Association Rules for Recommendations with Multiple Items

Abhijeet Ghoshal
College of Business, University of Illinois at Urbana Champaign, Champaign, IL, 61820, USA, abhijeet1998@gmail.com

Sumit Sarkar
Naveen Jindal School of Management, The University of Texas at Dallas, Richardson, TX, 75080, USA, sumit@utdallas.edu

Abstract

In web based environments, a site has the ability to recommend multiple items to a customer in each interaction. Traditionally, rules used to make recommendations either have single items in their consequents or have conjunctions of items in their consequents. Such rules may be of limited use when the site wishes to maximize the likelihood of the customer being interested in at least one of the items recommended in each interaction (with multiple interactions comprising a session). Rules with disjunctions of items in their consequents and conjunctions of items in their antecedents are more appropriate for such environments. We refer to such rules as disjunctive consequent rules. We have developed a novel mining algorithm to obtain such rules. We identify several properties of disjunctive consequent rules that can be used to prune the search space when mining such rules. We demonstrate that the pruning techniques drastically reduce the proportion of disjunctive rules explored, with the pruning effectiveness increasing rapidly with an increase in the number of items to be recommended. We conduct experiments to compare the use of disjunctive rules with that of traditional (conjunctive) association rules on several real world datasets and show that the accuracies of recommendations made using disjunctive consequent rules are significantly higher than those made using traditional association rules. We also compare the disjunctive consequent rules approach with two other state-of-the-art recommendation approaches – collaborative filtering and matrix factorization. Its performance is generally superior to both these techniques on two transactional datasets. The relative performance on a very sparse click-stream dataset is mixed. Its performance is inferior to that of collaborative filtering and superior to that of matrix factorization for that dataset.

Key words: Data mining, disjunctive rules, personalization, bounce rate, collaborative filtering, matrix factorization

1. Introduction

Internet-based applications have proliferated in recent years, and firms are increasingly resorting to the use of personalization and recommendation services. For instance, retail web sites such as Amazon.com and Costco use a variety of personalization techniques to recommend books and gifts to their customers. Advertising servers such as Google (which owns Doubleclick) and Yahoo target appropriate banner advertisements to visitors on their clients' sites. Studies have shown that personalized recommendations enable firms to effectively target customers with products and services (Häubl and Trifts 2000, Tam and Ho 2003). More recently, it has been reported that personalization systems are helping firms considerably boost their sales. For example, Cinematch, the recommender system for Netflix, has been credited with driving 60% of the rentals from Netflix (Thompson 2008), and Google has witnessed a 38% increase in the traffic to its news website by improving its recommender system (Das et al. 2007). These reports point to the growing importance of personalization in e-business environments.

Rule-based approaches are one of the prominent techniques used to provide recommendations for personalization applications (Adomavicius and A. Tuzhilin, 2007). Rule-based systems are popular for a variety of reasons. They can be used in an unobtrusive manner by making recommendations triggered by a customer's actions. For example, when a customer is shopping at a site, then based on the contents of the customers shopping basket, a rule-based system can identify specific products to recommend to the customer. Similarly, if a visitor is traversing pages on a content provider's site, an advertising server can identify relevant advertisements to show to the visitor based on the pages traversed. Importantly, rules are easy to understand, which appeals to marketers interested in cross-selling or product placement; consequently, rules can also be used to justify or explain why specific recommendations are being made. Finally, rules are computationally very efficient for real-time interactions, thus making them an excellent choice for recommendation systems.

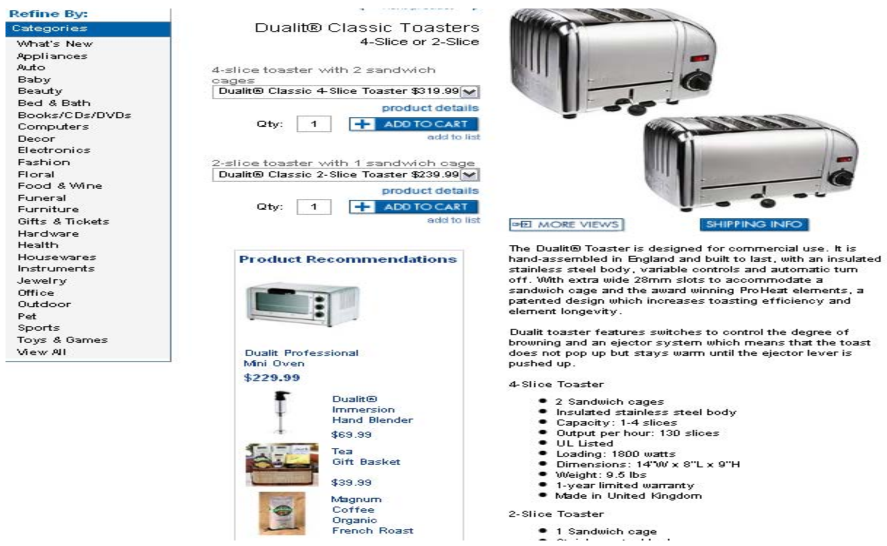
Rules used for target marketing applications are often called association rules. They have been successfully used for market basket analysis (Gordon 2008, Lewin 2009). Several firms use association rules based recommender systems to provide personalized recommendation to customers based on their past purchase history. For instance, Forsblom et al. (2009) develop a mobile application for a Nokia smartphone that uses association rules to recommend retail products to customers. Prominent solution providers like IBM include association rule mining capabilities in business analytics software they develop (IBM 2009a, 2010), and specifically promote the use of association rules for providing online recommendations (IBM 2009b).

Association rules are typically obtained by mining transactional data that records baskets of items purchased by customers (Agrawal et al. 1993). These rules are of the form C1$\rightarrow$ C2, where C1 and C2 are called the antecedent and the consequent of the rule, respectively, and C1 and C2 can comprise of either a single item or a set of multiple items. The number of possible combinations of the items in C1 and C2 can be extremely large, and the mining process attempts to identify only those rules that have a high level of confidence and, further, are supported by a reasonably large number of transactions. Each mined rule is characterized by a confidence parameter and a support parameter, where the values of these parameters must exceed minimum confidence and support thresholds specified by domain experts. The threshold values are used by mining algorithms to identify statistically meaningful rules without having to exhaustively explore all possible combinations of items for the antecedents and the consequents.

An important characteristic of traditional association rules is that the consequents comprise of only conjunctions of items. For example, if a rule mined from a transactional database has two items I1 and I2 in the consequent, then the confidence of the rule will be the probability that both items I1 and I2 appear in a transaction when the items in the antecedent of the rule also appear in a transaction. We refer to such rules as *conjunctive* or *traditional* association rules. As we explain below, conjunctive rules may be of limited use when a site has the ability to recommend multiple items to a user in each interaction (an interaction refers to a user clicking on one of the links displayed on a web page delivered to the user).

For illustration, consider Figure 1 which shows a snapshot of a page from Costco's web site. The page shows details of a toaster, along with recommendations for three other products (a mini oven, an immersion hand blender, and Magnum coffee). If the user is interested in any of the recommended items, the user can evaluate the item in greater detail by clicking on the link associated with that item. When such a link is clicked, additional information about the selected product is displayed, along with recommendations for more items.[1] When recommending a set of items using traditional rules, it is implicitly assumed that either (i) the recommended set is one in which the items are collectively of greatest interest to the user (i.e., the user is interested in every item in the set as a bundle), or (ii) the set consists of those items that are individually of greater interest to the user than other possible items. These approaches do not consider the risk of losing the user if the user does not find any of the recommended items to be interesting. This issue can be of considerable importance to firms. As noted by Rosenberg (2001) and others (Drogan and Hsu 2003, Wang 2008), a goal of personalization is to deliver some piece of content (e.g., an advertisement, product, or piece of information) the user finds sufficiently interesting that the session lasts at least one more click. Considering this, we suggest an alternative objective for firms to consider: recommend a set of items that maximizes the probability that the user will find at least one of the recommended items to be interesting. Such an objective is closely related to a metric called the *bounce rate*, which has recently been identified as very important for evaluating the quality of pages delivered by a site. It is defined as the percentage of users who leave a web page without clicking on any link (Kaushik 2012). James (2011) notes the importance to a site of minimizing this metric in order to improve conversion rates, and goes on to observe that personalization should be used to reduce the bounce rate of a site. This objective is particularly relevant given the nature of web-based interactions, where a user is able to click on only one link at a time in an interaction. This is the objective we examine in this research.

[1] During a session (typically comprising multiple interactions) the user can view and purchase multiple items.



Dualit® Classic Toasters
4-Slice or 2-Slice

4-slice toaster with 2 sandwich cages
Dualit® Classic 4-Slice Toaster $319.99

2-slice toaster with 1 sandwich cage
Dualit® Classic 2-Slice Toaster $239.99

Product Recommendations

Dualit Professional Mini Oven
$229.99

Dualit® Immersion Hand Blender
$69.99

Tea Gift Basket
$39.99

Magnum Coffee Organic French Roast

The Dualit® Toaster is designed for commercial use. It is hand-assembled in England and built to last, with an insulated stainless steel body, variable controls and automatic turn off. With extra wide 28mm slots to accommodate a sandwich cage and the award winning ProHeat elements, a patented design which increases toasting efficiency and element longevity.

Dualit toaster features switches to control the degree of browning and an ejector system which means that the toast does not pop up but stays warm until the ejector lever is pushed up.

4-Slice Toaster

- 2 Sandwich cages
- Insulated stainless steel body
- Capacity: 1-4 slices
- Output per hour: 130 slices
- UL Listed
- Loading: 1800 watts
- Dimensions: 14"W x 8"L x 9"H
- Weight: 9.5 lbs
- 1-year limited warranty
- Made in United Kingdom

2-Slice Toaster

- 1 Sandwich cage

Figure 1: Example page from Costco's web site

Traditional approaches are not designed to maximize this objective. For example, in Figure 1, to recommend a set of three items using traditional approaches, the site could consider conjunctive rules with three item consequents. Alternatively, it could consider recommending items that appear in the consequents of three separate rules. Neither approach is very satisfactory as far as this objective is concerned. In the former approach, because the consequents of the rules are conjunctions, traditional mining algorithms can only identify rules where *all* of the items in the consequent have been viewed/bought in historical transactions. Therefore, the confidence associated with a conjunctive rule is the probability of a user being interested in *all* the items in the consequent of a rule. Finding rules with the desired number of items in the consequent that satisfy the minimum confidence threshold could be difficult in practice (e.g., in Figure 1, all the three items, mini oven, immersion hand blender, and Magnum coffee, would have to appear in a sufficient number of transactions that include a toaster). In addition, even if enough such rules existed, using such a rule to make recommendations will usually be sub-optimal when the firm wishes to minimize the bounce rate.

The second approach (identifying a set of items from separate association rules) may be preferable in this regard. Wang and Shao (2004) propose a method for multiple-item recommendations by considering rules with highest confidences comprising of maximal antecedents,[2] and then recommending the requisite number of distinct consequents of the rules with highest confidences. While this approach is reasonable (and more likely to be used in practice), it also does not attempt to maximize the probability that at least one of the recommended items is of interest to the user.

While the literature on recommender systems is quite extensive,[3] extant approaches have not considered this objective. Our research fills this gap by discovering rules that have in their consequents disjunctions of items (i.e., rules of the form "If I1 then I2 or I3"). The semantics of such rules, that we call *disjunctive consequent rules* (or *disjunctive rules* in short), model this objective naturally and directly. We believe such rules will be very useful to firms that wish to minimize bounce rates for their web pages using recommender systems, and exploit cross-selling opportunities in that manner. We have developed an algorithm that mines disjunctive rules from transactional data. The algorithm first identifies antecedents of potential rules that have a specified support threshold (the antecedents consist of conjunctions of items). For each such antecedent, the algorithm attempts to identify a rule with disjunctive consequent having the desired cardinality and that has a confidence higher than the specified confidence threshold. If no rules satisfy the confidence threshold, the algorithm returns the rule with the highest confidence. Similar to algorithms mining traditional rules, our algorithm is iterative in nature, identifying disjunctive rules with increasing cardinalities starting with single item consequents.

