Neighborhood-Restricted Mining and Weighted Application of Association Rules for Recommenders

Fatih Gedikli and Dietmar Jannach

Technische Universität Dortmund,
44221 Dortmund, Germany
{firstname.lastname}@tu-dortmund.de

Abstract. Association rule mining algorithms such as Apriori were originally developed to automatically detect patterns in sales transactions and were later on also successfully applied to build collaborative filtering recommender systems (RS). Such rule mining-based RS not only share the advantages of other model-based systems such as scalability or robustness against different attack models, but also have the advantages that their recommendations are based on a set of comprehensible rules. In recent years, several improvements to the original Apriori rule mining scheme have been proposed that, for example, address the problem of finding rules for rare items. In this paper, we first evaluate the accuracy of predictions when using the recent IMSApriori algorithm that relies on multiple minimum-support values instead of one global threshold. In addition, we propose a new recommendation method that determines personalized rule sets for each user based on his neighborhood using IMSApriori and at recommendation time combines these personalized rule sets with the neighbors' rule sets to generate item proposals. The evaluation of the new method on common collaborative filtering data sets shows that our method outperforms both the IMSApriori recommender as well as a nearest-neighbor baseline method. The observed improvements in predictive accuracy are particularly strong for sparse data sets.

1 Introduction

Association rule mining is a popular knowledge discovery technique which was designed as a method to automatically identify buying patterns in sales transactions, or, in a more broader view, to detect relations between variables in databases. One of the earliest efficient techniques to find such rules is the Apriori algorithm proposed by Agrawal and Srikant in [1]. A common example of an association rule that could be found in the sales transactions in a supermarket could be [2]: $cheese \Rightarrow beer \quad [support = 10\%, confidence = 80\%]$ which can be interpreted that in 10% of all transactions beer and cheese were bought together (support of the rule) and that in 80% of the transactions, in which cheese was bought, also beer was in the shopping basket (confidence of the rule). Confidence and support are thus statistical measures that indicate the "strength" of the pattern or rule.

L. Chen, P. Triantafillou, and T. Suel (Eds.): WISE 2010, LNCS 6488, pp. 157–165, 2010.

Quite obviously, the knowledge encoded in such automatically detected association rules (or frequent itemsets) can be exploited to build recommender systems (RS). Since the rules can be mined in an offline model-learning phase, rule mining-based approaches do not suffer from scalability problems like memory-based algorithms [3]. A further advantage of these approaches lies in the fact that association rules are suitable for explaining recommendations.

Regarding the predictive accuracy of rule mining-based approaches, previous research has shown that the accuracy of rule mining-based recommenders is comparable to nearest-neighbor (kNN) collaborative filtering approaches. However, using the original Apriori algorithm can lead to the problem of reduced coverage as shown in [3]. This phenomenon can be caused by the usage of a global minimum support threshold in the mining process, which leads to the effect that no rules for rare items can be found. Lin et al. [4] therefore propose an "adaptive-support" method, in which the minimum support value is determined individually for each user or item (depending on whether item associations or user associations are used). Their experiments show a slight increase in accuracy when compared with the baseline kNN-method.

More recently, Kiran and Reddy [5] proposed a new method called IMSApriori that uses a particular metric to determine appropriate minimum support values per item (see also [2]) in order to mine rare itemsets; their experiments indicate that this method is better suited to mine rare itemsets than previous methods. An evaluation of the approach for recommendation purposes has, however, not been done so far.

In this work, we evaluate the predictive accuracy of a recommender system based on the IMSApriori algorithm and describe our extension to the *Frequent Itemset Graph* used in [6] for enabling a fast recommendation process. In addition, we propose a new scheme for association rule-based recommendation called NRR (*Neighborhood-Restricted Rule-Based Recommender*), which is based on the idea to learn a personalized set of rules for each user based on his nearest neighbors and not based on the whole database of transactions. Similar to kNN-approaches, the underlying idea of this is that close neighbors will be better predictors than others. After the model-building phase, the user's personalized knowledge base is at recommendation time combined with the rule sets of his nearest neighbors to generate recommendation lists.

2 Algorithms

In the following we will shortly summarize the ideas of the IMSApriori algorithm in order to give the reader a quick overview of the algorithm parameters that were varied in the experimental evaluation. In addition, we will describe how the *Frequent Itemset Graph* proposed, e.g., in [6], has to be extended for a recommender based on IMSApriori.

