]$, c $s=[a_1,a_2,\ldots a_n]$,, and $m \leq n$.
2. we say that *l* is the largest common ancestor of s and c, and write $l = s \sqcap c$ iff $l \sqsubseteq s$, $l \sqsubseteq c$, and there is no *l'* such that $l' \sqsubseteq s$, $l' \sqsubseteq c$, and $l \neq l'$.
3. c is a child of s iff $s \sqsubseteq c$ and there is no *l*, different from s and c, such that $s \sqsubseteq l \sqsubseteq c$.

Definition 2(Itemset tree):
An item set tree T,consists of a root and a (possibly empty) set,$\{T_1,T_2,\ldots..,T_K\}$, each element of which is an itemset tree.The root is a pair [s,f(s)] where s is an itemset and f(s) is a frequency.If s_i denotes the itemset associated with the root of the i-th subtree, then $s \sqsubseteq s_i, s \neq s_i$ must be atisfied for all i.

An IT-tree is a partially ordered set of pairs,[itemset,f],where the f-value tells us how many occurrences of the itemset the node represents. An algorithm that builds the IT-tree in a single pass through the database is presented in [4] that also proves some of the algorithm's critical properties. For example, the number of nodes in the IT-tree is upper-bounded by twice the number of transactions in the original database (although experiments indicate that, in practical applications, the size of the IT-tree rarely exceeds the size of the database). Moreover, each distinct transaction database is represented by a unique IT-tree and the original transaction can be reproduced from the IT-tree. Note that some of the itemsets in IT-tree (e.g.,[1, 2, 4] in Fig. 2) are identical to at least one of the transactions contained in the original database, whereas others (e.g.,[1;,2]) were created during the process of tree building where they came into being as common ancestors of transactions from lower levels. They modified the original tree building algorithm by flagging each node that is identical to at least one transaction. In Fig. 2, the flags are indicated by black dots. This flagged IT-tree will become the base of our rule generation algorithm.

Example 1: (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. 2

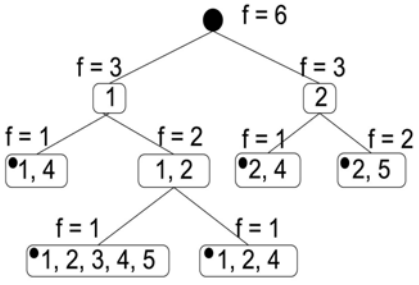

Fig. 2. The IT-tree constructed from the database D in Example 1.

3.2 Rule Generation Mechanism

The proposed rule generation algorithm makes use of the flagged IT-tree created from the training data set. The algorithm takes an incoming itemset as the input and returns a graph that defines the association rules entailed by the given incoming itemset. 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 $f(r_i^{(a)})$.

As shown in Fig. 3, a line, *l*i;j, between the two lists links an antecedent $r_i^{(a)}$ i with a consequent Ij. The cardinality of the link, $f(l_{ij})$, represents the support count of the rule $r_i^{(a)} \rightarrow$ Ij. The frequency counts denoted by fo(·) 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 fo ($r^{(a)}$) records the number of times s ⊓ Ti = $r^{(a)}$. Thus, fo(l_{ij}) records the number of times where s ⊓ Ti = $r^{(a)}$. All the frequency counts are initialized to zero at the beginning of the algorithm and updated as we traverse the IT-tree.

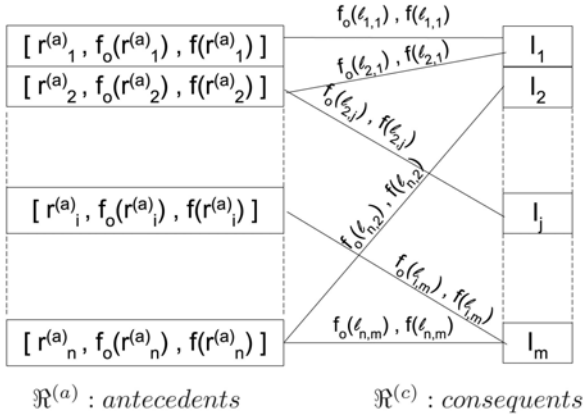

Fig. 3. The Rule Graph, G. $f(r_i^{(a)})$= frequency count of antecedent, $f(l_{ij})$,=support count of rule $r^{(a)} \rightarrow$ Ij.