An important difference between disjunctive rules and traditional association rules is that as we go from lower to higher cardinalities, the confidence of the mined rules increase. Consequently, finding disjunctive rules turns out to be inherently a harder problem as compared to mining traditional association rules. In mining traditional rules, rules are derived from itemsets (conjunctive sets of items) that meet the support constraint. As the cardinalities of itemsets increase, the search space reduces. This is because an itemset that is a superset of another has lower support than the latter, ensuring that as higher cardinality itemsets are being mined, only those itemsets need to be considered for which all possible subsets have been found to have the

[2] An antecedent is maximal if it is a subset of the visitor's basket, but none of its supersets present as antecedents of other rules are a subset of the basket.
[3] Additional related works are discussed in Part II of the online supplement.

desired support. This property does not carry over to mining disjunctive rules, drastically increasing the search space and making the problem much more difficult.

While finding disjunctive rules is a hard problem, we show that it is tractable for many real world applications with the help of pruning techniques we have developed. We identify several properties of disjunctive rules that are used to prune the search space when mining such rules. These pruning techniques determine bounds on the probabilities (confidences) associated with potential disjunctive rules, and then use these bounds to eliminate from consideration rules that are dominated by other ones. Experiments demonstrate that the effectiveness of the pruning techniques increases rapidly with an increase in the cardinality of desired rules. Our approach is scalable for large datasets since the complexity of our algorithm is linear in the number of transactions in the dataset. Nevertheless, if a site is interested in recommending a large number of items in each interaction, it may not be feasible to mine disjunctive rules with the desired cardinality. For such situations we develop an efficient approach to identify the desired number of items using disjunctive rules with lower cardinalities.

We conduct experiments to compare the use of disjunctive rules with extant recommendation techniques on three real world datasets (two transactional and one click-stream). Because disjunctive rules can be easily deployed in environments where traditional association rules are currently used, we conduct several experiments to compare these two rule-based approaches. We find that the success rates for recommendations made using disjunctive rules are significantly higher than those made using traditional association rules for every experiment. For the most part, the improvements are greater when more items are recommended in an interaction. We also compare our approach with two other state-of-the-art recommendation approaches – collaborative filtering and matrix factorization. Its performance is generally superior to both these techniques on the two transactional datasets. The relative performance on the click-stream dataset (which is very sparse) is mixed. Its performance is inferior to that of collaborative filtering and superior to that of matrix factorization for that dataset. Our findings suggest that the disjunctive rules approach may be more suitable for market basket applications.

We formally describe the problem in Section 2, and present our methodology to mine disjunctive rules in Section 3. Section 4 presents experiments conducted to examine the effectiveness of the mining algorithm. Section 5 compares recommender systems implemented using disjunctive consequent rules with those using traditional association rules. Section 6 compares our approach with collaborative filtering and matrix factorization. Section 7 provides concluding remarks and discusses areas of future work.

2. Problem Statement

We first illustrate using an example the kind of rules we want to mine from transactional data and then define our problem formally. Consider a dataset with 12 transactions as shown in Table 1, and a recommender system faced with deciding which *two items* to recommend to a customer who has included items I6 and I7 in her shopping basket.

The rules of interest are those that have items I6 and I7 in the antecedent. Therefore, it is desirable that the recommendations are based on transactions where I6 and I7 are both present. The relevant transactions are 1, 2, 3, 4, 6, 8, 9 and 10 (we disregard the other transactions for this example subsequently). Removing items I6 and I7 from these transactions (since these are already in the shopper's basket), we are left with transactions with the items as shown in Table 2.

Table 1: Example Dataset

Transaction ID	Items	Transaction ID	Items
1	I1 I3 I6 I7	7	I1 I4 I6
2	I1 I3 I4 I6 I7	8	I1 I3 I6 I7
3	I1 I5 I6 I7	9	I1 I4 I6 I7
4	I2 I3 I6 I7	10	I2 I4 I6 I7
5	I1 I3 I7	11	I1 I4
6	I1 I3 I5 I6 I7	12	I2 I3 I5

Table 2: Dataset for antecedent (I6, I7)

Row ID	Transaction ID	Items	Row ID	Transaction ID	Items
1	1	I1 I3	5	6	I1 I3 I5
2	2	I1 I3 I4	6	8	I1 I3
3	3	I1 I5	7	9	I1 I4
4	4	I2 I3	8	10	I2 I4

The items appearing in Table 2 are I1, I2, I3, I4, and I5, and the system must choose two of these items to recommend. If the recommendation system used the highest confidence rule with two items in its consequent, it would recommend I1 and I3 as this is the only pair of items that appears in four of the transactions and no pair appears in more transactions. On the other hand, if the recommender system were to sort rules based on their confidences and then pick the consequents of the top two rules (Zaïane 2002), the rules it would use are:

- R1: (I6, I7) $\rightarrow$ I1 (confidence = 0.75), and
- R2: (I6, I7) $\rightarrow$ I3 (confidence = 0.625).

Again, the recommender system would recommend I1 and I3 since they appear as the consequents of rules with the highest confidences.

However, when a firm is interested in finding the two items such that the probability of the customer choosing *at least* one of them is maximized in the next interaction (given the antecedent with items I6 and I7), the system should recommend I1 and I2, because at least one of them is present in all 8 transactions, i.e., the probability is 1 that one of these two items would be chosen by the customer. On the other hand, if the system had recommended items I1 and I3 (as would be the case when using extant approaches) the probability that at least one of these items would be chosen is only 0.875.

The problem is to find, from a given database, rules whose antecedents are conjunctive and consequents are disjunctive. We call these rules *disjunctive consequent rules*, and refer to the set of items in the consequent as a *unionset*. Antecedents having items x_1 to x_m are denoted as $\wedge(x_1,\ldots,x_m)$, or alternatively as $\wedge_{j=1}^{n} x_j$. A unionset with items y_1 to y_n is denoted as $\vee(y_1,\ldots,y_n)$ (or equivalently, $\vee_{j=1}^{n} y_j$). The cardinality of an antecedent (consequent) is the number of items present in it. We use modified notions of *support* and *confidence* as measures of interestingness for disjunctive rules. Our definition of support is as provided by Rastogi and Shim (2002), which is the probability the antecedent of a rule appears in a transaction. Our confidence measure, *disjunctive confidence,* is defined as the probability that *one or more* items in the consequent appears in a transaction in which the antecedent also appears. Mathematically, a rule is: $\bigwedge_{j=1}^{m} x_j \Rightarrow \bigvee_{j=1}^{m} y_j$, with support and disjunctive confidence as shown below.

Support = $P(\bigwedge_{j=1}^{m} x_i) = \frac{\text{Number of transactions}(\wedge_{j=1}^{m} x_i)}{\text{Total Number of transactions}}$, and

Disjunctive confidence = $\frac{P((\wedge_{j=1}^{m} x_j)\wedge(\vee_{j=1}^{n} y_j))}{P(\wedge_{j=1}^{m} x_j)} = \frac{\text{Number of transactions }((\wedge_{j=1}^{m} x_j)\wedge(\vee_{j=1}^{n} y_j))}{\text{Number of transactions }(\wedge_{j=1}^{m} x_j)}$, where $x_1,\ldots,x_m, y_1,\ldots,y_\mathrm{n} \in I$.

Analogous to mining traditional rules, we use specified values of minimum support and *minimum disjunctive confidence* to direct the mining approach. These thresholds are obtained from domain experts. In addition, the experts also need to provide the maximum number of items the system is expected to recommend in an interaction which corresponds to the cardinality of the consequents of mined rules. The mining process first identifies all the itemsets that meet the desired support threshold. These itemsets become potential antecedents for disjunctive rules. For each such itemset, the mining process identifies unionsets that meet the desired disjunctive confidence threshold given the cardinality constraint (i.e., the number of items to recommend).

An important difference between mining traditional association rules and disjunctive consequent rules is if a unionset is a superset of another unionset, the disjunctive confidence associated with the former is higher than that associated with the latter. Thus, as the cardinality of mined unionsets increases, the disjunctive confidence associated with such rules also increases. This can eventually lead to an enormous number of such rules if the cardinality is high. Therefore, the goal is to obtain, for each mined antecedent, at least one rule that reaches the disjunctive confidence threshold given the cardinality. If no such rule is available, our procedure identifies the rule with the highest disjunctive confidence.

3. Disjunctive Consequent Rule Mining

Disjunctive rules are mined in two main steps: (i) finding the relevant antecedents, and (ii) for each antecedent finding a unionset that satisfies the disjunctive confidence threshold. First, we find all the antecedents which meet the minimum support threshold. Any existing itemset mining algorithm (e.g., Apriori) can be used for this purpose. Then, for each antecedent we generate a new dataset from the original dataset that includes those transactions that contain all the antecedent items. We remove from each transaction in this new dataset all the items in the antecedent; we refer to this revised dataset as the *antecedent dataset*. We then mine the antecedent dataset for a unionset that meets the confidence threshold. The confidence for a disjunctive rule is the proportion of transactions in the antecedent dataset that includes one or more items from the unionset. This can be viewed as the support for the unionset in the antecedent database – we refer to this as the *disjunctive support* (D-Sup) for the unionset.

The unionset mining process is iterative in nature. It starts by considering unionsets of size one and checking if the confidence threshold is met. This involves finding the support for each item that appears in the antecedent dataset. If no unionset of size one meets the confidence threshold, unionsets of cardinality two are generated to determine if any of them meet the threshold. To verify this, the disjunctive supports for such unionsets are obtained. This process is repeated by gradually increasing the cardinalities of unionsets considered until either the maximum cardinality threshold is exceeded or the confidence threshold is satisfied by a unionset. If the former condition applies, we are left with the unionset with maximum disjunctive confidence, which is less than the threshold.

An important part of this process is the computation of the disjunctive supports for unionsets with cardinalities greater than one. In principle, the supports for such unionsets can be exhaustively enumerated by using the support of the subsets of the itemsets that correspond to the items in the unionset. For example, for the unionset $\vee(y_1, y_2, y_3)$, we have:

$$\begin{aligned}\text{D-Sup}(\vee(y_1, y_2, y_3)) = {} & \text{Support}(y_1) + \text{Support}(y_2) + \text{Support}(y_3) - \text{Support}(\wedge(y_1, y_2)) - \\ & \text{Support}(\wedge(y_2, y_3)) – \text{Support}(\wedge (y_1, y_3)) + \text{Support}(\wedge(y_1, y_2, y_3)).\end{aligned}$$

This procedure can be very time intensive because of the large number of itemsets to consider for each unionset. For example, if there are 3000 items in an antecedent dataset, we have to enumerate around three trillion itemsets to determine the supports of all cardinality four unionsets. Such a scheme will not scale well for large datasets and high cardinality unionsets. To reduce computational effort, we do the following. First, we use an FP tree representation (Han et al. 2004) to compactly represent the antecedent dataset (the FP Tree creation process is detailed in Part III of the online supplement). We then develop a technique to generate unionsets that are likely to maximize the disjunctive support and compute the disjunctive support of such unionsets using the FP tree. Finally, we identify several pruning rules that reduce the number of unionsets that need to be generated and checked.

3.1. Finding the Disjunctive Support of a Unionset from an FP Tree

Figure 2 shows the FP Tree. Every path starting from the root of the FP tree represents a set of transactions. At each node, the count represents the number of transactions in which that item is present within the set of transactions represented by the path. For each node, its parent corresponds to an item that co-occurs in at least one transaction and has equal or higher support than the child node.

Index		
Item	Abs Sup	Node
I1	6	2
I3	5	3, 6
I4	3	4, 9, 10
I5	2	5, 8
I2	2	7,11

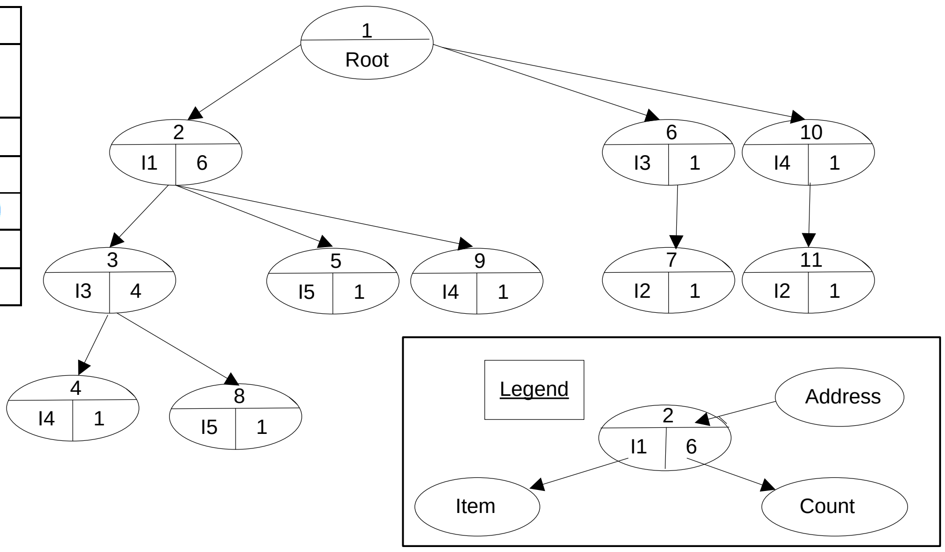


Figure 2: FP tree representation of the antecedent dataset

The main intuition behind the algorithm to determine the disjunctive support of a unionset is to eliminate the possibility of double-counting transactions in which two or more items from the unionset appear. Figure 3 presents the algorithm for a unionset $Y = \vee(y_1, y_2, \ldots y_i)$, where items in Y are arranged in decreasing order of their (individual) supports.