2.1 IMSApriori

In order to deal with the problem of "missing rules" for rare, but interesting itemsets, different proposals have been made in literature. IMSApriori [5], which is used in this work, is a very recent one that builds on the idea of having several minimum support thresholds, an idea also proposed earlier as MSapriori in [2]. The general idea is to calculate a minimum item support (MIS) value for each item with the goal to use a lower support threshold for rare itemsets. In [2] a user-specified value β (between 0 and 1) is used to calculate a MIS value based on the item's support and a lower support threshold value LS as $MIS(item) = max(\beta \times support(item), LS)$. In order to be counted as a *frequent* itemset, itemsets containing only frequent items have to pass a higher minimum support threshold than itemsets consisting of frequent and rare or only rare items. Thus, rare itemsets are found when using a low value for LS while at the same time not too many uninteresting, but more frequent rules are accepted.

Recently, in [5], a different approach to calculate the MIS values was proposed because MSapriori fails to detect rare itemsets in situations with largely varying item support values. This phenomenon can be attributed to the fact that due to the constant proportional factor β the difference between the item support and the MIS value decreases when we move from frequent to rare items. The main idea of the *improved* MSapriori (IMSApriori) is therefore the use of the concept of "support difference" (SD) to calculate MIS values as $MIS(item) = max(support(item) - SD, LS)$. SD is calculated as $SD = \lambda(1 - \alpha)$, where λ is a parameter "like mean, median, mode, maximum support of the item supports" and α is a parameter between 0 and 1. The net effect of the support difference concept is that the difference between item support values and the MIS values remains constant so that rare items can also be found in data sets with strongly varying item supports. Finally, in this approach, an itemset is considered to be frequent if its support is higher than the minimum of the MIS values of its components. Regarding the generation of candidates, it has to be noted that the Apriori assumption that all subsets of frequent itemsets are also frequent does not hold and that a different algorithm for finding frequent itemsets has to be used.

2.2 The Neighborhood-Restricted Rule-Based Recommender (NRR)

The idea of the herein proposed NRR algorithm is to learn personalized rule sets for each user in an offline phase and to exploit these rule sets in combination with the neighbor's rule sets to generate more accurate predictions. The algorithm is summarized in Algorithm 1. The parameters of the algorithm include – beside the IMSApriori parameters – two neighborhood sizes (for rule learning and for the prediction phase). In the online phase, the calculated user-specific frequent itemsets (UserFISs) of the target user and of the neighbors of the target user are used to calculate predictions using the *Extended Frequent Itemset Graph* (EFIG) which is introduced in the next section. The resulting confidence scores

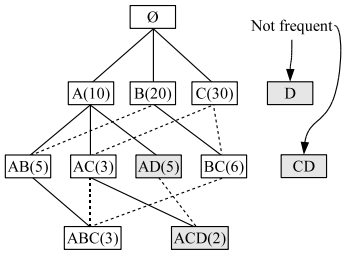
Algorithm 1 NRR (sketch)

In: user, ratingDB, learnNeighborSize, predictNeighborSize, λ, α
Out: recommendedItems
(Offline:) UserFISs = CalcUserFISsIMSApriori(ratingDB, learnNeighborSize, λ, α)
neighborhood = user $\cup$ findNeighbors(user, predictNeighborSize, ratingDB)
recommendedItems = $\emptyset$
for all u $\in$ neighborhood **do**
  userRecs = Recommend(u, buildEFIG(UserFISs(u)))
  weightedUserRecs = adjustConfidenceScoresBySimilarity(userRecs, user, u)
  recommendedItems = recommendedItems $\cup$ weightedUserRecs
end for
recommendedItems = sortItemsByAdjustedScores(recommendedItems)

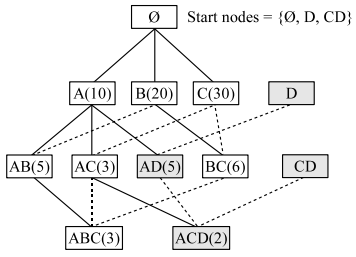
are weighted according to the similarity of the target user and the neighbor (using Pearson correlation as a metric). These user-specific predictions are finally combined and sorted by the weighted confidence scores.

