Neighborhood-restricted mining and weighted application of association rules for recommenders

Fatih Gedikli and Dietmar Jannach

Technische Universität Dortmund,
44227 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 a standard IMSApriori recommender and 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 [AS94]. A common example of an association rule that could be found in the sales transactions in a supermarket could be [LHM99]:

$$cheese \Rightarrow beer \quad [support = 10\%,\ confidence = 80\%]$$

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). Confidence and support are thus statistical measures that indicate the "strength" of the pattern or rule.

Quite obviously, the knowledge encoded in such automatically detected association rules (frequent itemsets) can be exploited to build recommender systems (RS)[1]. A simple recommendation algorithm capable of producing a "top-N" list of items could include the following steps: (1) Use Apriori to detect all association rules that surpass a minimum support threshold. These rules can be mined from real purchase data or from "like" statements in a rating database. (2) Take those rules that are "supported" by the target user (i.e., where the user has purchased all items on the rule's left-hand-side) and compute the set of items that are predicted by those rules and which the user has not yet purchased. (3) Sort the recommendation list by the confidence values of the involved rules. Early successful experiments using such a method for recommendation purposes with a slight variation of the prediction scheme are for example reported in [SKKR00].

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 [SMB07]. A further advantage of these approaches lies in the fact that the underlying model, i.e., the set of rules, is explicit and comprehensible to users. Thus, not only interesting consumer behavior phenomena can be learned from it, the rules can also be used to explain the recommendations to end users as to increase the users' confidence in the system's proposals. Finally, from a business perspective, the explicit rule bases can also be easily extended manually with additional domain knowledge, which is not easily possible with other learning-based recommendation methods.

While the accuracy of rule mining-based recommenders is comparable to nearest-neighbor (kNN) collaborative filtering approaches, using the original Apriori algorithm can lead to the problem of reduced coverage as shown in [SMB07]. 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. [LAR02] 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 [KR09] proposed a new method called IMSApriori that uses a particular metric to determine appropriate minimum support values per item (see also [LHM99]) 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 [NM03b] 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

[1] See [AT05] for an overview.

neighbors and not on the whole database of transactions, see also [Zan08]. Similar to kNN-approaches, the underlying idea of this is that close neighbors will be better predictors than others. The user's personalized knowledge base is then combined with the rule sets of his nearest neighbors to generate recommendation lists.

The paper is organized as follows. After an example and the introduction of the basics of the used rule mining methods, we describe our NRR method to learn personalized rule learning and prediction generation in detail. Afterwards, the results of an evaluation on typical CF data sets with different density levels and parameter settings are discussed. The paper ends with a conclusion and a short discussion of further works.

2 Example

Let us illustrate the different ideas in this paper with a simplified example. Consider the rating database in Figure 1 in which a "1" indicates that a user liked or purchased an item, a "0" corresponds to a dislike statement. Empty cells mean that no information is available. Let us assume that our goal is to make a recommendation for *User1*.

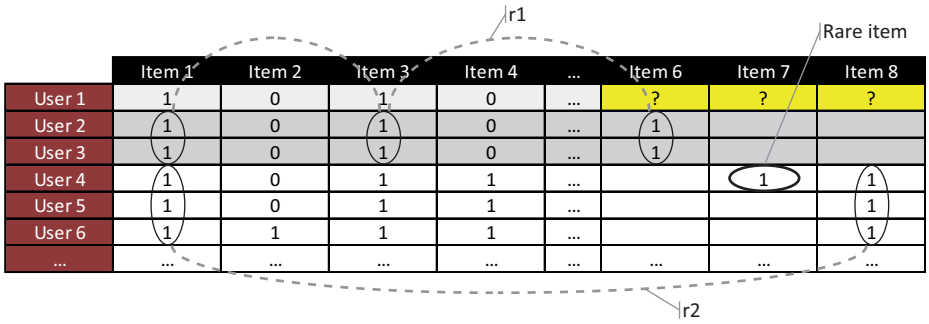


Fig. 1. Example setting for personalized rule bases.

