Predicting Missing Items in Shopping Carts using Fast Algorithm

Srivatsan. M
Sri Venkateswara College of Engineering, Chennai- 602105

Sunil Kumar. M
Sri Venkateswara College of Engineering, Chennai- 602105

Vijayshankar. V
Sri Venkateswara College of Engineering, Chennai- 602105

Leela Rani P
Assistant Professor
Sri Venkateswara College of Engineering, Chennai- 602105

ABSTRACT
Prediction in shopping cart uses partial information about the contents of a shopping cart for the prediction of what else the customer is likely to buy. In order to reduce the rule mining cost, a fast algorithm generating frequent itemsets without generating candidate itemsets is proposed. The algorithm uses **Boolean vector with relational AND operation** to discover frequent itemsets and generate the association rule. Association rules are used to identify relationships among a set of items in database. Initially Boolean Matrix is generated by transforming the database into Boolean values. The frequent itemsets are generated from the Boolean matrix. Then association rules are to generated from the already generated frequent itemsets. The association rules generated form the basis for prediction. The incoming itemset i.e the content of incoming shopping cart will also be represented by a Boolean vector and AND operation is performed with each transaction vector to generate the association rules. Finally the rules are combined to get the predictions. Dempster's rule of combination (DRC) is used to combine the evidences. Finally the predicted items are suggested to the user.

Keywords
Association Rule Mining. Boolean Vector, Prediction, Basic Belief Assignment, Demster Shafer Theory of Rule Combination.

1. INTRODUCTION
1.1 Data Mining
Data Mining refers to extracting or mining information from large amounts of data. Data mining has attracted a great deal of attention in the information industry and in society as a whole in recent years, due to the wide availability of huge amounts of data and the imminent need for turning such data into useful information and knowledge.

Data mining, "*the extraction of hidden predictive information from large databases*", is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions.

The automated, prospective analysis offered by data mining move beyond the analysis of past events provided by retrospective tools typical of decision support systems. Data mining tools can answer business questions that traditionally were too time consuming to resolve. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations.

Most companies collect and refine massive quantities of data. Data mining techniques can be implemented rapidly on existing software and hardware platforms to enhance the value of existing information resources and can be integrated with new products and systems as they are brought on-line. When implemented on high performance client/server or parallel processing computers, data mining tools can analyze massive databases to deliver answers to many questions.

The information and knowledge gained can be used for application ranging from market analysis, fraud detection, and customer retention, to production control and science exploration. Data Mining plays an important role in online shopping for analyzing the subscribers' data and understanding their behaviors and making good decisions such that customer acquisition and customer retention are increased which gives high revenue.

1.2 Association Rule Mining
Association Rule Mining [1] is a popular and well researched ethod for discovering interesting relations between variables in large databases. Association rules are statements of the form {X1, X2, …, Xn} => Y meaning that if all of X1, X2,… Xn is found in the market basket, and then we have good chance of finding Y. the probability of finding Y for us to accept this rule is called the confidence of the rule. Normally rules that have a confidence above a certain threshold only will be searched. In many situations, association rules involves sets of items that appear frequently. For example, a good marketing strategy cannot be run involving items that no one buys. Thus, much data mining starts with the assumption that sets of items with support are only considered.

The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customer and which items bring them better profits when placed with in close proximity. The two types of finding association between products existing in a large database are Boolean [7] and Quantitative. Boolean association rule mining finds association for the entire dataset. Quantitative association rule mining finds association for the clusters formed from the dataset.

1.3 Prediction
Data mining automates the process of finding predictive information in large databases. Questions that traditionally

required extensive hands-on analysis can now be answered directly from the data quickly.

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.

Early attempts for prediction used classification [8] and performance was favourable. In this project, any item is allowed to be treated as a class label its value is to be predicted based on the presence of other items. Put another way, knowing a subset of the shopping cart's contents, we want to "guess" (predict) [1] the rest. Suppose the shopping cart of a customer at the checkout counter contains bread, butter, milk, cheese, and pudding. Could someone who met the same customer when the cart contained only bread, butter, and milk, have predicted that the person would add cheese and pudding?

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 (because of the task's combinatorial nature); second, identifying the most relevant rules and combining their sometimes conflicting predictions may easily incur prohibitive computational costs. In this work, both of these problems are solved 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.

This paradigm can be exploited in diverse applications. For example, in the each "shopping cart" contained a set of hyperlinks pointing to a Web page; 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.