2.3 The Extended Frequent Itemset Graph

The *Frequent Itemset Graph* (FIG) is a data structure (a directed acyclic graph) proposed in [6], which can be used by a recommendation engine to make real-time recommendations without learning explicit association rules first. Figure 1(a) shows such a graph in which the elements of the frequent itemsets are lexicographically sorted and organized in a tree structure where the size of the itemsets are increased on each level. Note that the numbers in brackets stand for the support values of the itemsets. Given, for example, a set of past transactions $T = \{A, D\}$ of user u, recommendations can be produced by traversing the tree in depth-first order and looking for (single-element) supersets of $\{A, D\}$ in the graph. In the example, given the superset $\{A, D\}$, C could be recommended to u if the recommendation score of item C is high enough. The recommendation score of this item corresponds to the confidence value $\frac{support(\{A,D\} \cup \{C\})}{support(\{A,D\})} = \frac{2}{5}$ which



(a) Example for a Frequent Itemset Graph.



(b) *Extended* Frequent Itemset Graph example.

Fig. 1. Extended Frequent Itemset Graph approach

is at the same time the confidence value of the association rule $\{A, D\} \Rightarrow \{C\}$. The solid arrows in the figure indicate how the graph would be traversed in depth-first order.

Since the assumption that "any subset of a frequent itemset must be frequent" does not hold when using multiple minimum-support values, the standard FIG-based method has to be extended. Let us assume that in the example Figure 1(a) the itemsets $\{D\}$ and $\{C, D\}$ are not frequent, although they are subsets of the frequent itemset $\{A, D\}$ and $\{A, C, D\}$ respectively. We could therefore not recommend $\{A\}$ to users who purchased $\{C, D\}$ or $\{D\}$ alone although this would be plausible.

We propose to solve this problem by extending the FIG in a way that it also contains all subsets of the frequent itemsets and connect these additional nodes with their supersets as shown in Figure 1(b). In order to find frequent itemsets like $\{A, C, D\}$ from $\{C, D\}$ we re-start the depth-first search on the not-yet-visited parts of the subgraph beginning from the additional nodes.

Algorithm 2 shows the steps that are necessary for building the *Extended* Frequent Itemset Graph (EFIG).

Algorithm 2 Build EFIG.

```
1: for k = n to 1 do
2:    for all frequent k-itemsets f do
3:       for all k − 1 subsets s of f do
4:          add new edge (s, f);
5:          if s ∉ L_{k−1} then
6:             add new start node s;
7:          end if
8:       end for
9:    end for
10: end for
```

The algorithm works as follows. The EFIG is constructed bottom-up, starting with all frequent itemsets of size k, beginning with the maximum size n of the frequent itemsets (lines 1-2). Lines 3 and 4 connect each $k-1$ subset s of a k-frequent itemset with its superset f. If subset s is not a frequent itemset, i.e., $s \notin L_{k-1}$ where L_{k-1} contains all frequent itemsets of length $k-1$, s will be added to the set of *start nodes* which contains all possible entry-points to the graph. This set consists of the root node ($\emptyset$, the original start node) and all not frequent itemsets that are subsets of frequent itemsets. In the example Figure 1(b), the nodes D and CD are finally also included in the start nodes and can be considered in the recommendation process.

3 Experimental Evaluation

The proposed NRR algorithm has been evaluated in an experimental study on different data sets. In particular, the predictive accuracy was measured using

different sparsity levels and compared to (a) a recommender based on IMSApriori and a classical prediction scheme and (b) the user-based k nearest neighbor method using the Pearson correlation coefficient (kNN). In the following, we will summarize the findings of this evaluation.

3.1 Experimental Setup / Evaluation Metrics

Data sets. As data sets for the evaluation, we used the MovieLens rating database consisting of 100,000 ratings provided by 943 users on 1,682 items and a snapshot of the Yahoo!Movies data set containing 211,231 ratings provided by 7,642 users on 11,915 items[1]. In order to test our NRR scheme also in settings with low data density, we varied the density level of the original data sets by using subsamples of different sizes of the original data set. Four-fold cross-validation was performed for each data set; in each round, the data sets were split into a 75% training set and a 25% test set.

Accuracy metrics. In the study, we aim to compare the predictive accuracy of two rule mining-based methods and the kNN-method. We proceed as follows. First, we determine the set of existing "like" statements (ELS) in the 25% test set and retrieve a top-N recommendation list with each method based on the data in the training set[2]. In the kNN-case, the rating predictions are converted into "like" statements as described in [3], where ratings above the user's mean rating are interpreted as "like" statements. The set of predicted like statements returned by a recommender shall be denoted as *Predicted Like Statements (PLS)*.