Among others, a rule mining algorithm could detect rules such as

r1: $Item1, Item3 \Rightarrow Item6$ [$support = 33\%, confidence = 33\%$] and

r2: $Item1 \Rightarrow Item8$ [$support = 50\%, confidence = 50\%$]

where the second rule is "stronger" as it is supported by more evidence. Note that *Item7* is a "rare" item, i.e., only few ratings (in that simplified case only one) are available. When using the standard Apriori algorithm, a global minimum support threshold value is used in order to avoid the effect of "rule explosion". This however leads to the effect, that no rules can be learned for rare items because they never reach the global threshold value. Thus, as mentioned above, variations to the Apriori scheme like IMSApriori have been developed that employ multiple minimal support values to take the relative frequency of the items into account.

When making a prediction based on the standard scheme described in the introduction, both rule $r1$ and $r2$ apply as *User1* has rated both *Item1* and *Item3* positively. Since the confidence value of $r2$ is the higher one, the recommender would recommend *Item8* (place *Item8* before *Item6* in the recommendation list). The idea in our approach however is that closer neighbors are better predictors than other users. In the example, it could therefore be a good idea to recommend *Item6* because rule $r1$ is supported by the ratings of *User2* and *User3* who are very similar to the target user *User1* in their rating behavior (in that case even identical).

The NRR method proposed in this paper works as follows. First, we learn a personalized set of rules for each user by taking only the n closest neighbors into account. In an extreme setting, we could only use the ratings of *User2* and *User3* for learning the rules for *User1*. In that case, rule *r1* would be learned. The rule base for *User4*, on the other hand, would probably also include $r2$. When again applying the standard prediction scheme, *Item6* would now be recommended for *User1* as intended. However, the coverage of that approach could be very limited. Thus, we propose a neighborhood-based prediction scheme, in which also the rules of the neighbors are taken into account. When recommending items for *User1* we would therefore use the rules of *User2* and *User3*, but probably also those of *User4* (which include the rule $r2$). In that case, coverage is increased again. At the same time - if we give limited weights to rules of farther-away neighbors - $r1$ can remain the dominating rule and cause *Item6* to be at the top of the recommendation list without losing the other rules.

3 Algorithms

In the following we will shortly summarize the rough ideas of the used rule mining approaches Apriori and IMSApriori 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 [NM03b], has to be extended for a recommender based on IMSApriori.

Apriori. The original Apriori algorithm [AS94] works by iteratively generating a set of *candidate itemsets* in multiple phases. In our example above, it will start by constructing one-element itemsets, such as $\{Item1\}$ and $\{Item2\}$, and then check if these itemsets have minimum support, where the support of an itemset $X \Rightarrow Y$ is defined as the ratio of the number of transactions containing $X \cup Y$ to the number of all transactions. Itemsets that have not enough support are pruned. In the next phase, the remaining itemsets are combined with one more (frequent) element and checked against the database. This process is repeated until no more candidates can be generated. Overall, one of the ideas of the algorithm's implementation is that "any subset of a large itemset must be large"[2] [AS94].

[2] Frequent itemsets were originally called *large* itemsets.

Once all frequent itemsets are detected, association rule mining approaches use further quality metrics to measure the significance of the detected rules. The *confidence* of a rule $X \Rightarrow Y$ is defined as $\frac{support(X \cup Y)}{support(X)}$, which is a common metric to prune uninteresting rules. The threshold values for minimum *confidence* and *support* are often empirically determined and have to be specified by the user.

IMSApriori. In order to deal with the problem of "missing rules" for rare, but interesting itemsets, different proposals have been made. IMSApriori [KR09], 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 [LHM99]. 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 [LHM99] 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 [KR09], 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, 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.

Neighborhood-restricted Rule-based Recommender (NRR). As shown in the example, 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 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.