Algorithm 1 conducts a depth-first search in the IT-tree to identify the nodes that have nonempty intersections with s. Note that the items in s and ci are referred to by their corresponding integer representations and sorted in the ascending order.

Algorithm 1: Rule_mining

The individual steps of the algorithm can be summarized as follows: Let R =[sR, f(sR)] denote the root, ci denote the children of R, and s denote the incoming itemset. If the first item in ci is greater than the last item in s, it is certain that no tree node rooted at ci will contain items from itemset s. If s ⊓ ci =∅ and the last item in ci is greater than the last item in s, then, again, it is certain that no nodes in the subsequent trees have nonempty intersections with s. But if the first item of ci is less than the last item of s, subtree Ti with the root [ci, fc_i] may contain one or more items from s. The algorithm starts the search for rules in subtrees rooted at the children of ci. If s ⊓ $c_i \neq$ ∅, the intersection (say, r^a) is a candidate for a rule antecedent. However, if the node [ci; f(ci)] is not To invoke Rule_mining:

Algorithm that process the itemset tree T and returns the rule graph G that predicts unseen items in a user-specified itemset s. Let R denote the root of T and let {ci,f(c_i)} be Rs children.Let Ti denote the subtree whose root is {ci,f(c_i)}.

G = Rule_mining(s,T,{ }).
1: Rule_mining(s,T,G):
2: for all ci such that first_item(ci) ≤ last_item(s) do
3: if s ⊓ ci =∅ ; and last_item(ci) < last_tem(s) do
4: G← Rule_mining(s,Ti,G);
5: else if $r^{(a)}$ = s ⊓ $c_i \neq$ ∅; then
6: if ci is not flagged then
7: G← Rule mining(s,Ti,G);
8: else
9: if ci does not have children then fo ← f(c_i);
10: elsefo ← f(c_i)-∑f(c_i's children);Δf o is the frequency of ci in the database
11: end if
12: G← Update_Graph(G, r^a, ci, fo);
13: G← Rule mining(s,Ti,G);
14: end if
15: end if
16: end for
17: return G;

flagged (i.e., if itemset ci does not exist in the actual data set), the candidate antecedent looses the candidacy status. Now, the nodes in the subtrees starting from children of ci possess intersections with s that are equal to $r^{(a)}$ or larger than $r^{(a)}$ (i.e., the intersection is a superset of r^a). The algorithm thus continues the search for rules in the subtrees rooted at the children of ci. If ci is flagged, the number of occurrences of ci in the data set is calculated as f(ci)-∑f(c's children). Then, r^a becomes a rule antecedent and each item in ci \$r^a$ becomes aconsequent. The new rules, $r^a \rightarrow$ ij, where ij €(ci\$r^a$),are added to the rule graph Each nonempty intersection of s with a flagged node of the tree generates set of association rules of the form ra→ ij, where r^a is the intersection of s with the node and ij is an item in node such that ij doesnot contain r^a. Note that aflagged node represents an actual transaction in the data set; the number of flagged nodes is upper-bounded by N (the number of transactions in the data set). These rules are added to the rule graph using Algorithm 2. The idea is to update the frequency counts of all rules $r_i^{(a)} \rightarrow$ ij, where $r^a \sqsubseteq r_i^a$. If the new rule does not exist in the rule graph, it has to be added to the graph and the frequency count has to be updated using all the rules of the form $r_i^{(a)} \rightarrow$ ij, where $r^a \sqsubseteq r_i^a$.