Proposition 1: *Algorithm "FindDisjunctiveSupport" provides the disjunctive support of the unionset. (All proofs appear in the online supplement.)*

To illustrate, we calculate the support of the unionset $\vee$(I1, I3, I2). Initially, the support counter s is set to zero. We start with item I1, which is present only at node 2 of the FP tree. Since only the root node is an ancestor of I1, we add the count of I1 to s resulting in s = 6. Next we consider item I3 which appears at nodes 3 and 6. Node 3 has I1 as an ancestor, therefore it is disregarded. Node 6 does not have any item from the unionset as an ancestor. Therefore, the count for this node (i.e., 1) is added to *s* resulting in *s* = 7. We next proceed to I2 which is present at nodes 7 and 11. Node 7 has I3 as an ancestor, hence this node is disregarded. However, node 11 does not have any item from the unionset as an ancestor. Therefore, the count for this node (i.e., 1) is added to *s* resulting in *s* = 8. We are left with a disjunctive support of 8 for the unionset, which is easily verified from Table 2.

```
Algorithm FindDisjunctiveSupport
Given: Unionset $\vee(y_1,\ldots,y_i)$ and FP tree
1. Create a support counter $s$ and initialize it to zero.
2. For $j = 1$ to $i$, repeat step 3.
3. For each node in the FP tree where $y_j$ appears, repeat step 4.
4. Traversing upwards towards the root, check for nodes corresponding to other items that
   are in the unionset. Because of the construction of the tree, a node corresponding to an
   item with higher support can only be an ancestor node.
        4.a. If no such items are found, add the count of the node to support counter $s$.
        4.b. If an item in the unionset is found in the path to the root, disregard that
             occurrence of $y_j$.
5. The final value of the counter $s$ is the disjunctive support for the unionset.
```

Figure 3: Algorithm to find the disjunctive support for an unionset using the FP tree

3.2. Generating Unionsets and Strategies for Pruning

The idea is to generate and check the disjunctive supports for the unionsets as sparingly as possible. To generate the unionsets, we start with cardinality one, creating a list of unionsets (that are individual items at this stage) with their associated supports.[4] We create a list of these items (i.e., unionsets of cardinality one) arranged in decreasing order of support and denote this list as $\mathcal{Z}$. To consider unionsets of cardinality *i*, we generate sets of *i* items in order of individual supports. This order can help identify unionsets with high disjunctive supports sooner. For

[4] For cardinality one unionsets, the disjunctive supports are the same as traditional supports.

example, if the list $\mathcal{Z}$ is {I1, I3, I4, I2}, the unionsets of cardinality three are generated in the following sequence:

$\vee$(I1, I3, I4), $\vee$(I1, I3, I2), $\vee$(I1, I4, I2), $\vee$(I3, I4, I2).

If all unionsets of cardinality *i* were considered, it would require generating and finding supports of a very large number of unionsets. To reduce the search effort, before determining the disjunctive support of each generated unionset of size *i*, several pruning rules are used to determine if there is any possibility that the unionset's support will exceed the minimum confidence threshold. If not, that unionset need not be explored further. To achieve that, the pruning rules use a lower bound for the disjunctive support of all feasible unionsets. This lower bound is defined as the value of the highest disjunctive support achieved at the current stage of the pruning process and gets updated as the mining proceeds. The pruning rules check whether the upper bound for the disjunctive support of an unionset (which is the disjunctive confidence of the associated rule) is greater than the overall lower bound. If the upper bound for an unionset is lesser, the unionset is pruned along with other unionsets dominated by the pruned unionset (an unionset dominates another unionset when the former's confidence is higher than the latter's). This helps eliminate many feasible unionsets while ensuring the rules with highest confidences are never pruned. If the upper bound for a unionset is greater than the overall lower bound, the disjunctive support for that unionset is calculated and stored along with the unionset in a list we call $\mathcal{V}$ ($\mathcal{V}$ is a subset of the set of all possible unionsets of cardinality *i*).

The overall lower bound value is updated whenever the support of a newly generated unionset is higher than the current lower bound. When the value of the lower bound crosses the confidence threshold, we stop the mining process for that antecedent. If the lower bound does not cross the minimum threshold even after considering all the unionsets of cardinality *i*, unionsets of cardinality *i*+1 are considered. Before starting the mining process for unionsets of cardinality *i*+1, the unionsets in $\mathcal{V}$ are copied into a list $\mathcal{U}_i$ (the explored unionset list of cardinality *i*) and $\mathcal{V}$ is emptied. Unionsets in $\mathcal{U}_i$ are used to prune the unionsets of cardinality *i*+1.

To improve the efficiency of the process, it is important to obtain a good lower bound as quickly as possible. To achieve this, we first generate all cardinality *i* supersets of the unionset of cardinality *i*-1 with highest support (if there are more than one such unionsets, we generate all their cardinality i supersets as well). For example, if Y is a cardinality i-1 unionset with highest support, we consider cardinality i unionsets of the form $(\vee(Y, v))$ where $v \notin Y$. These candidate unionsets are also generated in order of individual item supports and pruned wherever possible. When we are finished with exploring all potential supersets of the cardinality i-1 unionset with highest support, we continue by generating cardinality i unionsets from $\mathcal{Z}$ as described earlier.

3.2.1 Pruning Rules

We next describe the pruning rules. We define $max\text{-}sup_{i-1}$ as the maximum support achieved by a unionset of cardinality i-1. $W = \vee(w_1, \ldots, w_i)$ is a unionset of cardinality i, $1 < i \leq k$, where k is some pre-specified maximum cardinality. Items in W are arranged in order of decreasing individual disjunctive support. LB_i is the current lower bound of disjunctive support for unionsets of cardinality i. It is initialized to $max\text{-}sup_{i-1}$ when unionsets of cardinality i are to be mined, and is updated whenever a better solution is obtained.

Rule 1a: If $max\text{-}sup_{i-1}$ + Support(w_i) < LB_i, then D-Sup(W) < LB_i, and W can be pruned.

Rule 1b: If Rule 1a holds for W and Support(v) $\leq$ Support(w_i), then D-Sup(X) < LB_i for any X where $v \in X$. Hence, all such X can be pruned.

Rule 1a identifies the first condition under which the disjunctive support for W cannot be more than LB_i, and therefore the support of W need not be calculated. Rule 1b states that W dominates all those unionsets which contain at least one item with support less than or equal to w_i. Hence all such dominated unionsets also cannot have disjunctive support more than the lower bound and can be pruned.

Rule 2a: If $\sum_{j=1}^{i} \text{Support}(w_j) < LB_i$, then D-Sup($W$) < LB_i, and W can be pruned.

Rule 2b: If Rule 2a holds for W and Support(v) $\leq$ Support(w_i), then D-Sup(X) < LB_i for any X where $v \in X$ and W-w_i = X-v. Hence, all such X can be pruned.

Rule 2a identifies the second condition under which the disjunctive support for W cannot be more than LB_i. If the sum of the individual supports of items in W is less than LB_i, then the disjunctive support for W must be less than LB_i. Analogous to Rule 1b, Rule 2b also identifies unionsets that W dominates. These dominated unionsets also cannot have disjunctive support more than the lower bound.

Rule 3a: If there exists some j such that $Y = W\text{-}w_j$ and D-Sup(Y) + Support(w_j) $\leq$ LB_i, then D-Support(W) < LB_i, and W can be pruned.

Rule 3b: If Rule 3a holds for a W such that $j = i$, and Support(v) $\leq$ Support(w_i), then D-Sup(X) < LB_i for any X where $X = (\vee(Y,v))$. Hence, all such X can be pruned.

Rule 3a considers each sub-unionset of W (say Y) of cardinality i-1, and searches the list $\mathcal{U}_{i\text{-}1}$ to check if D-Sup(Y) has been calculated earlier. If we have at least one Y such that D-Sup(Y) + D-Sup(W-Y) is less than the lower bound, then the disjunctive support of W can never exceed LB_i. Further, for such a Y, if (W-Y) is the last item of W, then W dominates all those unionsets which differ from W only in the last item and that item has a lower individual support than the last item in W. All such dominated unionsets can also be pruned. Otherwise, only W is pruned.

Proposition 2: *If the conditions for any of the three rules are satisfied for a unionset W, then the disjunctive support for that unionset is less than the current lower bound.*

We note that Rules 1a and 2a are subsumed by Rule 3a. Rule 1a is subsumed because if W is pruned by Rule 1a then $max\text{-}sup_{i\text{-}1}$+ Sup($w_i$) < LB_i. We know that all sub-unionsets of W of cardinality i-1 have support less than or equal to $max\text{-}sup_{i\text{-}1}$. Thus, if Rule 1a holds, Rule 3a must also hold. Rule 2a is subsumed by Rule 3a because if W is pruned by Rule 2a, then $\sum_{j=1}^{i} \mathrm{Sup}(w_j)$ < LB_i. That implies $\sum_{j=1, j\neq l}^{i} \mathrm{Sup}(w_j)$+ Sup($w_l$) < LB_i. We know that D-Sup(W-w_l) $\leq \sum_{j=1, j\neq l}^{i} \mathrm{Sup}(w_j)$, hence D-Sup($W$-$w_l$)+ Sup($w_l$) < LB_i. Hence Rule 3a also holds.

From the above discussion, Rule 1 and Rule 2 could be viewed as redundant. However, Rules 1 and 2 are very easy to check whereas Rule 3 requires searching the sub-unionsets of W in list $\mathcal{U}$. Further, Rules 1 and 2 enable us to easily identify a large number of dominated unionsets. Therefore, for computational efficiency we evaluate Rules 1 and 2 before Rule 3.

As each of the rules can prune more than one unionset, the unionset generation scheme needs to implement a mechanism to recognize and skip the pruned unionsets. Initially, before any unionset is generated, we set a flag at the last item of the list $\mathcal{Z}$. When a unionset is pruned by Rule 1, we check whether the support of the last item of the unionset is greater than the support of the flagged item. If so, we reset the flag to the item in $\mathcal{Z}$ that appears as the last item of the pruned unionset. Thereafter, we do not consider any unionset that includes an item with support less than the flagged item. Similarly, when Rule 2 prunes a unionset, we again check whether the support of the last item of the pruned unionset is greater than the support of the flagged item. If that is the case, we reset the flag to the item in $\mathcal{Z}$ which appears last in the pruned unionset. The flag set by rule 2 is reset back to the last item of $\mathcal{Z}$ when we change the (i-1)th item of the unionset. An example illustrating how the pruning rules are used is provided in Part IV of the online supplement.

As mentioned earlier, we have introduced a process to improve the lower bound before generating unionsets from the list $\mathcal{Z}$. We use the pruning rules during this process also. The sequence of using the rules remains the same, except that Rule 2 is not required for these checks. This is because Rule 2 is subsumed by Rule 1 for the unionsets under consideration. The reason is as follows. We consider the unionsets with highest support from list $\mathcal{U}$. Let Y be a unionset with highest support among all unionsets of cardinality i-1 where $Y \in \mathcal{U}$. The sum of supports of individual items of Y is greater than or equal to $max\text{-}sup_{i-1}$ and therefore if Rule 2a were to hold for any unionset $\vee(Y,v)$, Rule 1a must hold as well. Consequently, Rule 1b also subsumes Rule 2b for these unionsets.

After exploring all potential unionsets of cardinality i, we proceed to cardinality i+1. For a given antecedent, mining stops when either we have found a unionset having disjunctive support more than the minimum threshold, or we have explored/pruned all unionsets of cardinality k. In the former case if i is less than k, adding any item to the unionset ensures that its support is more than the threshold.