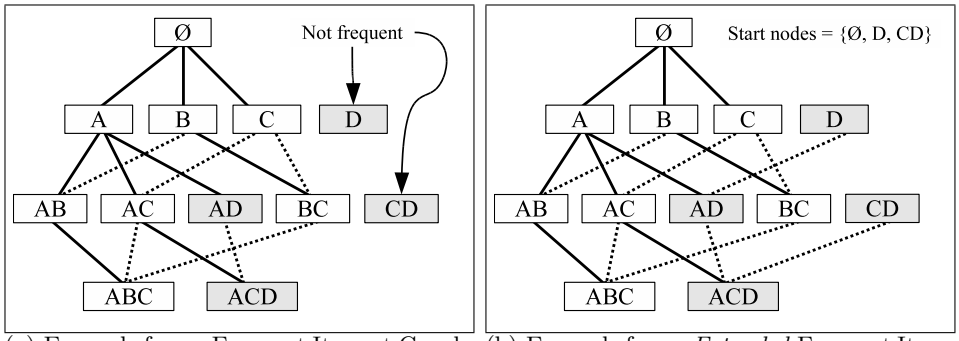
Algorithm 1 NRR algorithm (sketch).

```
In: user, ratingDB, learnNeighborSize, predictNeighborSize, λ, α
Out: recommendedItems
(Offline:) UserFISs = CalcUserFISsIMSApriori(ratingDB, learnNeighborSize, λ, α)
neighborhood = user ∪ findNeighbors(user, predictNeighborSize, ratingDB)
recommendedItems = ∅
for all u ∈ neighborhood do
   userRecs = Recommend(u,buildEFIG(UserFISs(u)))
   weightedUserRecs = adjustConfidenceScoresBySimilarity(userRecs, user, u)
   recommendedItems = recommendedItems ∪ weightedUserRecs
end for
recommendedItems = sortItemsByAdjustedScores(recommendedItems)
```

The Extended Frequent Itemset Graph. The *Frequent Itemset Graph* (FIG) as proposed in [MDLN01] is a data structure to organize the frequent itemsets in a way that allows us to generate recommendations directly from the frequent itemsets (i.e., without the need to derive all association rules first). Figure 3(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. 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 supersets of $\{A, D\}$ in the next level of the graph. In the example, given the superset $\{A, D\}$, C could be recommended to u. The solid arrows in the figure indicate how the graph would be traversed in depth-first order.

Since, however, 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 3(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.

In our work, we 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 3(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. Note



(a) Example for an Frequent Itemset Graph. (b) Example for an *Extended* Frequent Itemset Graph.

Fig. 2. Extended Frequent Itemset Graph approach.

that the negative effects on the scalability of the approach itself are very limited, because only small portions of the graph have to be analyzed in these additional traversals.

4 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 standard correlation-based kNN-method. In the following, we will summarize the findings of this evaluation.

4.1 Experimental Setup and Evaluation Metrics

Data sets. As data sets for the evaluation, we used the 100k-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[3]. Regarding the user characteristics, the MovieLens data set only contains users who have rated at least 20 items; the minimum number of rated items per user in the Yahoo! data set is 10. In addition, in the Yahoo! data set, each item was at least rated by one user.

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 as described in [SKKR01]. The smallest subsample contained 10% of the original data. In this subsample, the average number of ratings per user was around 10 as opposed to 100 for the original MovieLens data set. Further measurements were taken in steps of 10% up to the 90% data

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

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 follow the evaluation procedure proposed in [NM03a] and 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 of length $|ELS|$ with each method based on the data in the training set[4]. In the kNN-case, the rating predictions are converted into "like" statements as described in [SMB07], 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)*, where $|PLS| \leq |ELS|$.

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*[5] 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, recommendations and the corresponding precision and recall values were calculated for all users in the data set and then averaged. These 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. Regarding the algorithms, note that 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 [SKKR01].

The IMSApriori implementation used in the experiments corresponds to above-described algorithm and learns the rules from the whole database of 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 were analyzed by conducting multiple experiments on the MovieLens data set with a fixed density level of 70%. The value of the parameter *neighborhood-size-predict* was empirically determined to be 100, see Figure 3 (a). To analyze the sensitivity of this parameter, we performed experiments in which we varied the number of neighbors used for making predictions and fixed the parameter *neighborhood-size-learn* at 30 as suggested as an optimal value for this data set in literature.

[4] 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.

[5] In [NM03a], this metric is called *coverage*.