Algorithm 2. Simplified algorithm to update the rule graph.
Let G denote the current rule graph.
Let ci denote an itemset from a node and fo denote the number of appearances of ci in the database.
Let r^a = ci ⊓ s where s is the incoming itemset.
To invoke Update_Graph use: G = Update_Graph(G, r^a, Ci, fo).
1: Update_Graph(G, r^a, ci, fo):
2: for all r^a in $R^{(a)}$ do

3: if $r_i^a \sqsubseteq r^a$ then
4: update the frequency count of r_i^a, $f(r_i^a) \leftarrow f(r_i^a)+fo$;
5: for all $r_i^a \rightarrow$ ij where ij $\in$(ci\ r^a) do
6: update frequency count of rule $r^a \rightarrow$ ij, $f(li;j) \leftarrow f(li;j)+ fo$;
7: end for
8: if $r_i^a = r^a$ then
9: update frequency count record, fo $(r_i^a) \leftarrow f_o(r_i^a)+f_0$;
10: for all $r_i^a \rightarrow$ ij where ij $\in$(ci \ r^a) do
11: update frequency count record of rule, $f_0(lij) \leftarrow fo(li;j)+ fo$;
12: end for
13: end if
14: else if $r^a \sqsubseteq r_i^a$ then
15: update the frequency count of new rule antecedent $f(r^{(a)}) \leftarrow f(r^{(a)}) + f_0 (r_i^{(a)})$;
16: for all $r_i^{(a)}$ in G where ij $\in$ (ci\ $r^{(a)}$) do
17: update the frequency count of new rule $r^{(a)\rightarrow}$) ij, (add $f_0(lij)$);
18: end for
19: end if
20: end for
21: for all ij $\in$ (ci\ $r^{(a)}$) do
22: if $\beta(r^{(a)} \rightarrow i_j) \in$ G then
23: add the rule to the graph with corresponding frequency counts;
24: end if
25: end for
26: return G;

The size of the antecedent list R^a of the rule graph G is upper-bounded by $\min(N,2^P)$, where p is the size of the itemset s. The size of R^c is upper-bounded by (n-P), where n is the number of distinct items in the data set. At the beginning, each list is empty; they grow as we traverse through the itemset tree. Algorithm 1 scans the IT-tree and calls Algorithm 2 each time it comes across a flagged node. At each call, Algorithm 2 carries out one traversal of the ruleset (and possibly adds some rules to it). Thus the worst case complexity of the rule generation process is $O(N^2)$. In reality, the computation complexity is much lower because the number of nodes that have nonempty intersections with s usually constitute only a small fraction of N. In addition, when p is small, the size of the rule antecedent list could be much smaller than N (i.e., when $2p < N$).

Example 2 (A Rule Generation Example). We consider the same data set as in the previous example, viz., D ={ [1,4],[2,5],[1,2,3,4,5],[1,2,4],[2,5],[2,4] }. Assume that the incoming itemset s =[2,3]. Fig. 4 shows the step-by-step building of the rule graph by Algorithm 1. The itemset s has nonempty intersections with four flagged nodes, [1,2,3,4,5],[1,2,4],[2,4] and [2,5] .A set of rules is added to the rule graph with each nonempty intersection.
Consider the intersection with the flagged node ci =[1, 2, 3, 4, 5]. The intersection, $r^{(a)}$=[2,3], is added to the antecedent list of the rule graph and the consequents 1,4,5 are added to the consequent list together with links connecting antecedent and consequents. Since the frequency count of the node is fo = 1all the fo (·)values in the graph assume 1, as shown in Fig. 4a (lines 21-25 of Algorithm. 2). At the node [1,2,4] the intersection, [2], is taken as $r^{(a)}$. Frequency counts $f(r^{(a)})$,f_0(,$r^{(a)}$)and all frequency counts of both candidate ([2]→1),([2]→4) are initialized to frequency count of the node [1, 2, 4], fo =1. Since the current rule graph contains [2, 3] in R^a and [2] is a subset of [2, 3], update the frequency counts according to lines 15-18. Add the new rules to the graph (lines 21-25). At the node [2, 4], the intersection is again [2]. Since it is already in the graph, update the frequency counts of the antecedent and corresponding rule (i.e., [2]→4) according to lines 4-13. Processing of the node [2;,5] is similar to the previous case. However, in this case, a new rule [2]→5 is added to the graph (line 21-25).