In all these databases, prediction of unknown items can play a very important role. For instance, a patient's symptoms are rarely due to a single cause; two or more diseases usually conspire to make the person sick. Having identified one, the physician tends to focus on how to treat this single disorder, ignoring others that can meanwhile deteriorate the patient's condition. Such unintentional neglect can be prevented by subjecting the patient to all possible lab tests. However, the number of tests one can undergo is limited by such practical factors as time, costs, and the patient's discomfort. A decision-support system advising a medical doctor about which other diseases may accompany the ones already diagnosed can help in the selection of the most relevant additional tests.

1.4 Existing Approach

The existing system uses flagged Itemset trees for rule generation purpose. An itemset tree, T, consists of a root and a (possibly empty) set, {T1; . . . ;Tk}, 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 si denotes the itemset associated with the root of the ith subtree, then s is a subset of si; s not equal to si, must be satisfied for all i. The number of nodes in the IT-tree is upper-bounded by twice the number of transactions in the original database.

Note that some of the itemsets in IT-tree [4] are identical to at least one of the transactions contained in the original database, whereas others 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. These are indicated by black dots. This is called flagged IT-tree [4].

Here is an example for an IT-tree [4].

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

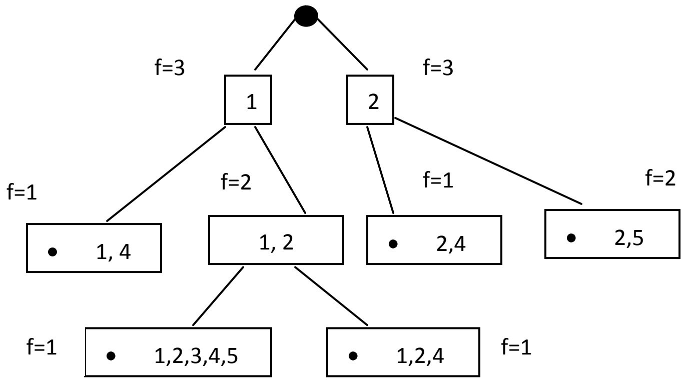

Fig.1.1. Example IT-Tree

Here some items in this tree are flagged to represent them as weak entity so that they are not carried for the next stage of processing.

The disadvantages of existing approach are

- Time taken to construct IT-Tree [4] is more when compared to Boolean matrix method.
- This method requires more memory for processing.

1.5 Dempster's Rule of Combination

Dempster's rule of combination (DRC) [6] is used to combine the discovered. When searching for a way to predict the presence of an item in partially observed shopping carts, association rules are used. However, many rules with equal antecedents differ in their consequents and some of these consequents contain the desired item to be predicted, others do not. The question is how to combine (and how to quantify) the potentially conflicting evidences. DRC [6] is used for this purpose. Finally the predicted items are suggested to the user.

2. SYSTEM DESIGN

Any project developed today is said to be good only if it has some basic characteristics such as modularity, loose coupling and high cohesion. A component is classified as good quality only if it is modular, loosely coupled and has high cohesion i.e., each component should be independent of the other and each component must be focused only on its particular purpose. Finally the component should be modular so that the development of the components is understandable, can easily be enhanced in the future and also easy to locate and correct errors without affecting the other components involved in the project.

The following sections deals with how this system is designed, the modules involved and overall architecture diagram of the system which shows the modules present in the system.

2.1 Shopping Cart Prediction Architecture

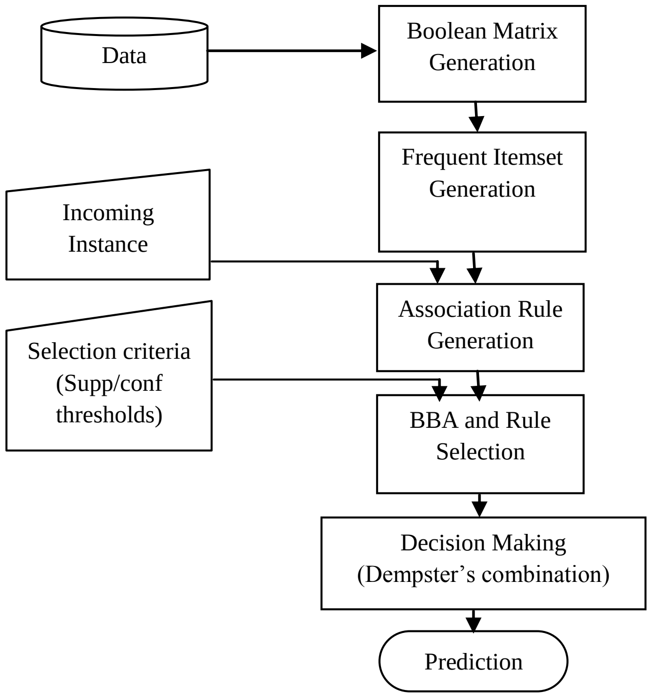
Data
Boolean Matrix Generation
Frequent Itemset Generation
Incoming Instance
Association Rule Generation
Selection criteria (Supp/conf thresholds)
BBA and Rule Selection
Decision Making (Dempster's combination)
Prediction