3.3. Complexity of the Algorithm

We first show that the problem of determining disjunctive rules belongs to the class of NP-Hard problems. We then show that while our algorithm is exponential in the cardinality of the desired unionset, it is linear in the size of the dataset.

Proposition 3: *Finding a unionset of cardinality less than or equal to k and disjunctive support greater than a threshold t is NP-Hard.*

To obtain the unionset for each identified antecedent, we create an antecedent dataset, generate an FP Tree for the antecedent dataset, and find either (i) a disjunctive consequent with confidence more than the threshold confidence or (ii) the disjunctive consequent with highest confidence given the desired cardinality. The complexity for generating the FP Tree is

$O(T \times n \times \log(n))$ where T is the number of transactions and n is the number of items. In the worst case (when no pruning occurs), the number of unionsets generated is $O(n^k)$ where k is the maximum cardinality of the unionsets. The complexity of determining the disjunctive support of an unionset using the FP Tree is $O(n \times T)$. Hence, the worst-case complexity of our algorithm is $O(n^{k+1} \times T)$. Thus, the algorithm is linear in the number of transactions, and exponential in the number of items to be recommended. The former implies that the approach can scale well for mining large datasets, something that is very desirable. The latter, on the other hand, indicates the approach is limited to mining rules with a modest number of items in the consequent. On examining several retailers' web sites that provide recommendations (e.g., Costco, Sam's Club, Bed Bath and Beyond, Target, and Kohl's), we find that most of these sites recommend between two to four items to a visitor in each interaction. Thus, since k is small for many sites, the algorithm is quite widely applicable. For sites that would like to include a larger number of items in their recommendation list, we discuss in Section 5.1 an approach to combine disjunctive rules.

4. Computational Performance of the Algorithm

We conducted experiments on three real world datasets Retail, BMS-POS, and BMS-2 to analyze the computational performance of the algorithm. The datasets are publicly available at the FIMI repository (http://fimi.cs.helsinki.fi/data/). The dataset Retail is a market basket dataset collected from a Belgian store (Brijs et al. 1999). BMS-POS is a point of sales dataset from a large electronics retailer. BMS-2 is a click-stream dataset from an e-commerce web site. The dataset characteristics are shown in Table 3.

Table 3: Characteristics of the datasets

Dataset Characteristics	Retail	BMS-POS	BMS-2
Total number of items	16,470	1,657	3,340
Total number of transactions	88,162	515,597	77,512
Average transaction length	10.3	6.5	4.6

To mine rules from the three datasets, we used a support threshold of 0.5%, disjunctive confidence threshold of 90%, and maximum cardinality for consequents of size four. For mining the antecedents, we used a publicly available implementation of Apriori.[5] Table 4 shows the statistics summarizing the characteristics of the rules we find after employing the disjunctive consequent rule mining algorithm on the three datasets. The first row shows the number of antecedents obtained after using the Apriori implementation. The second row shows the number of antecedents whose consequents have confidence more than the disjunctive confidence threshold. The third row shows the average time taken per antecedent for the three datasets.

Table 4: Summary statistics for mining disjunctive consequent rules

Results	Retail	BMS-POS	BMS-2
No. of antecedents generated using Apriori (support threshold 0.5%)	580	4,240	408
No. of antecedents for which the rules achieved disjunctive confidence threshold of 90%	106	3,494	220
Average time taken per antecedent (in seconds[6])	5.2	231	7.16

The number of antecedent itemsets generated for the dataset BMS-POS is much higher than that for Retail and BMS-2. This is expected since the number of transactions is much higher in BMS-POS as compared to the other two, whereas the number of items is relatively small (the reason for this is that BMS-POS records category level data in each transaction instead of item-level data and is therefore denser than the other datasets). Therefore, the potential number of itemsets that can cross the support threshold is also very high in BMS-POS. The average time taken to find the unionsets in BMS-POS is also higher than that for the other two datasets.

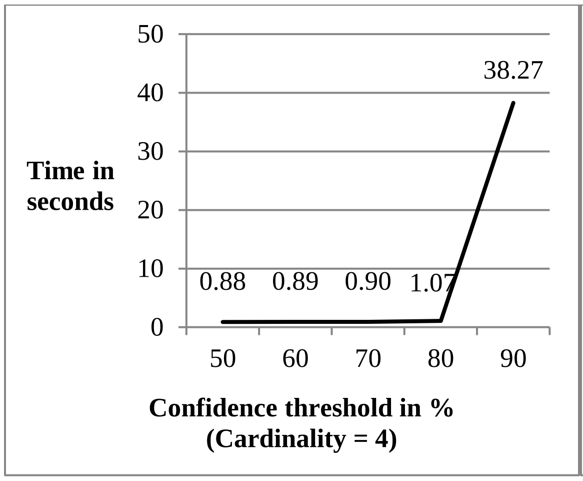
Akin to traditional rules, disjunctive rules will also be mined offline in practice. Hence, the average time taken for mining a rule will usually not be a critical limitation for large real world datasets (unless it is in the order of days or more). Furthermore, mining disjunctive rules for different antecedents can be performed in parallel which can reduce the overall elapsed mining time drastically.

[5] http://www.cs.bme.hu/~bodon/en/apriori/ (version 2.4.7)

[6] The time required to mine disjunctive rules depends on the machine specification. We have used a machine with processor speed 2.8GHz and 1.49 GB RAM. The algorithm is implemented on Java (version jre 1.6.03_03) allocating 700 MB for maximum heap size via command line.

4.1. Effectiveness of the Pruning Rules

In order to understand how efficient the algorithm is, the dataset Retail is mined at different confidence levels. Specifically, we record the times taken to mine disjunctive consequent rules using a support threshold of 0.1%, a unionset cardinality of four, and confidence thresholds of 50%, 60%, 70%, 80%, and 90%, respectively. The results are plotted in Figure 4. The *y* axis shows the average time taken per antecedent, and the *x* axis shows the disjunctive confidence thresholds used, for mining the rules. It is evident from the figure that the average time increases very little when the confidence threshold increases from 50% to 80%. However, there is a jump when the confidence threshold increases from 80% to 90%. This jump in average time for a confidence threshold of 90% is because of the difficulty, for a few of the antecedents, in finding a rule that meets the high threshold.



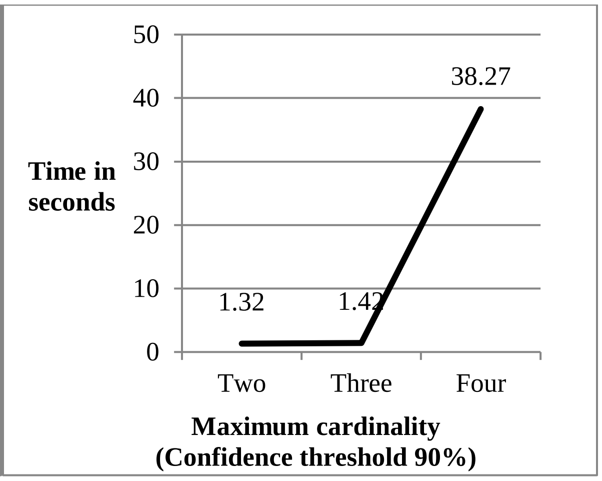


Figure 4: Average time taken per antecedent vs. confidence threshold for mining

Figure 5: Average time taken per antecedent vs. maximum cardinality of consequents

We next examine, for each of the confidence thresholds (and a support threshold of 0.1%), how the average mining time per antecedent changes with an increase in the maximum cardinality of the rules. As expected, the time taken increases with the cardinality for each confidence threshold. This increase is small for confidence thresholds up to 80%. For a confidence threshold of 90% (Figure 5), the average time taken per antecedent does not increase by much when the cardinality increases from two to three; the increase is substantive when the cardinality increases from three to four. This is a result of the combination of two factors: the difficulty in finding rules that meet the high confidence threshold, and the exponential increase in the number of possible combinations of items with increase in cardinality from three to four.

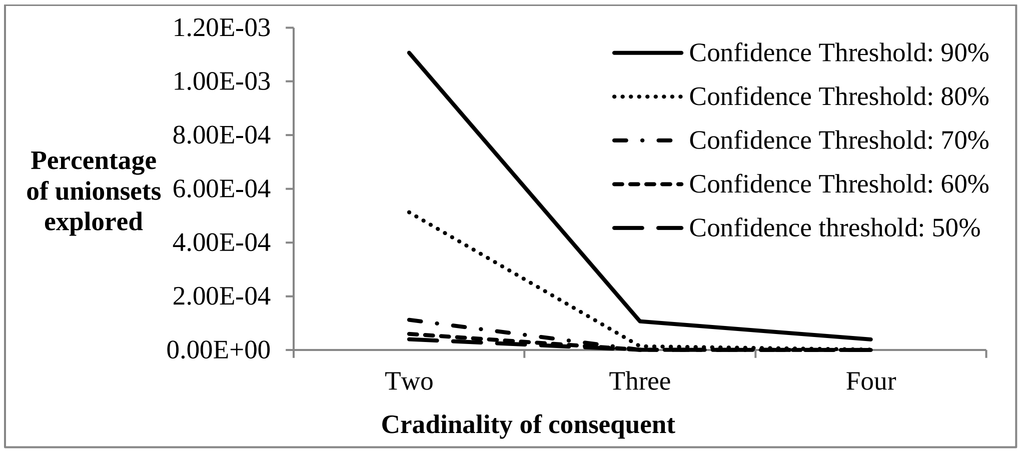


Figure 6: Average percentages of unionsets explored for different cardinalities

We also examine how effective the pruning rules are in eliminating feasible unionsets without requiring their disjunctive confidences to be explicitly computed. Figure 6 illustrates the average proportion of unionsets explored while mining rules with different cardinalities when the confidence thresholds for mining the rules are varied from 50% to 90%.[7] The five lines in the figure correspond to mining rules at these confidence thresholds. We notice that with an increase in the cardinality, the average proportion of unionsets explored reduces drastically for a given confidence threshold. For example, the solid line shows the decrease in the average proportion of explored unionsets with increase in cardinality when the disjunctive confidence threshold used is 90%. Overall, the effectiveness of the pruning rules increases rapidly with an increase in cardinality, with the improvement more marked for higher confidence thresholds which are the more difficult problem instances.

[7] A unionset is considered explored if its disjunctive support is computed from the FP-Tree. The percentage of unionsets explored is the average over all the antecedents for a given cardinality and confidence threshold.

5. Comparing Disjunctive Consequent Rules with Traditional Association Rules

We compare the performance of disjunctive consequent rules with that of traditional (conjunctive) association rules by implementing recommender systems using the two approaches and conducting experiments on the real-world datasets. To recommend multiple items using traditional association rules we consider the approach proposed by Wang and Shao (2004) – this is the benchmark.[8] We examine four different scenarios for the number of items to be recommended in an interaction: two, three, four, and eight. Because it is difficult to mine disjunctive rules with eight items in the consequent, we develop a methodology to combine rules with lower cardinality consequents. We first discuss how rules are combined for the two approaches, and then describe the experimental setup and the results.

5.1 Combining Rule Consequents

The traditional association rules and the disjunctive consequent rules are mined at pre-determined thresholds using a training dataset and stored in the respective systems for use in making recommendations. A list containing all items is also stored in both the systems and arranged in decreasing order of their supports. We refer to this list as the singleton list. This list is used to identify items to include for recommendation when there does not exist enough rules to recommend the desired number of items – this applies to both approaches.

The benchmark recommends *i* items for a given basket on the lines suggested by Wang and Shao (2004). The system first identifies eligible rules, where a rule is eligible if its antecedent is maximal and the items in its consequent are not present in the basket. Then, the *i* eligible rules with highest confidences and distinct consequent items are used to compose the recommendation list. Since the rules used are based on their confidences, rules with multiple conjunctive items in their consequents are always dominated by rules with consequents that

[8] We also considered the approach in Zaïane (2002). Our preliminary analysis provided results similar to Wang and Shao’s approach (differences were not statistically significant). We do not consider the approach suggested by Kim and Kim (2004) because that approach is restricted to using rules with single items in the antecedents as well as in the consequents. The other approaches consider rules that can have multiple items in both the antecedents and the consequents – therefore the rules used by such approaches are a superset of the rules used by Kim and Kim.

include exactly one of those items – therefore, only rules with singleton consequents get used. In case there are only j ($j< i$) eligible rules available for making recommendations, the remaining $i–j$ items are drawn from the singleton list based on the item supports while ensuring that these items are not in the basket or already in the recommendation list.