We can observe from the figure that the recall value increases from 46% to 61% when moving from a prediction neighborhood size of 10 to 100. At the same time, the precision value stays rather constant at 65% and does not decrease. Afterwards, we fixed the parameter *neighborhood-size-predict* at 100 and analyzed the sensitivity of *neighborhood-size-learn*, see Figure 3 (b). It can be seen that the initial value of 30 was actually a good choice for this parameter.

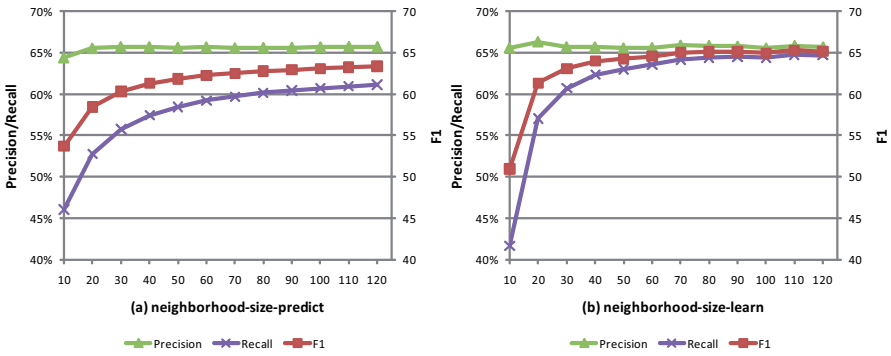


Fig. 3. Sensitivity of the parameters *neighborhood-size-predict* (a) and *-learn* (b).

Finding a suitable lower support threshold value (LS) for the rule learning methods is challenging. In [MDLN01], the authors argue that higher LS values are in general more desirable because they lead to a smaller model size, as the number of frequent itemsets decreases, which in turn leads to good scalability. Higher LS values however may lead to the effect that no rules for rare items can be found.

In addition, remember that in IMSApriori-based algorithms the minimum item support MIS for each item has to be at least LS, i.e., $MIS(i) >= LS$, for each item i. Therefore, when using a high global LS value, the problem can arise that each item gets the same MIS value, i.e., the value of the global minimum support threshold LS. As a consequence, the IMSApriori algorithm would behave like the standard Apriori algorithm which uses one global lower support threshold value. Note that this phenomenon can also appear when the density level of the data set is very low. A low density level implies low item support values, which can result in MIS values that are below the LS value.

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: 19%), which we have then used for all density levels and data sets[6].

Regarding computational complexity, the proposed NRR algorithm runs the IMSApriori algorithm once for each user, which can be done offline during the model building phase. Each IMSApriori execution however only has to consider a

[6] Our current work includes a more detailed analysis of optimal parameter values for specific density levels.

relatively small fraction of the whole database of transactions, or, more precisely, at most *neighborhood-size-learn* transactions, when learning rules. On a standard desktop computer (Intel Core 2 Duo CPU, 2.4 GHz, 3GB RAM), learning the rules for all users takes about 11 minutes for the MovieLens data set at a density level of 70%. The size of the resulting model is determined by the number of existing frequent itemsets. In the baseline IMSApriori recommender, 316 frequent itemsets were found. When using the NRR algorithm, the average model size is about 45 frequent itemsets for each user (MovieLens data set, density-level of 70%). The model size of course strongly depends on the selected LS values.

Results. Figure 4 summarizes the evaluation results for the three algorithms kNN, IMSApriori and NRR. The table shows the average values of the F-score as well as the precision and recall values for the different density levels for both the MovieLens and the Yahoo! data set.

		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. 4. Overall average F1, precision and recall values for different density levels.

The results show that our NRR algorithm consistently outperforms the kNN algorithm on the F1-measure and is better than the IMSApriori method 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 in settings with medium and higher density levels, also the pure IMSApriori version outperforms the kNN-method, which was not analyzed in previous research. Note that the accuracy gains are stronger for

the MovieLens data set, which can be partially attributed to the fact that the optimal parameters were empirically determined based on this data set.

A closer look at the precision and recall values shows that NRR has particular advantages with respect to the recall measure, i.e., NRR is capable of retrieving more relevant items than the other algorithms while at the same time precision values remain at a comparably high level. Again, this effect is particularly strong when the data sets are very sparse, which is a common situation in most real-world settings.