Fig.2.1. Shopping cart prediction architecture

Fig.4.1. shows the shopping cart prediction architecture in which the Boolean Matrix is generated by transforming the database into Boolean values. The frequent itemsets are generated from the Boolean matrix. At this stage we need the Support value. Then association rules are to generated from the already generated frequent itemsets. It takes minimum confidence from the user and discovers all rules with a fixed antecedent and with different consequent. The association rules generated form the basis for prediction.

We assign BBA [2] value to each association rule generated. This gives more weight to rules with higher support masses are assigned based on both their confidence and support values.

The incoming itemset i.e the content of incoming shopping cart will also be represented by a Boolean vector and AND operation is performed with each transaction vector to generate the association rules. Finally the rules are combined to get the predictions.

Dempster's rule of combination (DRC) [6] is used to combine the evidences. When searching for a way to predict the presence or absence of an item 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 the desired item to be predicted, others do not. The question is how to combine (and how to quantify) the potentially conflicting evidences. DRC [6] is used for this purpose. Finally the predicted items are suggested to the user.

3. IMPLEMENTATION

This topic consists of detailed description of each and every module with its advantages and data and execution flow of each module with algorithm. It helps to understand each and every module of the project more deeply and clearly. Each description consists of the basic concept of the module, input and also the excepted output.

3.1 Modules

The project has been divided into various modules and each module has been completed within a scheduled time line. The following are the modules of the project are boolean matrix generation,Frequent itemset generation, Association rule generation, BBA and decision making.

3.2 Boolean Matrix Generation

This module is to convert the data's in the database and the incoming instance to database into Boolean value (either 0's or 1's). If an item is present in the transaction it is marked with the Boolean value 1 else the item is marked as 0. Raw database "**rdb**" [7] is a m x n matrix where 'm' is the number of transactions and 'n' is the number of attributes. By using above mentioned rule the Raw database in converted into Boolean database "**bdb**" [8] (rdb [i, j] => bdb [i, j] where 'i' represent the rows and 'j' represent the columns).

The algorithm used is given here.

```
for all i<=m do
for all j<=n do
if jth item is present in ith row
set rdb[i,j] =1
else
set rdb[i,j] =0
end do
end do
```

The example input database given to this stage is shown here. The database contains eight transactions and seven items. In real world this will be of large size but this is used for illustration purpose.

Table.3.1. Example Input Database

Table							
ID	Field1	Field2	Field3	Field4	Field5	Field6	Field7
7	N	N	N	N	N	N	Y
8	Y	Y	N	N	Y	Y	N
9	Y	Y	N	N	Y	y	Y

Table

ID	Field1	Field2	Field3	Field4	Field5	Field6	Field7
10	N	N	Y	Y	Y	y	N
11	N	N	N	Y	Y	y	N
12	Y	Y	N	N	N	y	Y
13	N	N	Y	Y	Y	y	Y
14	N	N	Y	Y	Y	y	Y
15	N	N	Y	Y	Y	y	N

Table.3.1 shows that If an item is contained in a transaction, the corresponding attribute value will be 'Y', otherwise the value will be 'N'.The output of this module will be the Boolean matrix, which will look as this.

$$\begin{matrix}\mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0}\\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1}\\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0}\\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1}\\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1}\\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1}\\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0}\end{matrix}$$

If an item is contained in a transaction, the corresponding attribute value will be 1; otherwise the value will be 0.

3.3 Frequent Itemset Generation

This module finds out the frequent item set from the existing transaction based on the support value. It involves Join step and Prune step. This module takes the input from the previous stage and forms the frequent itemset from matrix table whose values are 1 for transaction. This module also generates the Boolean vector for the frequent item set along with support value. Boolean vector takes the value '**true**' for the item present in the itemset and takes the value '**null**' for the item not present in the itemset.