The ruleset that resides in G is given in Table 1. The rule [2, 3]→ 1 suggests that, if the itemset [2;,3] is present in a shopping cart, item 1 is likely to be added to the cart. Support of this rule is 1=6 and the confidence is 1. Note that the ruleset in Table 1 consists of only two distinct antecedents: [2] and [2, 3]. Since no minimum support or confidence threshold is applied yet, one may expect another ruleset with the antecedent [3]. However, our algorithm does not generate rules having antecedent [3]. Note that no transaction Ti in the data set D provides an intersection Ti ⊓ s = [3], that is, whenever item 3 appears in a transaction, one or more of other items from s happen to appear in Ti, too. So, item 3 alone does not provide any additional evidence for the given itemset s. This is why our rule-generating algorithm ignores such rules.It is important to note here that one might be interested in rules that suggest the absence of items.

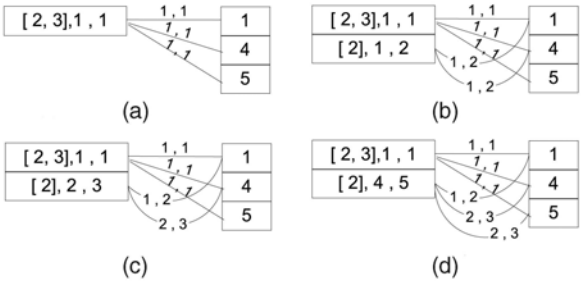

Fig. 4. Rule graph construction for the testing itemset [2, 3] using IT-tree in Fig. 2 (Testing itemset possesses nonempty intersections with only four nodes of the tree). (a) After node [1,2,3,4,5], (b) after node [1,2,4], (c) after node [2,4], and (d) shows the final rule graph, G, after node [2,5].

TABLE 1
Rule Set That Resides in G

Rule	Support Count	
	Antecedent	Rule
[2, 3] ⇒ 1	1	1
[2] ⇒ 1	5	2
[2, 3] ⇒ 4	1	1
[2] ⇒ 4	5	3
[2, 3] ⇒ 5	1	1
[2] ⇒ 5	5	3

For instance, [2, 3] → (1 = absent), that is, when items 2 and 3 are already present in the cart, then item 1 is unlikely to be added to the cart in the future. In this event, the IT-tree-building algorithm has to regard the (item; value) pair as an item. For instance(1 = present) is one item and (1 = absent) is another. Then, the generated ruleset will eventually consist of rules suggesting both the presence and absence of items. These rules then have to be combined to yield the final decision.Note that 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. How the selected rules are used for prediction is described in the next section.

4. EMPLOYING DS THEORY

When searching for a way to predict the presence or absence of an item ij in a partially observed shopping cart s, we wanted to use association rules. However, many rules with equal antecedents differ in their consequents—some of these consequents contain ij, others do not. The question is how to combine (and how to quantify) the potentially conflicting evidence. One possibility is to rely on the DS theory of evidence combination. Let us now describe our technique, which we refer to by the acronym DS-ARM(Dempster-Shafer-based Association Rule Mining).

4.1 Preliminaries

Let $\Theta = \{\theta(1), \ldots, \theta(n)\}$ be a finite set of mutually exclusive and exhaustive propositions signifying the "scope of expertise" about some problem domain. It is referred to as its *frame of discernment (FoD).*A proposition $\theta(i)$, which is referred to as a *singleton*, represents the lowest level of discernible information. The elements in 2Θ, i.e., the power set of Θ, form all propositions of interest. A proposition that is not a singleton is referred to as a *composite*, e.g., $(\theta(1), \theta(2))$.

Definition:

The mapping $m : 2\Theta \rightarrow [0, 1]$ is a basic belief assignment (BBA) or mass assignment for the FoD Θ if $m(\Theta) = 0$ and $\sum_{A \subseteq \Theta} m(A) = 1$.

The BBA of a proposition is free to move into its individual singletons. This is how the DS theory models ignorance. A proposition that possesses a nonzero BBA is referred to as a focal element; and the set of focal elements is the *core* and is denoted by *F*. The triple {Θ,F,m} is referred to as the body of evidence (BoE); and the number of focal elements is |*F*|.