The consequents of disjunctive rules are combined when the number of items to be recommended exceeds the highest cardinality of the consequents of the mined rules.[9] We use a combination approach analogous to that for the benchmark. The eligible rules are those whose antecedents are maximal subsets of the basket. The items first included in the recommendation list are those that are in the consequent of the eligible rule that has the highest confidence and a consequent with no overlap with the basket. Let j be the cardinality of the consequent of the chosen rule and $\vee(y_1,\ldots,y_j)$ its consequent list, where $j <i$. We then add to the recommendation list, in an iterative manner, the items in the consequent of the eligible rule with the next highest confidence provided the consequent is (i) disjoint with $\vee(y_1,\ldots,y_j)$ and the basket, and (ii) it has a cardinality no more than the space available in the recommendation list. If enough items are not identified in this manner, we consider consequents of eligible rules (in decreasing order of their confidences) that include some items not yet part of the recommendation list or the basket – these items are added next.[10] This process continues until i items have been identified. If i distinct items have not been identified, the singleton list is used to fill the gap.

Although the above approach is not guaranteed to provide the set of items that minimizes the bounce rate, it is very efficient (it is linear in the number of available rules) and easy to implement. It also has a desirable property. If the consequents of rules that are being combined are conditionally independent of each other given the basket, then it can be shown that picking consequents in this sequence dominates picking other consequents of the same cardinality.

[9] It is possible that even when the cardinality of mined rules is equal to the number of items to be recommended in an interaction, the rule with the highest confidence may have fewer than the desired number of items. This occurs if a rule reaches the mining threshold with a consequent consisting of items fewer than the target cardinality. While this is infrequent, when this occurs we use the same approach as discussed for larger recommendation lists. As is also true for such situations, the approach is not guaranteed to be optimal.
[10] An alternative approach is to not look for disjoint consequents in the first place, and add to the recommendation list those items in the consequent of the next eligible rule that have not already been included in the list. We find this approach does not perform as well as the one we have adopted.

5.2 Experimental Setup

We describe here how the experiments are conducted on the dataset Retail. We create a training set and a test set using the original dataset where the training set contains 80% of the transactions chosen randomly from the dataset and the remaining 20% constitute the test set.[11] The training datasets are used to mine the traditional association rules and disjunctive consequent rules at pre-specified thresholds. The transactions from the corresponding test datasets are used to create baskets and provide recommendations. We remove the transactions with only one item from the test datasets, and randomize the sequence of items in the remaining transactions.

We set up the experiment in such a way that it mimics the interactions of a customer at a web site during a session. In practice, every time a customer at a retail web site adds an item to her basket, the site recommends additional items based on the current contents of the basket. In order to replicate this process as well as possible, each transaction in a test dataset is used to create multiple test baskets iteratively. The first test basket created from a transaction contains the first item in the transaction. Each recommendation system makes recommendations for the basket, which are then compared with the remaining items in the test transaction. A recommendation is counted as successful if at least one of the recommended items is present in the remainder of the transaction. In the case of a success, one of the correctly recommended items is randomly selected for addition to the basket to create the next test basket for the recommender system. In case the recommendation does not lead to a success, an item is chosen randomly from the rest of the transaction and added to the existing basket to create the next basket. This process is repeated until the new test basket includes half the items in the transaction.[12] This is done for every transaction in a test dataset. We ensure that the same approach is used for creating baskets for both the recommender systems. [13]

[11] We created multiple such training and testing datasets and conducted experiments on all of them (random subsampling). The results are very similar across all the datasets. Therefore, we report the results obtained from one such training and test dataset.

[12] We have also conducted experiments by creating baskets until all but one item (instead of half the items) of the test transaction are included in the basket. The relative improvement achieved by our approach remains very similar.

[13] Because the two systems are independent, a transaction from the test dataset may generate different baskets for the two recommender systems. For example, when a recommendation leads to a failure, the item added to the existing basket for creating the new one is picked randomly from the items remaining in the transaction. Therefore, a new basket for the two systems may be different even when the previous baskets are identical.

In the experiments, the confidence threshold for disjunctive consequent rules (DCRs) is fixed at 90%. Three different support thresholds are considered: 0.1%, 0.3%, and 0.5%. The traditional association rules (TARs) are also mined at these three support levels. For each support level, five confidence thresholds are considered: 30%, 40%, 50%, 60%, and 70%. We consider multiple confidence thresholds for traditional association rules because these thresholds do not directly correspond to that of the disjunctive consequent rules. For TARs, a confidence threshold greater than 70% results in very few rules while a threshold less than 30% generates many rules with little predictive power. The performance of the disjunctive consequent rules mined with a specific support threshold is compared with the performances of traditional association rules mined with the same support threshold and the five different confidence thresholds.

The worst-case complexity of identifying eligible rules is O($n \times l$) where n is the number of items and l is the total number of available rules (this can be achieved by keeping the items of the basket in a hashtable and checking the presence of items of the antecedents and consequents of the rules in the basket). Given the eligible rules, the complexity of determining items to recommend is O(l). Therefore, the overall complexity is O(nl).

5.3. Results for the Dataset Retail

Table 5 shows the results for our experiments when rules are mined at support thresholds of 0.1% and 0.5%, and confidence thresholds of 30% and 40%, respectively (results for other support and confidence thresholds are provided in Part V of the online supplement). The table shows the percentages of successes when disjunctive consequent rules (DCRs) are used, the percentages of successes when traditional association rules (TARs) are used, and the relative improvement from using the disjunctive rules. The first set of results corresponds to when traditional association rules mined at a confidence threshold of 30% are used for comparison with disjunctive rules. The four rows show the results when two, three, four and eight items were recommended, respectively. Similarly, the other rows correspond to the results when traditional association rules mined at a confidence threshold of 40% are used to recommend items. The number of baskets for which recommendations are made in each experiment is 86,630.

We find that the disjunctive rules consistently outperform traditional association rules in every experiment. This holds for the entire range of confidence thresholds considered for traditional association rules. The improvements are statistically significant at the 99.5% confidence level. The improvement from using disjunctive rules is always higher with an increase in the number of recommended items up to four item recommendations. This is expected as the mined rules explicitly consider consequents of these cardinalities, and items are usually recommended using the rule consequent with the highest probability. As the cardinality of the consequent increases, this probability also increases. On the other hand, such a consideration is not possible when using traditional association rules.

Table 5: Recommendation accuracy from using disjunctive rules (DCR) and traditional rules (TAR). All differences are statistically significant at the 99% confidence level

Confi-dence	# Items Recom-mended	Support Threshold - 0.1%			Support Threshold - 0.5%		
		% Success: DCR	% Success: TAR	Relative Improv-ement	% Success: DCR	% Success: TAR	Relative Improv-ement
30%	2	25.25%	23.95%	5.44%	23.95%	23.22%	3.14%
	3	28.25%	26.38%	7.09%	26.70%	25.69%	3.93%
	4	30.55%	28.11%	8.69%	28.56%	27.42%	4.17%
	8	35.18%	31.35%	12.20%	31.93%	30.69%	4.03%
40%	2	Same as above	23.72%	6.49%	Same as above	23.14%	3.50%
	3		26.16%	8.00%		25.65%	4.07%
	4		27.90%	9.50%		27.39%	4.28%
	8		31.17%	12.85%		30.67%	4.10%

The improvement is also higher for eight item recommendations (relative to fewer recommendations) when rules are mined with a support threshold of 0.1%. When rules mined at 0.5% support thresholds are used the improvement in performance is smaller compared to when four items are recommended. On close examination we find that the fewer rules obtained using the high support threshold leads to considerably more items being recommended using the singleton list on average (instead of from the mined disjunctive rules). This phenomenon (i.e., larger numbers of items being picked on average from the singleton list when higher support thresholds are used) is not observed using traditional rules. As a result, the difference between the performances of the two systems reduces.

The relative improvement from using disjunctive rules falls systematically when the support threshold increases from 0.1% to 0.5%. The reason is similar to that discussed above, i.e., at higher support thresholds fewer rules are available for use in both approaches. This leads to an increased use of the singleton list in making recommendations which negatively impacts the performance of both the approaches. In the extreme case, if no eligible rules were available to either approach, the performance of both approaches would converge to using the singleton list to recommend every item.

We conduct similar experiments on the datasets BMS-2 and BMS-POS; the results are reported in Part VI of the online supplement.

6. Comparisons with Collaborative Filtering and Matrix Factorization

We compare the quality of recommendations obtained using the disjunctive consequent rules approach with those obtained when using collaborative filtering and matrix factorization, two other prominent techniques that are commonly used for making recommendations. For collaborative filtering, we use as our benchmark the item-to-item collaborative filtering approach which is suitable for binary (e.g., transactional) data.[14] For matrix factorization, we use the algorithm popularly known as FunkSVD (Funk 2006). We vary the different design parameters (e.g., model size for collaborative filtering, etc.) when conducting experiments using these two approaches, and report the best results obtained. Training and test datasets are created as discussed in Section 5. We report in Table 6 the results obtained for the dataset Retail; additional details of the two approaches and detailed results of experiments conducted on datasets BMS-POS and BMS2 are provided in Part VII of the online supplement.

[14] The user-to-user collaborative filtering approach is designed for ratings data. The unrated items are treated as missing data and ignored for predicting ratings of users. Therefore, it is not typically used for transactional data.

Table 6: DCR vs. Collaborative Filtering and Matrix Factorization

Dataset	# of items recomm -ended	DCR	Collaborative Filtering (CF)		Matrix Factorization (MF)	
		% Success	% Success	Relative Improvement of DCR over CF	% Success	Relative Improvement of DCR over MF
Retail	2	25.25%	18.87%	33.83%	23.72%	6.45%
	3	28.25%	22.00%	28.43%	27.79%	1.66%
	4	30.55%	24.64%	24.00%	29.63%	3.11%
	8	35.18%	30.00%	17.25%	32.31%	8.89%

DCR provides better recommendations than either collaborative filtering or matrix factorization for all the cardinalities considered (i.e., number of items recommended) on the Retail dataset. The improvement over collaborative filtering is statistically significant at the 99% confidence level for all the cardinalities. The improvement over matrix factorization is statistically significant at the 99 % confidence level when the number of items recommended is two, four or eight; the improvement is statistically significant at the 95% confidence level when three items are recommended.

For the dataset BMS-POS, DCR improves upon collaborative filtering when up to four items are recommended – the improvements are significant at the 99% confidence level when two or three items are recommended, and at the 95% confidence level when four items are recommended (details are provided in Table A4 of the online supplement). Collaborative filtering performs better when eight items are recommended – this difference is statistically significant at the 99% confidence level. DCR performs better than matrix factorization for all cardinalities considered – these differences are all statistically significant at the 99% confidence level.

The collaborative filtering based system performs significantly better (at the 99% confidence level) than both DCR and matrix factorization in the experiments conducted on the dataset BMS-2. We speculate that the inferior performance of DCR is because of the nature of this dataset – it consists of click-stream data (instead of transactional data) and is much sparser than the transactional datasets. This sparsity leads to few rules being mined. The traditional association rule approach (TAR) also performed poorly for this dataset (details are in Table A3 of the online supplement). While DCR improves on the performance of TAR considerably, its performance still falls short of collaborative filtering. Matrix factorization performs very poorly on this dataset – its performance is significantly inferior (at the 99% confidence level) compared to DCR as well. The ability of the matrix factorization approach to generate reliable item-factor and user-factor vectors is likely hampered because of the very sparse nature of BMS-2.

Overall, these experiments show that the performance of DCR matches up very well with two very prominent recommendation techniques in the literature, collaborative filtering and matrix factorization. In particular, its performance is superior to these techniques on the two transactional datasets for all but one experimental setting (where it is superior to matrix factorization but inferior to collaborative filtering). We note that the relative performances of all the approaches vary considerably with the datasets used. Therefore, in addition to considering implementation related issues (e.g., ease of deployment, response times, explanation capabilities, etc.), firms should carefully evaluate the success rates from using different viable recommendation techniques in order to identify the best one for their specific context.

7. Conclusions and Future Research

Traditional rule-based recommendation systems do not consider the risk of losing a user if the user does not find any of the recommended items to be interesting. This can be an important consideration for many sites. In this research, we propose the use of rules to recommend a set of items such that they address the objective of minimizing the above risk. The goal, then, is that in each interaction (i.e., web page delivered by a site) the site will make product recommendations in an attempt to maximize the probability that the user will find at least one of the recommended items to be interesting. Such an objective is closely related to the bounce rate, an important metric to evaluate the quality of pages delivered by a site. To accomplish this goal, we consider rules that have in their consequents disjunctions of items (i.e., disjunctive rules). Such rules provide semantics that map naturally to the stated goal.