The algorithm used is shown here. It has two steps as explained above.

```
for each column $c_i$ of pdb
if sum($c_i$) >= new_support
$f_1 = i_i$
else delete $c_i$ from pdb
for each row $r_j$ of pdb
if sum($r_j$) < 2
delete $r_j$ from pdb
for (k=2;| $f_{k-1}$|>k-1;k++)
{
 produce k-vectors combination for all columns of bdb;
 for each k-vectors combination {$c_{i1}$,$c_{i2}$, $c_{i3}$ … ,$c_{ik}$ }
 {
 b= $c_{i1}$ • $c_{i2}$ •….•$c_{ik}$
 if sum(b)>= new_support
    fk={ $i_{i1}$, $i_{i2}$,……,$i_{ik}$ };
    }
    for each item $i_i$ in $f_k$
    if | $f_k(i_i)$| < k
    delete the column $c_i$ according to item $i_i$ from bdb;
    for each row $r_j$ from bdb
    if sum($r_j$) < k+1
    delete $r_j$ from bdb;
    k=k+1
    }
    return f= $f_1$ u $f_2$ … u $f_k$
```

The input given to this stage is the Boolean matrix generated in previous module. This is the sample input that is given for illustration. The support value is also given as input.

$$\begin{matrix}\mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0}\\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1}\\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0}\\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1}\\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1}\\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1}\\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0}\end{matrix}$$

Support Value: 50

The output of this stage is the frequent itemsets generated from Boolean matrix. Each frequent itemset generated is also converted into a Boolean vector as shown in the table below.

```
Input configuration: 7 items, 8 transactions, minsup = 50.0%
Frequent 1-itemsets
[3, 4, 5, 6, 7]
Frequent 2-itemsets
[3 4, 3 5, 3 6, 4 5, 4 6, 5 6, 5 7, 6 7]
Frequent 3-itemsets
[3 4 5, 3 4 6, 3 5 6, 4 5 6, 5 6 7]
Frequent 4-itemsets
[3 4 5 6]
```

Table.3.2. Frequent itemsets generation output

Itemset	Boolean Vector	Support
3	[null, false, false, true, false, false, false, false]	0.5
4	[null, false, false, false, true, false, false, false]	0.625
5	[null, false, false, false, false, true, false, false]	0.875
6	[null, false, false, false, false, false, true, false]	1.0
7	[null, false, false, false, false, false, false, true]	0.625
3 4	[null, false, false, true, true, false, false, false]	0.5
3 5	[null, false, false, true, false, true, false, false]	0.5
3 6	[null, false, false, true, false, false, true, false]	0.5
4 5	[null, false, false, false, true, true, false, false]	0.625
4 6	[null, false, false, false, true, false, true, false]	0.625
5 6	[null, false, false, false, false, true, true, false]	0.875
5 7	[null, false, false, false, false, true, false, true]	0.5
6 7	[null, false, false, false, false, false, true, true]	0.625
3 4 5	[null, false, false, true, true, true, false, false]	0.5
3 4 6	[null, false, false, true, true, false, true, false]	0.5
3 5 6	[null, false, false, true, false, true, true, false]	0.5
4 5 6	[null, false, false, false, true, true, true, false]	0.625
5 6 7	[null, false, false, false, false, true, true, true]	0.5

Table.3.2. shows the frequent itemsets along with the Boolean Vector generated from Boolean Matrix. User Support value is needed for this. If an item is present in the frequent itemset, it takes the value 'True' in the Boolean Vector, otherwise the value will be 'False'. It involves Join step and Prune step. The frequent itemset generated should have desired support value.

3.4 Association Rule Generation

This module is used to generate association rules from the already generated frequent itemsets.The algorithm uses the fact that:

"If there exists two rules A->C and A->{C U X} where X doesnt belongs to A U C then the confidence of the second cannot be larger than the first one".

The algorithm checks if a given set is a subset of another set or not. To perform this operation each item in an itemset is represented as an integer where a bit corresponding to as item is set to 1.

For example, suppose a database with 8 attributes, itemset {1,2, 5} is represented as 38 as follows. 0 0 1 0 0 1 1 0

To check if set {2,5} is a subset of {1,2,5} we represent {2,5} like above and is evaluated to 36. Now we perform AND operation 38 & 36. The result is checked for equality with the first itemset ({2, 5}). If they are equal then it is a subset otherwise it is not. In this case the result is obvious. Similarly difference of two sets is done during production of the rules.

