Towards Addressing the Coverage Problem in Association Rule-based Recommender Systems

R. Uday kiran[†] and Masaru Kitsuregawa[†‡]

[†] Institute of Industrial Science, The University of Tokyo, Tokyo, Japan.
[‡] National Institute of Informatics, Tokyo, Japan.
uday_rage@tkl.iis.u-tokyo.ac.jp and kitsure@tkl.iis.u-tokyo.ac.jp.

Abstract. Association rule mining is an actively studied topic in recommender systems. A major limitation of an association rule-based recommender system is the problem of reduced coverage. It is generally caused due to the usage of a single global minimum support (*minsup*) threshold in the mining process, which leads to the effect that no association rules involving rare items can be found. In the literature, Neighborhood-Restricted Rule-Based Recommender System (NRRS) using multiple *minsups* framework has been proposed to confront the problem. We have observed that NRRS is computationally expensive to use as it relies on an Apriori-like algorithm for generating frequent itemsets. Moreover, it has also been observed that NRRS can recommend uninteresting products to the users. This paper makes an effort to reduce the computational cost and improve the overall performance of NRRS. A pattern-growth algorithm, called MSFP-growth, has been proposed to reduce the computational cost of NRRS. We call the proposed framework as NRRS*. Experimental results show that NRRS* is efficient.

Keywords: Recommender system, association rules and coverage problem.

1 Introduction

Association rules [1] are an important class of regularities that exist in a database. Recently, these (association) rules were used in developing recommender systems [2, 3]. It is because their performance is comparable to nearest-neighbor (kNN) collaborative filtering approaches, and do not suffer from scalability problems like memory-based algorithms as the rules can be mined in an offline model-learning phase [4].

A major limitation of a rule-based recommender system is the problem of **reduced coverage** [4, 5]. This phenomenon is generally caused due to the usage of a single *minsup* threshold in the mining process, which leads to the effect that no rules involving rare items can be found. Fatih and Dietmar [5] have exploited the concept of rule mining with multiple *minsups* [8], and introduced NRRS (Neighborhood-Restricted rule-based Recommender System) to address the coverage problem. The NRRS uses an Apriori-like algorithm known as Improved Multiple Support Apriori (IMSApriori) [8] to generate frequent itemsets from each user's dataset. Next, it uses a tree-structure known as Extended Frequent Itemset-graph (EFI-graph) to store the rules generated from the frequent itemsets. Recommendations to the user are generated by performing depth-first search on EFI-graph. We have observed the following performance issues in NRRS:

– The NRRS is a computationally expensive recommender system. It is because of the following two reasons. First, the rules discovered with the multiple *minsups* do not satisfy the anti-monotonic property. This increases the search space, which in turn increases the computational cost of mining the frequent itemsets. Second, the IMSApriori suffers from the same performance problems as the Apriori [1], which involves generating huge number of candidate itemsets and requiring multiple database scans.
– In [10], it has been reported that EFI-graph causes NRRS to recommend uninteresting items to the users.

With this motivation, we propose an improved NRRS known as NRRS*. The key contributions of this paper are as follows:

1. We show that the frequent itemsets discovered with the multiple *minsups* framework satisfy the "convertible anti-monotonic property" [11].
2. Pei et al. [11] have theoretically shown that the search space of convertible anti-monotonic property is same as the search space of an anti-monotonic property if the frequent itemsets were discovered by a pattern-growth algorithm. Therefore, we propose a pattern-growth algorithm known as Multiple Support Frequent Pattern-growth (MSFP-growth). The MSFP-growth do not suffer from the same performance problems as the IMSApriori. As a result, it can effectively discover frequent itemsets from each user's dataset.
3. In [10], we have proposed EFI-graph++ to store the rules that satisfy the convertible anti-monotonic property. Since the rules discovered with the multiple *minsups* framework satisfy the same, we employ EFI-graph++ to store the rules and recommend products to the users [10].
4. Performance results on MovieLens dataset demonstrate that NRRS* can provide better recommendations and is more scalable than the NRRS.

The rest of the paper is organized as follows. Section 2 discusses the background and related work on association rule mining and rule-based recommender systems. Section 3 describes MSFP-growth and NRRS* algorithms. The experimental evaluations of NRRS and NRRS* have been reported in Section 4. Finally, Section 5 concludes the paper.