We use standard information retrieval accuracy metrics in our evaluation. *Precision* is defined as $\frac{|PLS \cap ELS|}{|PLS|}$ and measures the number of correct predictions in PLS. *Recall* is measured as $\frac{|PLS \cap ELS|}{|ELS|}$ and describes how many of the existing "like" statements were found by the recommender.

In the evaluation procedure, we use "top-10", that is, the list of the top ten movies for a test user with predicted rating values above the user's mean rating, and calculate the corresponding precision and recall values for all users in the test data set. The averaged precision and recall values are then combined in the usual F-score, where $F = 2 * \frac{precision * recall}{precision + recall}$.

Algorithm details and parameters. For the neighborhood-based algorithms, we used Pearson correlation as a similarity metric both for the kNN-baseline method and for determining the neighborhood in the NRR algorithm. For the kNN-method, we additionally applied *default voting* and used a neighborhood-size of 30, which was determined as an optimal choice in literature.

The IMSApriori implementation used in the experiments corresponds to above-described algorithm and learns the rules from the whole database of

[1] `http://www.grouplens.org/node/73`, `http://webscope.sandbox.yahoo.com`

[2] The top-N recommendation lists are created either based on the confidence of the producing rule or based on the prediction score of the kNN-method.

transactions. Recommendations are generated by using the Extended Frequent Itemset Graph structure.

For the NRR method, two further parameters can be varied: *neighborhood-size-learn* is the number of neighbors used to learn association rules; *neighborhood-size-predict* determines on how many neighbors the predictions should be based.

The sensitivity of these parameters was analyzed through multiple experiments on the MovieLens and Yahoo! data set with different density levels. The parameter values for both data sets and all density levels were empirically determined to be 900 and 60 for *neighborhood-size-predict* and *-learn* respectively.

In order to establish fair conditions in our study, we have used individual, empirically-determined LS values for each rule learning algorithm (IMSApriori: 3%; NRR: 9%), which we have then used for all density levels and data sets.

3.2 Results

Figure 2 summarizes the evaluation results for the three algorithms kNN, IMSApriori and NRR. The results show that our NRR algorithm consistently outperforms the IMSApriori method on the F1-measure and is better than the kNN algorithm in nearly all settings for both data sets. The observed accuracy improvements are particularly high for low density levels, i.e., for sparse data sets. With higher density levels, the relative improvements become smaller for both data sets.

As a side-observation, we can see that the pure IMSApriori version does not always reach the accuracy level of the kNN-method, especially in settings with lower and medium density levels. However, the herein proposed NRR algorithm,

		Density →	10%	20%	30%	40%	50%	60%	70%	80%	90%
MovieLens	F1	kNN	30,76	41,48	48,06	51,25	55,39	57,13	58,89	59,99	61,32
		IMSApriori	3,75	32,80	50,00	55,94	59,34	60,97	62,66	62,84	62,82
		NRR	37,14	44,09	50,80	56,54	59,10	61,38	63,10	63,51	63,57
	Precision	kNN	39,85%	53,62%	59,76%	61,73%	64,44%	65,29%	66,18%	66,61%	67,07%
		IMSApriori	5,92%	48,43%	63,03%	64,91%	65,31%	65,35%	65,70%	65,24%	64,48%
		NRR	47,25%	57,73%	63,62%	65,74%	65,50%	65,49%	65,69%	65,30%	64,75%
	Recall	kNN	25,05%	33,83%	40,21%	43,81%	48,57%	50,78%	53,05%	54,57%	56,47%
		IMSApriori	2,76%	24,81%	41,43%	49,15%	54,38%	57,15%	59,90%	60,61%	61,24%
		NRR	30,60%	35,67%	42,28%	49,60%	53,84%	57,75%	60,71%	61,81%	62,42%
YahooMovies	F1	kNN	9,24	15,93	21,91	27,95	33,30	37,53	41,32	44,02	46,29
		IMSApriori	6,95	17,06	24,96	31,95	37,86	41,76	43,84	45,61	47,05
		NRR	15,10	20,70	26,81	31,50	36,53	40,30	43,05	45,00	47,01
	Precision	kNN	10,93%	19,75%	27,31%	35,28%	41,95%	47,35%	52,08%	55,47%	58,23%
		IMSApriori	7,65%	20,20%	30,11%	39,37%	46,07%	51,11%	53,81%	56,28%	57,70%
		NRR	17,38%	24,76%	32,37%	38,71%	44,65%	49,46%	52,73%	54,94%	57,36%
	Recall	kNN	8,01%	13,35%	18,30%	23,15%	27,61%	31,09%	34,24%	36,49%	38,41%
		IMSApriori	6,37%	14,77%	21,32%	26,89%	32,13%	35,31%	36,99%	38,35%	39,73%
		NRR	13,36%	17,79%	22,88%	26,55%	30,90%	34,00%	36,37%	38,11%	39,83%