This algorithm is capable of finding all association rules with a fixed antecedent and with different consequents from the frequent itemsets subject to a user specified minimum confidence very quickly.

It takes minimum confidence from the user and discovers all rules with a fixed antecedent and with different consequent. This module also takes the frequent item set and the incoming shopping cart instance to generate the association rule with the corresponding support and confidence value.

The algorithm used is shown here.

```
for all $f_k$, $f_k \in F$, 1<=k<=maxsize-1 do begin
    rsup=support($f_k$)*miconf
    found=0
    for all $f_m$, $f_m$ _ $F_k$ +1<= m <=maxsize do begin
    if (support($f_m$)>=rsup) then begin
    if($f_k \subset f_m$) then begin
    found=found+1
    conf=support($f_m$)/ support($f_k$)
    generate the rule $f_k$ = ($f_m$ - $f_k$) &= conf and support=support($f_m$)
    end if
    else
    if (found<2)
    continue step1 with next k
    else found=0
    endif
    endif
    end do
    end do
```

The input given to the rule generation process is the Boolean vectors representing each transaction and also the contents of incomplete shopping cart. This is shown in Table 5.3.

Table.3.3. Rule Generation Input

Itemset	Boolean Vector	Support
3	[null, false, false, true, false, false, false, false]	0.5
4	[null, false, false, false, true, false, false, false]	0.625
5	[null, false, false, false, false, true, false, false]	0.875
6	[null, false, false, false, false, false, true, false]	1.0
7	[null, false, false, false, false, false, false, true]	0.625
3 4	[null, false, false, true, true, false, false, false]	0.5
3 5	[null, false, false, true, false, true, false, false]	0.5
3 6	[null, false, false, true, false, false, true, false]	0.5
4 5	[null, false, false, false, true, true, false, false]	0.625
4 6	[null, false, false, false, true, false, true, false]	0.625
5 6	[null, false, false, false, false, true, true, false]	0.875
5 7	[null, false, false, false, false, true, false, true]	0.5
6 7	[null, false, false, false, false, false, true, true]	0.625
3 4 5	[null, false, false, true, true, true, false, false]	0.5
3 4 6	[null, false, false, true, true, false, true, false]	0.5
3 5 6	[null, false, false, true, false, true, true, false]	0.5
4 5 6	[null, false, false, false, true, true, true, false]	0.625
5 6 7	[null, false, false, false, false, true, true, true]	0.5

```
Enter confidence: 80
Enter incoming shopping cart contents: 3 4
```

Table.3.4. Rule Generation Output

Rule	Supp	conf
3 4 ->5	0.5	1.0
3 4 ->6	0.5	1.0

Table.3.4. shows the output rules generated along with the support and confidence values.

3.5 BBA And Decision Making

3.5.1 PARTITIONED-SUPPORT

In many applications, the training data set is skewed. Thus, in a supermarket scenario, the percentage of shopping carts containing, say canned fish, can be 5 percent, the remaining 95 percent shopping carts not containing this item. Hence, the rules that suggest the presence of canned fish will have very low support while rules suggesting the absence of canned fish will have a higher support.

Unless compensated for, a predictor built from a skewed training set typically tends to favor the "majority" classes at the expense of "minority" classes. In many scenarios, such a situation must be avoided.To account for this data set skewness, we propose to adopt a modified support value termed 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.,

$$\textbf{p\_supp} = \textbf{support}(r^{(a)} \cup r^{(c)}) / \textbf{support}(r^{(c)})$$

3.5.2 BBA:

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 use a novel method to assign to the rules masses based on both their confidence and support values. This weight value is called Basic Belief Assignment (BBA) [2].

We assign BBA value to each association rule generated.

$$\beta = ((1+\alpha^2) \times \text{conf} \times \text{p\_supp}) / (\alpha^2 \times \text{conf} + \text{p\_supp}); \quad \alpha \in [0,1];$$

Dempster's rule of combination (DRC) [6] is used to combine the evidences. When searching for a way to predict the presence or absence of an item in a partially observed shopping carts, we wanted to use association rules. However, many rules with equal antecedents differ in their consequents—some of these consequents contain the desired item to be predicted, others do not. The question is how to combine (and how to quantify) the potentially conflicting evidences. DRC [6] is used for this purpose. Some illustrations used from DRC [6] are explained in following paragraph.

We 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. Finally the best rule is selected by comparing the mass values. The predicted item is then suggested to the user.