4.2 Concrete Application to Our Task

4.2.1 Basic Belief Assignment

In association mining techniques, a user-set minimum support decides about which rules have "high support." Once the rules are selected, they are all treated the same, irrespective of how high or how low their support. Decisions are then made solely based on the confidence value of the rule. However, a more intuitive approach would give more weight to rules with higher support. Therefore, we propose a novel method to assign to the rules masses based on both their confidence and support values. However, the support value should have a smaller impact on the mass.In many applications, the training data set is skewed. To account for this data set skewness, we propose to adopt a modified support value as follows:

Definition (Partitioned-Support). The partitioned-support p_supp of the rule $r^{(a)} \rightarrow r^{(c)}$, is the percentage of transactions that contain r^a among those transactions that contain $r^{(c)}$, i.e.,

$$p\_supp = support(r^{(a)} \cup r^{(c)})/support(r^{(c)}). \quad - (2)$$

With Definition 2 in place, we take inspiration from the traditional Fα-measure [24] and use the weighted harmonic mean of support and confidence to assign the following BBA to the rule $r^{(a)} \rightarrow ij$:

$$m(i_j/r^a) = \begin{cases} \beta, & \text{for } i_j = \text{present}, \\ 1-\beta, & \text{for } i_j = \Theta, \\ 0, & \text{otherwise} \end{cases} \quad - (3)$$

Where

$$\beta = \frac{(1+\alpha^2)*conf*p\_supp}{\alpha^2*conf+p\_supp} \quad - (4)$$

Where

$\alpha \in [0,1]$

Note that, for the task at hand, ij = 1 and hence, $\Theta = \{(i_j = \text{present}), (i_j = \text{absent})\}$. Note that, as α decreases, the emphasis placed upon the partitioned-support measure in m(·) decreases as well. With this mass allocation, the effectiveness of a rule is essentially tied to both its confidence and partitioned support.

4.2.2 Discounting Factor

Following the work in [15], the reliability of the evidence provided by each contributing BoE is addressed by incorporating the following discounting factor:

$$d=[1+Ent]^{-1}[1+\ln(n=(r^{(a)})]^{-1}, \quad - (5)$$

$$\text{with } Ent= - \sum_{i_j \in \Theta} m(i_j/r^{(a)})\ln[m(i_j/r^{(a)})].$$

Recall that n denotes the number of items in the database. The term $[1+Ent]^{-1}$ accounts for the uncertainty of the rule about its consequent. The term $[1+\ln(n=(r^{(a)})]^{-1}$ accounts for the nonspecificity in the rule antecedent. Note that d increases as Ent decreases and length of rule antecedent increases. As dictated by 11, the BBA then gets accordingly modified. The DRC is then used on the modified BoEs to combine the evidence.

Example 3.Table 2 shows a ruleset generated for itemset (bread, milk) and a supper market data set that contains five

distinct items viz. {egg, bread, butter, milk, wheat bread}. Integers from 1 to 5 are used to denote the presence of items and 6 to 10 are used to denote the absence of items. For instance, (egg = present)=1,(bread = present)=2,(egg = absent)=6 etc. Last two columns show the computed BBA and d values of the rules. In this example, mass assignment is done using three times less weight for the partitioned-support compared to the confidence, i.e., α=0:33 and d is computed using (5).