We have developed an algorithm that mines such rules from transactional data. Finding disjunctive rules is hard in general. To make the mining task computationally viable, we develop pruning rules to avoid enumerating all possible disjunctive consequents. Experiments to evaluate the effectiveness of our pruning rules show that when the number of items to be recommended increases, the proportion of disjunctive sets explored reduces drastically. The improvement is most marked for very high confidence thresholds, which are the scenarios where the mining is relatively more time consuming. We show that the mining problem is tractable for most retail firms, since such firms often recommend only a few items in each interaction. For firms that wish to recommend a large number of items in each interaction, we develop an efficient approach to combine the consequents of disjunctive rules with lower cardinalities.

Experiments are conducted on three real-world datasets to compare the recommendation performance of disjunctive rules with those of traditional association rules for a range of design parameters. We find the system using disjunctive rules significantly outperforms the system using traditional rules in every experiment. The improvement is higher when more items are recommended. These experiments demonstrate that the proposed methodology is robust across datasets and has the potential to substantially improve firm performance. We also compare the performance of the proposed approach with two other prominent recommendation techniques in the literature – collaborative filtering and matrix factorization. Its performance is generally superior (with one solitary exception) to both these techniques on the two transactional datasets considered for our experiments. The relative performance on the click-stream dataset (which is very sparse) is mixed. Its performance is inferior to that of collaborative filtering and superior to that of matrix factorization for that dataset. We should point out that rule-based systems, by virtue of their ease of comprehension, have an important advantage over collaborative filtering systems which may be perceived as a black box by managers. Rules can also be relatively easily modified to incorporate contextual knowledge directly by practitioners/experts – this flexibility makes such systems very versatile.

A limitation of our mining algorithm is its inability to obtain optimal rules in a reasonably small time when a large number of items are desired in the consequent. The problem becomes even more challenging if the minimum confidence threshold is also very high. While our current methodology is able to make recommendations with a larger number of items by combining the consequents of rules with lower cardinalities, future research could consider developing heuristics in the mining process itself that are computationally efficient and still provide good (if not optimal) solutions.

A limitation with the objective we have studied is that it is myopic in nature, i.e., it does not explicitly consider the possible actions of the user beyond the immediate interaction. For example, the set of links offered in one interaction could impact the choice of the user on pages delivered in subsequent interactions. Also, it is possible that a user may use the back button to revisit a previously viewed page, and click on another link on that page. Future research could model this dynamic aspect of making recommendations in order to improve the conversion rates across an entire session (or even multiple sessions over a given timeframe).

Finally, further research is needed to establish what factors determine which kind of a recommender system will perform better than others. Our experiments clearly demonstrate that the relative performances can be greatly impacted by the application domain. A systematic approach is needed that considers both the underlying theoretical basis for the different approaches and the data characteristics.

Acknowledgements

We would like to thank Michael Ekstrand for his considerable help in using *Lenskit* for the experiments that involve collaborative filtering and matrix factorization.

References

Adomavicius G. and A. Tuzhilin. 2007. Validation Sequence Optimization: A Theoretical Approach. *INFORMS Journal on Computing*, 19(2), Spring 2007, pp. 185–200.

Agrawal, R., T. Imielinski, and A. Swami. 1993. Mining Association Rules between Sets of Items in Large Databases. *In Proc. of the ACM SIGMOD Conference on Management of Data*, Washington, D.C., May 1993, pp. 207-216.

Brijs, T., G. Swinnen, K. Vanhoof, G. Wets. 1999. The Use of Association Rules for Product Assortment Decisions: A Case Study. *KDD-99* San Diego CA USA, 1999.

Das, A., M. Datar, A. Garg, and S. Rajaram. 2007. Google News Personalization: Scalable Online Collaborative Filtering. *Proceedings of the 16th International World Wide Web Conference*, 2007, pp. 271-280.

Drogan, M. and J. Hsu. 2003. Enhancing the Web Customer’s Experience: Techniques and Business Impacts of Web Personalization and Customization. *Proceedings of the Information Systems Education Conference*, 2003.

Funk, S. 2006. Netflix Update: Try This at Home, *http://sifter.org/~simon/journal/20061211.html*. Last accessed on Jan 12, 2013.

Forsblom,A., P. Nurmi, P. Floreen, P. Peltonen and P. Saarikko. 2009. Massive- An Intelligent Shopping Assistant. *Proceedings of the Workshop on Personalization in Mobile and Pervasive Computing*, Trento, Italy, 2009.

Gordon, L. 2008. Leading Practices in Market Basket Analysis. How Top Retailers are Using Market Basket Analysis to Win Margin and Market Share. *Factpoint Group*, 2008.

Han J., J. Pei, Y. Yin, and R. Ma. 2004. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data mining and Knowledge Discovery*, 2004

Häubl G. and V. Trifts. 2000. Consumer Decision Making in Online Shopping Environments: The Effects of Interactive Decision Aids. *Marketing Science*, 19 (1), 2000, pp. 4-21.

IBM. 2009a. IBM SPSS Retail Market Basket Analysis. *ftp://service.boulder.ibm.com/ software /uk/data/ibm-spss-retail-datasheet.pdf*. Accessed on April 11, 2012.

IBM. 2009b. 2012. Retail Market Basket Analysis. *https://www-304.ibm.com/easyaccess /fileserve/?contentid=193973*. Accessed on April 11, 2012.

Lewin, B.A. 2009. Beyond The Grocery and Retail Store: Applying Market Basket Analysis to the Service Industry. *A 1010Data White Paper*, 2009.

James. 2011. How to Increase Your Conversion Rate with Personalization and Discovery. *Boosting E-Commerce*, November 10, 2011.

Kaushik, A. 2012. Standard Metrics Revisited: #3: Bounce Rate, *http://www.kaushik.net/ avinash/standard-metrics-revisited-3-bounce-rate/*, Accessed on March 11, 2012.

Kim C. and J. Kim. 2003. A Recommendation Algorithm Using Multi-Level Association Rules. *IEEE/WIC International Conference on Web Intelligence* (WI'03).

Rastogi R. and K. Shim. 2002. Mining Optimized Association Rules with Categorical and Numeric Attributes. *IEEE Transactions on Knowledge and Data Engineering*, 14(1), January/February, 2002.

Rosenberg M. 2001. The Personalization Story. *IT World.com*, November 5, 2001.

Tam, K. Y. and S. Y. Ho. 2003. Web Personalization: Is it Effective? *IT Professional*, 5(5), 2003, pp. 53-57.

Thompson, C. 2008. If You Liked This, You're Sure to Love That. *The New York Times Magazine*, 2008.

Wang, F.H. and H. M. Shao. 2004. Effective Personalized Recommendation Based on Time-Framed Navigation Clustering and Association Mining. *Expert Systems with Applications*, 27, 2004, pp. 365–377.

Wang, J. 2008. Data Warehousing and Mining: Concepts, Methodologies, Tools and Application, Volume 5. *Information Science Reference*, 1st Edition, 2008, page 2605.

Zaïane, O.R. 2002. Building a Recommender Agent for e-Learning Systems. *Proceedings of the International Conference on Computers in Education*, 2002.

Online Supplement to "Association Rules for Recommendations with Multiple Items"

Abhijeet Ghoshal
Carlson School of Management
University of Minnesota, Minneapolis, MN, 55406

Sumit Sarkar
Jindal School of Management
The University of Texas at Dallas, Richardson, TX, 75080

Part I. Proofs of Propositions

Proof of Proposition 1: We know that D-Sup(y_1) ≤ D-Sup(y_2) ≤…≤ D-Sup(y_i). So items with a higher support may appear as ancestors of items with a lower support but not as descendants. Consider any node corresponding to an item y_j. The count corresponding to that node represents the number of transactions where y_j and its ancestors appear. Thus, if any ancestor y_k of y_j is a part of the unionset, then all the transactions corresponding to y_k have already been accounted for and should not be double counted. Otherwise, the count for the node gets added to the disjunctive support for the unionset. •

Proof of Proposition 2: The proof for each rule is provided separately.

Rule 1a: We know that D-Sup($\vee(w_1, w_2,\ldots, w_i)$) ≤ D-Sup ($\vee(w_1, w_2,\ldots, w_{i-1})$)+ Support($w_i$)
≤ $max\text{-}sup_{i-1}$ + Support(w_i). (i)

The RHS of equation (i) is thus an upper bound for the disjunctive support for W. Hence, if this upper bound is smaller than the current lower bound, then the disjunctive support of the unionset must be smaller than the lower bound.

Rule 1b: If a unionset W' contains w_j' such that Support(w_j') < Support(w_i), then, since we know that D-Sup(W-w_j') ≤ $max\text{-}sup_{i-1}$, it follows that

D-Sup(W-w_j') + Support(w_j') < $max\text{-}sup_{i-1}$ + Support(w_i).

Hence, all unionsets that contain at least one item with support less than or equal to the support of w_i cannot have disjunctive support more than the lower bound. So we do not need to generate any unionset having items with support less than or equal to the support of w_i.

Rule 2a: D-Sup($\vee(w_1, w_2,\ldots, w_i)$) ≤ $\sum_{j=1}^{i} \text{Support}(w_j)$. (ii)

As was the case for Rule 1, the RHS of (ii) is an upper bound for D-Sup(W). Hence, if $\sum_{j=1}^{i} \text{Support}(w_j) \leq LB_i$, then D-Sup(W) < LB_i.

Rule 2b: This is analogous to the proof for Rule 1b. If a unionset *W'* is such that the $W\text{-}w_j' = W\text{-}w_i$ and Support(w_i) > Support(w_j'), then *W'* cannot have support more than LB_i. Therefore, we do not need to generate any unionset containing an item w_j' with individual support less than that for w_i, when all the other items in the unionsets are same as in ($W\text{-}w_i$).

Rule 3a: The sub-unionsets of *W* of cardinality *i*-1are $W-(w_1)$, $W-(w_2)$,…, $W-(w_i)$.

D-Sup(W) ≤ D-Sup($W\text{-}w_j$) + Support(w_j) $\forall\ j=1$ to i. (iii)

For each value of *j* the RHS of equation (iii) becomes an upper bound for D-Sup(W). Hence, if there exists some *j*, such that D-Sup($W\text{-}w_j$) + Support(w_j) < LB_i, then D-Sup(W) < LB_i.

Rule 3b: If *W* is such that D-Sup($W\text{-}w_i$) + Support(w_i) < LB_i and a subset *W'* of *W* exists such that $W'\text{-}\ w_j' = W\text{-}w_i$ and Support(w_i) > Support(w_j'), then it follows that D-Sup($W\text{-}w_j'$) + Support(w_j') < D-Sup($W\text{-}w_i$) + Support(w_i) < LB_i. Thus, *W'* can be pruned. Hence, all the unionsets that differ from *W* only in the last item and are dominated lexicographically can be pruned. •

Proof of Proposition 3: The proof is by reduction using the Partial Vertex Cover problem (Kneis et al. 2008). The Partial Vertex Cover problem is:

Input: A graph $G = (V, E)$, positive integers *k*, *t*

Question: Is there a $C \subseteq V$, $|C| \leq k$, such that *C* covers at least *t* edges?

Given an arbitrary instance of the Partial Vertex Cover problem, we now describe a construction of an instance of our problem.

- *V* is the set of items in the dataset,
- *E* is the set of transactions (i.e., the dataset),
- *C* is an unionset,
- *k* is the user specified maximum cardinality of the unionsets, and
- *t* = minimum disjunctive support threshold.

The decision question for our problem then is:

"Does there exist a unionset $C\subseteq V$, $|C|\leq k$, such that at least one item in the unionset is present in at least *t* transactions in the antecedent dataset?"

It is easy to verify that the construction of the decision problem from an instance of Partial vertex cover can be done in polynomial time. We show that a unionset with *k* or fewer items in it and disjunctive support greater than or equal to the threshold can be found from a dataset with transactions *E* if and only if we can find a partial vertex cover for the graph *G*= (*V*,*E*).

If Part: If we identify a partial vertex cover *C*, then we also identify a unionset *C* of *k* or fewer items. Since the vertices in *C* cover at least *t* edges, the corresponding items in *C* are present in at least *t* transactions. Hence, the disjunctive support of the unionset *C* is greater than or equal to *t* which is the minimum disjunctive support threshold. Thus, we obtain a unionset from the dataset *E* that has *k* or fewer items and has disjunctive support greater than or equal to the threshold.