The input given to this stage is the set of rules generated along with their support and confidence values as shown in Table 5.5.

Table.3.5. BBA and Decision Making Input

Rule	Supp	conf
3 4 ->5	0.5	1.0
3 4 ->6	0.5	1.0

Table.3.6. BBA and Decision Making Output

Rule	Supp	Conf	p_supp	BBA
3 4 ->5	0.5	1.0	0.57142857142857140	0.9313986939084056
3 4 ->6	0.5	1.0	0.50	0.9105764493348661

Predicted Result: 5

Table 3.6 shows the output of this stage i.e, the calculated BBA [6] values. The prediction made is also shown below the table.

3.6 Comparison

The performance of both the existing tree approach and the proposed approach is analyzed with databases of different sizes. The results found are very much surprising in the proposed approach compared to the tree approach. The time of prediction has decreased to a great extent compared to existing tree approach.

Table.3.7. Execution Time Comparison

Number of Transactions	Tree Approach	Proposed Approach
	Execution Time	Execution Time
100	48.7	0.797
80	44.172	0.625
60	42.031	0.578
40	40.015	0.593
20	38.721	0.547

Table.3.7. shows the comparison of execution time between the existing tree approach and proposed approach for different transactions.

3.7 Performance Evaluation

The Fig.3.1. shows the performance evaluation graph which compares the performance of both the existing tree approach and proposed approach and displays the time taken to execute for different transactions in seconds.

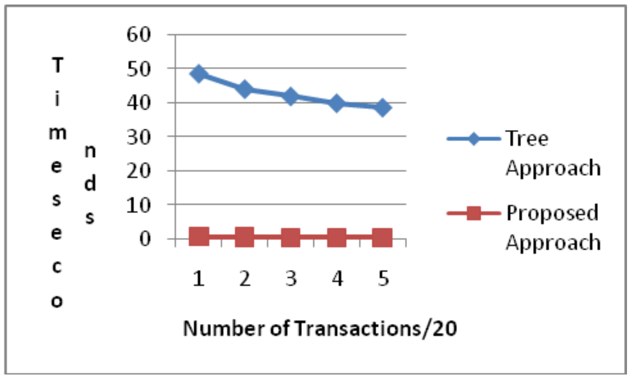

Fig.3.1. Performance Evaluation Graph

4. CONCLUSION

The fast algorithm generating frequent itemsets without generating candidate itemsets proposed performs well compared to existing approaches. The execution time is much improved as shown in performance testing. The algorithm used Boolean vector with relational AND operation to discover frequent itemsets and generate the association rule. Association rules formed the basis of prediction. The algorithm is applied in a demo shopping cart application. When user adds each item to the cart the algorithm is executed and the prediction is displayed.

The Advantages of proposed work are

- The proposed algorithm doesn't generate the candidate itemsets.
- It uses only a single pass over the database.
- The memory consumed is also very less.
- Processing speed is more when compared to rules generated using item set tree and DS theory.

5. FUTURE ENHANCEMENT

The method proposed in this project was tested with a demo shopping cart and the performance is found acceptable. But further improvements can be made to reduce the cost of rule generation process. Also new data structures can be proposed that will be more suitable to handle large number of itemsets as in shopping cart.

6. REFERENCES

[1]. Kasun Wickramaratna, Miroslav Kubat and Kamal Premaratne, "Predicting Missing Items in Shopping Carts", IEEE Trans. Knowledge and Data Eng., vol. 21, no. 7, july 2009.

[2]. M.Anandhavalli, Sandip Jain, Abhirup Chakraborti, Nayanjyoti Roy and M.K.Ghose "Mining Association Rules Using Fast Algorithm", Advance Computing Conference (IACC), 2010 IEEE 2nd International.

[3]. H.H. Aly, A.A. Amr, and Y. Taha, "Fast Mining of Association Rules in Large-Scale Problems," Proc. IEEE Symp. Computers and Comm. (ISCC '01), pp. 107-113, 2001.

[4]. R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. Int'l Conf.Very Large Databases(VLDB '94), pp.487-499, 1994.

[5]. 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.

[6] Apriori Algorithm Reference URL: http://www2.cs.uregina.ca/~dbd/cs831/notes/itemsets/itemset_prog1.html

[7] 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.

[8] W. Li, J. Han, and J. Pei, "CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules," Proc. IEEE Int'l Conf. Data Mining (ICDM '01), pp. 369-376, Nov./Dec. 2001.

[9] 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.