TABLE 2
An Example Ruleset

		support count						$m(\bullet \mid r_a)$		
	rule	ante.	rule	cons.	conf.	supp.	p_supp	proposition	BBA	d
1	$[2,4] \Rightarrow 1$	3	2	3	0.67	0.33	0.67	$\langle egg \rangle$	0.67	0.29
2	$[2,4] \Rightarrow 6$	3	1	3	0.33	0.17	0.33	$\langle No\ egg \rangle$	0.33	0.29
3	$[2] \Rightarrow 6$	5	3	3	0.60	0.50	1.00	$\langle No\ egg \rangle$	0.62	0.25
4	$[2,4] \Rightarrow 3$	3	1	3	0.33	0.17	0.33	$\langle butter \rangle$	0.33	0.29
5	$[2] \Rightarrow 3$	5	3	3	0.60	0.50	1.00	$\langle butter \rangle$	0.62	0.25
6	$[2,4] \Rightarrow 8$	3	2	3	0.67	0.33	0.67	$\langle No\ butter \rangle$	0.67	0.29
7	$[2,4] \Rightarrow 10$	3	3	5	1.00	0.50	0.60	$\langle No\ w.bread \rangle$	0.94	0.39
8	$[2] \Rightarrow 10$	5	5	5	1.00	0.83	1.00	$\langle No\ w.bread \rangle$	1.00	0.42

To keep the rules as independent as possible, we then removed 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. For instance, rules 2 and 3 both suggest "no egg," and the antecedent of the second rule is a subset of the antecedent of the first rule. However, rule 2 has lower confidence than rule 3.

5. CONCLUSION

The mechanism reported in this paper focuses on one of the oldest tasks in association mining: based on incomplete information about the contents of a shopping cart, can we predict which other items the shopping cart contains? Our literature survey indicates that, while some of the recently published systems can be used to this end, their practical utility is constrained, for instance, by being limited to domains with very few distinct items. Bayesian classifier can be used too, but we are not aware of any systematic study of how it might operate under the diverse circumstances encountered in association mining. We refer to our technique by the acronym DS-ARM. The underlying idea is simple: when presented with an incomplete list s of items in a shopping cart, our program
first identifies all high-support, high-confidence rules that have as antecedent a subset of s. Then, it combines the consequents of all these (sometimes conflicting) rules and creates a set of items most likely to complete the shopping cart. Two major problems complicate the task: first, how to identify the relevant rules in a computationally efficient manner; second, how to combine (and quantify) the evidence of conflicting rules. We addressed the former issue by the recently proposed technique of IT-trees and the latter by a few simple ideas from the DS theory.

6. REFERENCES

[1] S. Noel, V.V. Raghavan, and C.H. Chu, "Visualizing Association Mining Results through Hierarchical Clusters," Proc. Int'l Conf.Data Mining (ICDM '01) pp. 425-432, Nov./Dec. 2001.

[2] P. Bollmann-Sdorra, A. Hafez, and V.V. Raghavan, "A Theoretical Framework for Association Mining Based on the Boolean Retrieval Model," Data Warehousing and Knowledge Discovery: Proc.Third Int'l Conf. (DaWaK '01), pp. 21-30, Sept. 2001.

[3] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," Proc. ACM Special Interest Group on Management of Data (ACM SIGMOD),pp. 207-216, 1993.

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

[5] V. Ganti, J. Gehrke, and R. Ramakrishnan, "Demon: Mining and Monitoring Evolving Data," Proc. Int'l Conf. Data Eng., 1999.

[6] J. Gehrke, V. Ganti, and R. Ramakrishnan, "Detecting Change in Categorical Data: Mining Contrast Sets," Proc. ACM SIGMODSIGACT- SIGART Symp. Principles of Database Systems, pp. 126-137, 2000.

[7] V. Raghavan and A. Hafez, "Dynamic Data Mining," Proc. 13th Int'l Conf. Industrial and Eng. Applications of Artificial Intelligence and Expert Systems IEA/AIE, pp. 220-229, June 2000.

[8] J. Zhang, S.P. Subasingha, K. Premaratne, M.-L. Shyu, M. Kubat, and K.K.R.G.K. Hewawasam, "A Novel Belief Theoretic Association Rule Mining based classifier for Handling Class Label ambiguities," Proc. Workshop Foundations of Data Mining (FDM '04), Int'l Conf. Data Mining (ICDM '04), Nov. 2004.

[9]] K.K.R.G.K. Hewawasam, K. Premaratne, and M.-L. Shyu, "Rule Mining and Classification in a Situation Assessment Application: A Belief Theoretic Approach for Handling Data Imperfections," IEEE Trans. Systems, Man, Cybernetics, B, vol. 37, no. 6 pp. 1446-1459, Dec. 2007.