Only if part: If we identify a unionset C that has k or fewer items in it and has a disjunctive support greater than or equal to t, then we have also found a set of k or fewer vertices that covers at least t edges. Therefore, we have a partial vertex cover.

Part II. Related Work

We review the extant literature focusing on those works that are closely related to ours. The discussion is divided in two parts. The first part describes those works that study rules with disjunction in antecedents and/or consequents. The second part reviews works that focus on providing multiple recommendations.

Starting with Agrawal et al. (1993), an enormous amount of research has been conducted on mining association rules where both the antecedent and consequent comprise of conjunctions of items. We are interested in rules that contain disjunctions of items in their consequents and restrict our discussion to research that considers such rules.[15] Fukuda et al. (1999) developed an algorithm for creating rules where the antecedent is a range of values (i.e., an interval). While an interval can also be viewed as a disjunction, in their approach intervals are present only in the antecedents. Rastogi and Shim (2002) developed algorithms to find association rules on categorical and numerical data, again with disjunctions only in the antecedents of rules. Zelenko (1999) also studied rules with disjunctions in the antecedents.

[15] There is a vast amount of literature on how to improve the efficiency of mining process of conjunctive association rules. Since they are not related to our work, we do not discuss those papers.

Kryszkiewicz and Gajek (2002) developed a procedure for concisely representing frequent itemsets (patterns). In the process they create disjunctive rules whose confidences are found using an exhaustive search strategy; naturally it does not scale well for large datasets. Nanavati et al. (2001) developed an algorithm to generate rules that include disjunctions between itemsets in the consequent for a predefined antecedent. They call such rules generalized disjunctive association rules. Their objective is to identify disjunctive rules that maximize a criterion ρ which is defined as: $\frac{P((I_1 \cup I_2) \cap A)}{P(I_1 \cap A) + P(I_2 \cap A)}$, where A is the pre-specified antecedent and I_1 and I_2 are two itemsets (itemsets can be singletons) mined from the transactions in which the items in A also occur. For example, consider finding a disjunctive rule with antecedent A and a consequent with two items. Their objective would prefer a pair of items X_1 and X_2 over another pair X_3 and X_4 when X_1 and X_2 never co-occur in any transaction and X_3 and X_4 do, even if $P((X_1 \cup X_2) \cap A) < P((X_3 \cup X_4) \cap A)$. In contrast, we would prefer X_3 and X_4 over X_1 and X_2 since the probability of choosing at least one out of X_3 and X_4 is higher than the probability of choosing at least one out of X_1 and X_2, given the antecedent A. Therefore, the algorithm proposed by Nanavati et al. does not generate the type of rules we are interested in.

As discussed earlier, Wang and Shao (2004) have presented a method to recommend multiple items based on conjunctive association rules. Zaïane (2002) suggests multiple items be recommended by sorting the rules based on their confidences and selecting the requisite number of distinct consequences of rules with highest confidences. In an analogous manner, Mobasher et al. (2001) recommend multiple web pages to visitors based on the visitor's sessions, using rules with highest confidences comprising of single consequents and antecedents that are subsets of a customer's session history. Kim and Kim (2003) consider rules with single items in both antecedents and consequents and use a greedy heuristic to recommend multiple items using these rules. None of these papers consider disjunction in rules, and consequently their recommendations do not result from disjunctive rules satisfying some confidence threshold.

Part III. FP Tree Creation

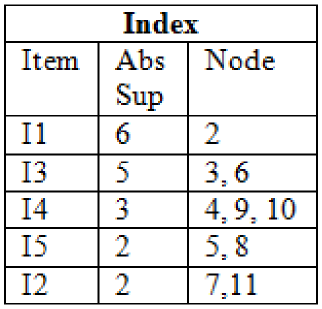
FP trees were originally proposed by Han et al. (2004) for mining traditional itemsets. Such trees can compactly represent a dataset, and are traditionally used to speed up the process of counting the (conjunctive) support of itemsets. We adapt Han et al.'s approach to determine the disjunctive support of unionsets in an efficient manner. We illustrate how an FP tree is created using the example in Section 2 (Han et al. provide the formal procedure for creating the FP tree). To create the tree, items in every transaction of the antecedent database are sorted in decreasing order of their supports in that database. Table A1 shows the transactions in Table 2 sorted in this manner[16] and Figure A1 (which is the same as Figure 2) shows the FP tree for Table A1.

Row ID	Transaction ID	Items	Row ID	Transaction ID	Items
1	1	I1 I3	5	6	I1 I3 I5
2	2	I1 I3 I4	6	8	I1 I3
3	3	I1 I5	7	9	I1 I4
4	4	I3 I2	8	10	I4 I2

Table A1: Dataset for antecedent (I6, I7) with items sorted in transactions

For ease of exposition, we refer to Row ID in Table A1 as the transaction number for creating the FP tree. The FP tree starts from a root node labeled 1, creating a path starting from the root node for each distinct transaction. The first transaction contains {I1, I3}. A node labeled I1 is added as a child to the root node and a node labeled I3 as a child to node I1. Counters for these new nodes are set to one. Transaction 2, consisting of {I1, I3, I4}, shares the path from the root to nodes I1 and I3 with the first transaction. A node corresponding to I4 is added as a child node to I3. The count associated with all the nodes I1, I3, and I4 are incremented by one; thus, the counts for nodes I1 and I3 become two each and the count for I4 becomes one. Transaction 3, which consists of {I1, I5}, leads to adding a child node I5 to node I1, and incrementing the counts of these two nodes by one. For the fourth transaction with {I3, I2}, a new node I3 is created as a child of the root node. A node corresponding to I2 is created as a child to the new node corresponding to I3 and counts are incremented for both these nodes by one. This process is repeated for all remaining transactions. Addresses of all nodes corresponding to an item are stored in the Node column of an index table.

[16] The items in a transaction are sorted while creating the FP tree, thereby requiring only one pass of the database.



Index		
Item	Abs Sup	Node
I1	6	2
I3	5	3, 6
I4	3	4, 9, 10
I5	2	5, 8
I2	2	7,11

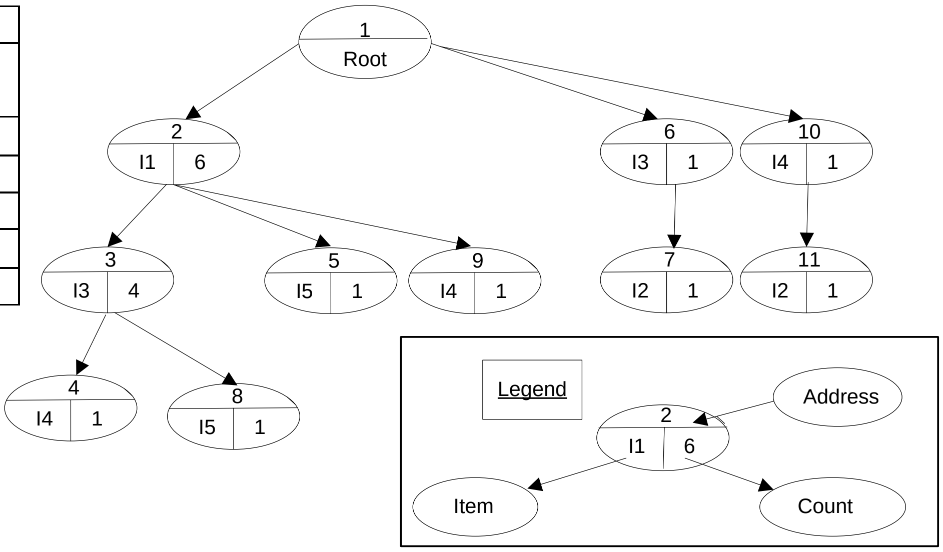


Figure A1: FP tree for dataset in Table A1

Every path starting from the root of the FP tree represents a set of transactions. At each node, the count represents the number of transactions in which that item is present within the set of transactions represented by the path. For each node, its parent corresponds to an item that co-occurs in at least one transaction and has equal or higher support than the child node.

Part IV. Example Illustrating the Pruning Rules

We illustrate the pruning process using the following example. Consider a list $\mathcal{Z}$ with items {I1, I2, I3, I4, I5, I6, I7}, where the items are arranged in decreasing order of support. The individual supports are assumed to be as follows:

Support(I1)=12 Support(I2)=10 Support(I3)=8 Support(I4)=7

Support(I5)=6 Support(I6)=5 Support(I7)=3.

List $\mathcal{U}_2$ contains the support of the cardinality-two unionsets that have been explicitly computed and stored. It is assumed to include:

D-Sup($\vee$(I2, I3))=10 D-Sup($\vee$(I3, I4))=12 D-Sup($\vee$(I2, I4))=15.

The other relevant parameters are assumed to be

max-sup$_2$ = 17 *LB*$_3$ = 22.

We further assume that at the current stage we are considering unionset $\vee$(I1, I6, I7). The pruning process proceeds as follows.

Rule 1a: $max\text{-}sup_2$ + Support(I7) = 17+3 = 20 < LB_3. Therefore, Rule 1a helps prune unionset $\vee$(I1, I6, I7). Furthermore, from Rule 1b, we do not need to consider any unionset with item I7 hereafter.

The next unionset to consider is $\vee$(I2, I3, I4).

Rule 1a: $max\text{-}sup_2$ + Support(I4) = 17 + 7 = 24 > LB_3. So Rule 1 cannot prune this unionset.

Rule 2a: Support(I2) + Support(I3) + Support(I4) = 10 + 8 + 7 = 25 > LB_3. So Rule 2 cannot prune this unionset either.

Rule 3a: This rule generates all the sub-unionsets of $\vee$(I2,I3,I4), and for each sub-unionset, it calculates the sum of the support of the sub-unionset and the support of the remaining item. Since D-Sup($\vee$(I2, I3)) + Support(I4) = 17 < LB_3, the unionset can be pruned.

Rule 3b further prunes the unionsets $\vee$(I2, I3, I5) and $\vee$(I2, I3, I6). It is not necessary to consider $\vee$(I2, I3, I7) as all unionsets with I7 have already been pruned.

The next unionset for consideration is $\vee$(I2, I4, I5). Neither Rule 1 nor Rule 2 can prune this unionset. However, since D-Sup($\vee$(I2, I4)) + Support(I5) = 15 + 6 = 21 < LB_3, the unionset is pruned by Rule 3a. Rule 3b helps prune $\vee$(I2, I4, I6).

Next up for consideration is the unionset $\vee$(I2, I5, I6). This is pruned by Rule 2a.

Unionset $\vee$(I3, I4, I5) is considered next. Rule 1 is not able to prune this unionset. However, Rule 2a does prune this since Support(I3) + Support(I4) + Support(I5) = 8 + 7 + 6 = 21 < LB_3. From Rule 2b, we do not need to consider any unionset containing items I5, I6 or I7 along with items I3 and I4, and no more cardinality three unionsets remain.

Part V. Detailed Results of Experiments on Dataset Retail Comparing Disjunctive Consequent Rules and Conjunctive Consequent Rules

Table A2 provides detailed results of all the experiments conducted to compare the recommender systems using disjunctive consequent rules and the conjunctive association rules. The support thresholds used for mining rules are 0.1%, 0.3% and 0.5%, respectively. The confidence thresholds used for mining conjunctive association rules are 30%, 40%, 50%, 60% and 70%, respectively. The confidence threshold used for mining disjunctive consequent rules is 90%. All improvements are statistically significant at the 99% confidence level.