Fig. 2. Top-10 F1, precision and recall values for different density levels

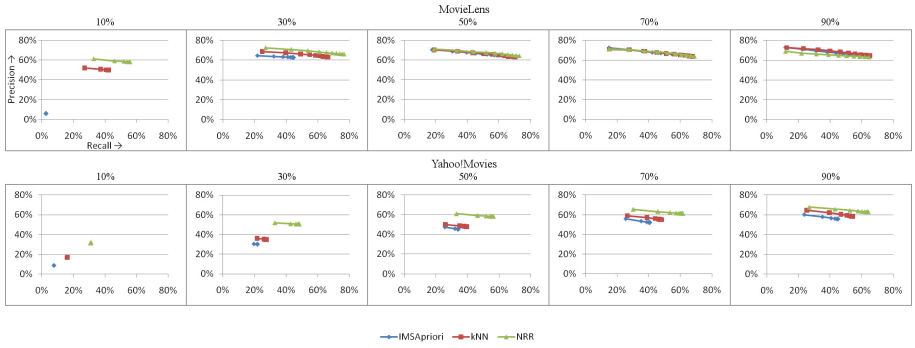


Fig. 3. Precision and recall values for varying list lengths and densities

which is based on the IMSApriori algorithm, consistently outperforms both the IMSApriori method and the kNN algorithm.

In a further experiment, we analyzed the accuracy of our method for different data sets, density levels and recommendation set sizes. Figure 3 shows the recall (x-axis) and precision (y-axis) values for different density levels. Each data point in a plot corresponds to the precision and recall values for given recommendation set size; the sizes were varied from 1 to 10 (i.e., top-1 to top-10 lists). The recall values naturally improve when the recommendation lists are longer.

In Figure 3, we can observe that the improvement of our NRR algorithm is stronger on the Yahoo! data set. A possible explanation for this observation could be the different sparsity levels of the two data sets, i.e., assuming that NRR works particularly well for sparse settings, it is intuitive that even better results can be achieved on the sparser Yahoo! data set (0.9976 sparsity) than on the MovieLens data set (0.9369 sparsity). As already observed in Figure 2, the performance improvements of NRR are higher for lower density levels.

4 Summary

Association rule mining is a powerful method that has been successfully used for various personalization and recommendation tasks in the past.

In this paper we have shown how the personalization of the learned model in rule mining-based approaches to recommendation can help to increase the accuracy of the system's prediction while at the same time the advantages of model-based approaches such as robustness against attacks and the possibility to generate explanations can be preserved. Data structures such as the Extended Frequent Itemset Graph can be used to efficiently generate recommendations online. Furthermore, given the explicit and comprehensible nature of the frequent itemsets, these (personalized) frequent itemsets can be easily manually extended with additional manually-engineered domain rules.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB 1994, Chile, pp. 487–499 (1994)
2. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: SIGKDD 1999, USA, pp. 337–341 (1999)
3. Sandvig, J.J., Mobasher, B., Burke, R.: Robustness of collaborative recommendation based on association rule mining. In: RecSys 2007, USA, pp. 105–112 (2007)
4. Lin, W., Alvarez, S., Ruiz, C.: Efficient adaptive-support association rule mining for recommender systems. Data Mining and Knowledge Discovery 6, 83–105 (2002)
5. Kiran, R.U., Reddy, P.K.: An improved multiple minimum support based approach to mine rare association rules. In: CIDM 2009, USA, pp. 340–347 (2009)
6. Mobasher, B., Dai, H., Luo, T., Nakagawa, M.: Effective personalization based on association rule discovery from web usage data. In: WIDM 2001, USA, pp. 9–15 (2001)