5 Summary

Association rule mining is a powerful method that has been successfully used for various personalization and recommendation tasks in the past; see for example its recent application for social tag prediction ([HRGM08], [WHD09]).

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.

Additional computational costs for the personalization task arise only in the offline phase in which multiple smaller frequent itemset collections are computed instead of one large one. At run-time, 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.

Our future work includes the evaluation of our approach on further data sets and a comparison with further algorithms. In addition, in our current work, we conduct experiments in which we first perform probabilistic clustering on the user base and then mine the frequent itemsets for each cluster. While we might not expect significant accuracy gains, this approach will lead to a substantially reduced model size which further improves the scalability of our approach. In addition, the neighborhood-size parameter will not be required in the training phase when a method like *AutoClass* [CKS+93] is used to determine the optimal number of clusters automatically.

References

[AS94] Rakesh Agrawal and Ramakrishnan Srikant, *Fast algorithms for mining association rules in large databases*, Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94) (Santiago de Chile, Chile), 1994, pp. 487–499.

[AT05] Gediminas Adomavicius and Alexander Tuzhilin, *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions*, IEEE Transactions on Knowledge and Data Engineering **17** (2005), no. 6, 734–749.

[CKS+93] Peter Cheeseman, James Kelly, Matthew Self, John Stutz, Will Taylor, and Don Freeman, *Autoclass: A bayesian classification system*, Readings in knowledge acquisition and learning: automating the construction and improvement of expert systems, Morgan Kaufmann, 1993, pp. 431–441.

[HRGM08] Paul Heymann, Daniel Ramage, and Hector Garcia-Molina, *Social tag prediction*, Proceedings 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'08) (Singapore, Singapore), 2008, pp. 531–538.

[KR09] R. Uday Kiran and P. Krishna Reddy, *An improved multiple minimum support based approach to mine rare association rules*, Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2009 (Nashville, TN, USA), 2009, pp. 340–347.

[LAR02] W. Lin, S. Alvarez, and C. Ruiz, *Efficient adaptive-support association rule mining for recommender systems*, Data Mining and Knowledge Discovery **6** (2002), 83–105.

[LHM99] Bing Liu, Wynne Hsu, and Yiming Ma, *Mining association rules with multiple minimum supports*, Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99) (San Diego, CA, United States), 1999, pp. 337–341.

[MDLN01] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa, *Effective personalization based on association rule discovery from web usage data*, Proceedings of the 3rd International Workshop on Web Information and Data Management (WIDM'01) (Atlanta, Georgia, USA), 2001, pp. 9–15.

[NM03a] Miki Nakagawa and Bamshad Mobasher, *A hybrid web personalization model based on site connectivity*, Proceedings of the 2003 WebKDD Workshop (Washington, DC, USA), 2003, pp. 59–70.

[NM03b] ———, *Impact of site characteristics on recommendation models based on association rules and sequential patterns*, Proceedings of the IJCAI'03 Workshop on Intelligent Techniques for Web Personalization (Acapulco, Mexico), 2003.

[SKKR00] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, *Analysis of recommendation algorithms for e-commerce*, Proceedings of the 2nd ACM Conference on Electronic Commerce (EC'00) (Minneapolis, MN, USA), 2000, pp. 158–167.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl, *Item-based collaborative filtering recommendation algorithms*, Proceedings of the 10th International Conference on World Wide Web (WWW'01) (Hong Kong), 2001, pp. 285–295.

[SMB07] J. J. Sandvig, Bamshad Mobasher, and Robin Burke, *Robustness of collaborative recommendation based on association rule mining*, Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys'07) (Minneapolis, MN, USA), 2007, pp. 105–112.

[WHD09] Jian Wang, Liangjie Hong, and Brian D. Davison, *Tag recommendation using keywords and association rules (RSDC'09)*, ECML PKDD Discovery Challenge 2009 (DC09) (Bled, Slovenia), vol. 497, CEUR Workshop Proceedings, 2009, pp. 261–274.

[Zan08] Markus Zanker, *A collaborative constraint-based meta-level recommender*, Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys'08) (Lausanne, Switzerland), 2008, pp. 139–146.