Confi-dence	# Items Recom-mended	Support Threshold - 0.1%			Support Threshold - 0.3%			Support Threshold - 0.5%		
		% Success: DCR	% Success: TAR	Relative Improv-ement	% Success: DCR	% Success: TAR	Relative Improv-ement	% Success: DCR	% Success: TAR	Relative Improv-ement
30%	2	25.25%	23.95%	5.44%	24.36%	23.39%	4.17%	23.95%	23.22%	3.14%
	3	28.25%	26.38%	7.09%	27.12%	25.79%	5.14%	26.70%	25.69%	3.93%
	4	30.55%	28.11%	8.69%	29.10%	27.54%	5.68%	28.56%	27.42%	4.17%
	8	35.18%	31.35%	12.20%	32.98%	30.77%	7.20%	31.93%	30.69%	4.03%
40%	2	Same as above	23.72%	6.49%	Same as above	23.27%	4.69%	Same as above	23.14%	3.50%
	3		26.16%	8.00%		25.73%	5.40%		25.65%	4.07%
	4		27.90%	9.50%		27.46%	5.97%		27.39%	4.28%
	8		31.17%	12.85%		30.71%	7.41%		30.67%	4.10%
50%	2	Same as above	23.58%	7.09%	Same as above	23.22%	4.91%	Same as above	23.10%	3.66%
	3		26.06%	8.42%		25.69%	5.58%		25.62%	4.21%
	4		27.79%	9.93%		27.41%	6.15%		27.36%	4.39%
	8		31.04%	13.33%		30.66%	7.59%		30.62%	4.27%
60%	2	Same as above	23.45%	7.68%	Same as above	23.21%	4.95%	Same as above	23.11%	3.66%
	3		25.89%	9.13%		25.66%	5.68%		25.63%	4.17%
	4		27.63%	10.59%		27.40%	6.19%		27.36%	4.38%
	8		30.87%	13.94%		30.65%	7.62%		30.62%	4.27%
70%	2	Same as above	23.32%	8.30%	Same as above	23.20%	5.00%	Same as above	23.13%	3.56%
	3		25.75%	9.73%		25.66%	5.67%		25.62%	4.22%
	4		27.47%	11.23%		27.40%	6.20%		27.37%	4.36%
	8		30.72%	14.52%		30.65%	7.62%		30.62%	4.26%

Table A2: Recommendation accuracy from using disjunctive rules (DCR) and traditional rules (TAR). All differences are statistically significant at the 99% confidence level.

Part VI. Comparing Disjunctive Consequent Rules and Conjunctive Consequent Rules on Datasets BMS-2 and BMS-POS

We also performed experiments on the other two datasets (BMS-POS and BMS-2) in order to ascertain if using disjunctive consequent rules led to improved performance for those datasets as well. As was done for the Retail dataset, each of these datasets were divided into training and testing datasets (i.e., 80% of randomly selected transactions from a dataset are included in the training dataset and rest in the testing dataset). The support threshold used for generating rules was set at 0.5% for both approaches. The confidence thresholds used for mining traditional rules were 30%, 40%, 50%, 60%, and 70%, respectively, and it was 90% for disjunctive rules.

The results of these experiments are shown in Table A3. The performance of the recommender system using disjunctive consequent rules is significantly better than the system using traditional association rules in every experiment for each dataset. The relative improvement for dataset BMS-POS is similar in magnitude to that observed for the dataset Retail. Interestingly, the improvements achieved for the dataset BMS-2 are considerably higher. We examined these performances in further detail and observed the following. First, the dataset BMS-2 is much more sparse compared to BMS-POS, which makes it relatively more difficult to obtain reliable rules – this is reflected in the relatively poorer recommendations of both the traditional approach as well as our approach. In addition, because of the fewer rules obtained by the traditional approach (especially at higher confidence thresholds), it ends up using items from the default singleton list far more often than does our approach. These two factors lead to the considerable difference in the performances of the two approaches. We have conducted experiments on BMS-2 at several other support levels, and found the relative improvements to remain about the same.

Confidence Threshold for TAR	# of Items Recom-mended	BMS-POS			BMS-2		
		% Success - DCR	% Success - TAR	Relative Improv-ement	% Success - DCR	% Success - TAR	Relative Improv-ement
30%	2	39.07%	37.67%	3.71%	17.91%	10.50%	70.58%
	3	44.98%	43.30%	3.87%	21.36%	12.10%	76.56%
	4	49.80%	47.67%	4.47%	23.89%	13.29%	79.76%
	8	62.53%	59.15%	5.72%	28.85%	16.57%	74.13%
40%	2	Same as above	36.89%	5.90%	Same as above	9.77%	83.33%
	3		42.41%	6.05%		11.34%	88.39%
	4		46.70%	6.64%		12.29%	94.39%
	8		57.92%	7.96%		15.24%	89.33%
50%	2	Same as above	35.78%	9.18%	Same as above	8.37%	113.99%
	3		41.29%	8.93%		9.78%	118.44%
	4		45.40%	9.70%		10.62%	124.96%
	8		57.27%	9.19%		13.64%	111.54%
60%	2	Same as above	33.62%	16.20%	Same as above	7.41%	141.72%
	3		39.29%	14.47%		8.58%	148.99%
	4		43.96%	13.29%		9.50%	151.48%
	8		56.30%	11.07%		12.52%	130.46%
70%	2	Same as above	32.38%	20.65%	Same as above	6.61%	170.97%
	3		38.59%	16.55%		7.78%	174.60%
	4		43.36%	14.86%		8.76%	172.72%
	8		56.00%	11.66%		11.98%	140.85%

Table A3: Comparing disjunctive consequent rules with traditional rules: BMSPOS and BMS-2. All differences are statistically significant at the 99% confidence level.

Part VII. Comparisons with Collaborative Filtering and Matrix Factorization

We use the implementations provided by Ekstrand et al. (2011) in an open source project named *Lenskit* (lenskit.grouplens.org). The implementations are refinements of the approaches proposed by Deshpande and Karypis (2004) for collaborative filtering and by Funk (2006) for FunkSVD. As noted by Ekstrand et al., Lenskit provides carefully tuned implementations of the leading algorithms. We should mention that in their experiments on three separate datasets, Ekstrand et al. find FunkSVD to perform the best on two datasets and the item-to-item collaborative filtering approach to perform the best on the third.

We provide here brief descriptions of the collaborative filtering and matrix factorization (FunkSVD) approaches. Additional details for collaborative filtering are available in Deshpande and Karypis (2004), for FunkSVD in Funk (2006) and Koren (2009), and for the implementations in Ekstrand et al. (2011). For collaborative filtering, given a dataset involving m items, the process requires two parameters as inputs – a *model size* (k) and a *neighborhood size* (l). The system computes the scores for items by multiplying an $m \times m$ similarity matrix (model) with a column vector representing the basket that contains 1 for items present in the basket and 0 for other items. The model size is the number of similarities retained in each column of the model; other similarities are set to 0. The neighborhood size is the number of similarities used to calculate the score of an item; other similarities are ignored. The item with the highest rating is recommended.

The matrix factorization technique determines latent factors (characteristics) using a technique analogous to singular value decomposition (SVD). It associates each user with a user-factor vector and each item with an item-factor vector, and makes predictions using the inner product of such vectors. Parameters of the model are learned with the objective of minimizing the differences between predicted and actual ratings while avoiding over-fitting (Koren et al. 2009). FunkSVD accomplishes this using a stochastic gradient descent learning algorithm.

We conduct experiments on all the three datasets discussed earlier. The experimental setup, i.e., creating training and test datasets and the corresponding baskets, is the same as for the experiments in Section 6. For the dataset Retail, we conduct experiments by mining disjunctive consequent rules at three different support thresholds: 0.1%, 0.3% and 0.5%. We conduct experiments on the dataset BMSPOS by mining rules at 0.2% and 0.5% support thresholds, respectively. Experiments on the dataset BMS-2 are conducted using rules mined at support thresholds of 0.1% and 0.5%. The confidence threshold for mining disjunctive rules is set at 90% in all the experiments for each dataset. For the dataset Retail, the best results were obtained when rules were mined at the support threshold 0.1. The best results for BMSPOS and BMS-2 were obtained when the rules were mined at support thresholds 0.2% and 0.1%, respectively. We report the results for these support thresholds.

For the collaborative filtering system, we experimented with values of model sizes up to 500 and neighborhood sizes up to 150 for all the datasets (larger model and neighborhood sizes could not be accommodated in memory). The performance of the collaborative filtering system was not very sensitive to changes in the model and neighborhood sizes. We report the results for the experiments with model size 500 and neighborhood size 150 since the system's accuracies are a little better with these settings than with the other settings. The similarity between items is measured using cosine similarity metric.

While FunkSVD was originally designed for non-zero ratings for user-item pairs (like any matrix factorization technique), it has subsequently been found to work well for binary data by replacing all zeros with a small number (XLVector 2012). Therefore, we modify the datasets in this manner to run FunkSVD. The resulting datasets are dense and cannot be used in their entirety by Lenskit. Therefore, from the training part of the Retail dataset, we randomly select two thousand transactions for model building. Similarly, from the training part of BMS-2, we randomly select four thousand transactions and from that of BMS-POS, we are able to use twelve thousand transactions (again randomly selected) as the training datasets for model building. We are able to select more transactions for BMS-POS because this dataset has much fewer items than the other two. All the modified training datasets have more than ten million values for user-item pairs (the largest dataset used by Ekstrand et al. has ten million values). We experimented with other (smaller) numbers of randomly selected transactions for creating ratings datasets – the results do not differ significantly.

Table A4 provides the results of experiments conducted on BMS-POS and BMS-2.

Dataset	# of items recomm-ended	DCR	Collaborative Filtering (CF)		Matrix Factorization (MF)	
		% Success	% Success	Relative Improvement by DCR over CF	% Success	Relative Improvement by DCR over MF
BMS-POS	2	40.09%	38.37%	4.49%	32.15%	24.72%
	3	45.93%	45.15%	1.74%	38.31%	19.91%
	4	50.88%	50.17%	1.41%	43.13%	17.97%
	8	63.26%	64.69%	-2.20%	56.12%	12.72%
BMS-2	2	32.33%	34.42%	-6.08%	4.99%	548.46%
	3	37.77%	41.00%	-7.89%	6.42%	488.26%
	4	41.83%	45.77%	-8.60%	7.78%	437.71%
	8	50.07%	57.00%	-12.16%	11.39%	339.60%

Table A4: DCR vs. Collaborative Filtering and Matrix Factorization

References

1. Agrawal, R., T. Imielinski, and A. Swami. 1993. Mining Association Rules between Sets of Items in Large Databases. *In Proc. of the ACM SIGMOD Conference on Management of Data,* Washington, D.C., May 1993, pp. 207-216.

2. Deshpande, M. and G. Karypis. 2004. Item-based Top N Recommendation Algorithms. *ACM Transactions on Information Systems*, 22(4), pp. 143-177.

3. Ekstrand, M.D., M. Ludwig, J.A. Konstan and J. T. Riedl. 2011. Rethinking the Recommender Research Ecosystem: Reproducibility, Openness, and LensKit. *Proceedings of the fifth ACM conference on Recommender systems*, 2011.

4. Fukuda T., Y. Morimoto, and S. Morishita, T. Tokuyama. 1999. Mining Optimized Association Rules for Numeric Attributes. *Journal of Computer and System Sciences*, 1999.

5. Kim C. and J. Kim. 2003. A Recommendation Algorithm Using Multi-Level Association Rules. *IEEE/WIC International Conference on Web Intelligence (WI'03)*.

6. Kneis J., A. Langer, and P. Rossmanith. 2008. Improved Upper Bound for Partial Vertex Cover. *Lecture Notes in Computer Science*, 5344, 2008, pp. 240-251.

7. Koren, Y., Bell, R. and Volinsky, C. 2009. "Matrix Factorization Techniques for Recommender Systems", *IEEE Computer* (42:8), pp. 30-37, August, 2009.

8. Kryszkiewicz M. and M. Gajek.2002. Concise Representation of Frequent Patterns Based on Generalized Disjunction-Free Generators. *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, 2002.

9. Mobasher B., H. Dai, T. Luo, and M. Nakagawa. 2001. Effective Personalization Based on Association Rule Discovery from Web Usage Data. *WIDM 2001*, Atlanta, GA, USA.

10. Nanavati A.A., K. I. Chitrapura, S. Joshi, and R. Krihnapuram. 2001. Mining Generalized Disjunctive Association Rule. *CIKM'01*, November 5-10, Atlanta, GA, USA.

11. Rastogi R. and K. Shim. 2002. Mining Optimized Association Rules with Categorical and Numeric Attributes. *IEEE Transactions on Knowledge and Data Engineering*, 14(1), January/February, 2002.

12. Wang, F.H. and H. M. Shao. 2004. Effective Personalized Recommendation Based on Time-Framed Navigation Clustering and Association Mining. *Expert Systems with Applications*, 27, 2004, pp. 365–377.

13. XLVector. 2012. XLVector – Recommender System. *http://xlvector.net/blog/?p=465*. Last accessed on Jan 12 2013.

14. Zaïane, O.R. 2002. Building a Recommender Agent for e-Learning Systems. *Proceedings of the International Conference on Computers in Education*, 2002.

15. Zelenko, D. 1999. Optimizing Disjunctive Association Rules. *Principles of Data Mining and Knowledge Discovery* , LNAI 1704, pp. 204−213, 1999.