2 Background and Related Work

2.1 Association Rule Mining

Discovering association rules with multiple *minsups* is an important generalization proposed by Liu et al. [6] to confront the "rare item problem" that occurs in the basic model of association rules [1]. The multiple *minsups* model of association rules is as follows [6]:

Let $I = \{i_1,\ i_2,\ \cdots,\ i_n\}$ be a set of items. Let T be a set of transactions (or a dataset), where each transaction t is a set of items such that $t \subseteq I$. A set of items $X \subseteq I$ is called an itemset (or a pattern). An itemset containing k number of items is called a k-itemset. An association rule is an implication of the form, $A \Rightarrow B$, where $A \subset I$, $B \subset I$ and $A \cap B = \emptyset$. The rule $A \Rightarrow B$ holds in T with *support s*, if s% of the transactions in T contain $A \cup B$. The rule $A \Rightarrow B$ holds in T with *confidence c*, if c% of transactions in T that support A also support B. An itemset (or a rule) is said to be **frequent** if its support is no less than the *minsup* threshold. The rules which satisfy both *minsup* and $minconf$ thresholds are called **strong rules**. The $minconf$ is the user-defined minimum confidence threshold value, while the *minsup* of a rule is expressed as follows: $minsup(A \Rightarrow B) = minimum(MIS(i_j)|\forall i_j \in A \cup B)$, where $MIS(i_j)$ is the user-defined *minimum item support* for an item $i_j \in A \cup B$.

An open problem with the multiple *minsups* framework is the methodology to specify the items' MIS values. Uday and Reddy [8] have tried to addressed this problem using the following methodology:

$$\begin{aligned} MIS(i_j) &= maximum(LS,\ S(i_j) - SD) \\ SD &= \lambda(1-\beta) \end{aligned} \tag{1}$$

where, LS is the user-specified lowest minimum item support allowed, SD refers to *support difference*, λ represents the parameter like *mean*, *median* and *mode* of the item supports and $\beta \in [0,1]$ is a user-specified constant. An Apriori-like algorithm, called IMSApriori, has been proposed to discover frequent itemsets. Researchers have also introduced algorithms based on pattern-growth technique [7, 9] to discover the patterns. Unfortunately, these algorithms cannot be directly applied in NRRS to discover the frequent itemsets. The reason is that **the algorithms require user-specified items' MIS values**. Thus, we have proposed an alternative pattern-growth algorithm, called MSFP-growth, which can easily be incorporated in a rule-based recommender system.

2.2 Rule-Based Recommender Systems

Recently, rule-based recommender systems have received a great deal of attention in both industry and academia [2, 3]. Most of these systems employ only a single *minsup* constraint to discover frequent itemsets from each user's dataset. As a result, they suffer from **coverage problem**. To confront the problem, researchers have introduced NRRS using the concept of multiple *minsup*-based frequent itemset mining [5]. The working of NRRS is as follows:

1. (**Offline phase.**) The users were grouped based on their purchased history. A transactional database is constructed using the purchased history of each user's group. Next, the complete set of frequent itemsets were discovered from each database using IMSAprori.

2. (**Online phase.**) For each user, the discovered frequent itemsets were stored in a lexical order in a tree-structure known as EFI-graph. Recommendations to the user were generated by traversing EFI-graph using depth-first search.

Since NRRS employs IMSApriori to discover frequent itemsets from each user's database, it is straight forward to argue that NRRS is a computationally expensive recommender system. Moreover, it was shown in [10] that EFI-graph can generate uninteresting recommendations to the users. In the next section, we propose NRRS*, which tries to address the performance issues of NRRS.

3 The Proposed Neighborhood-Restricted Rule-Based Recommender System*

3.1 MSFP-growth

To reduce the computational cost of mining frequent itemsets from each user's dataset, we have investigated the types of itemsets discovered with the multiple *minsups* framework and the nature of support-difference methodology. We made the following key observations:

1. The support-difference methodology is a monotonic (or an order-preserving) function. That is, if $S(i_1) \geq S(i_2) \geq \cdots \geq S(i_n)$, then $MIS(i_1) \geq MIS(i_2) \geq \cdots \geq MIS(i_n)$.
2. For an item $i_j \in I$, the support-difference methodology specifies items' MIS values such that $S(i_j) \geq MIS(i_j) \geq LS$.
3. The frequent itemsets discovered with the multiple *minsups* framework satisfy the convertible anti-monotonic property. The correctness is based on Property 1 and shown in Lemma 1.

Property 1. (Apriori Property.) If $Y \subset X$, then $S(Y) \geq S(X)$.

Lemma 1. *(Convertible anti-monotonic property.) In a frequent itemset, all its non-empty subsets containing the item with lowest MIS must also be frequent.*

Proof. Let $X = \{i_1, i_2, \cdots, i_k\}$, where $1 \leq k \leq n$, be an ordered set of items such that $MIS(i_1) \leq MIS(i_2) \leq \cdots \leq MIS(i_k)$. Let Y be an another itemset such that $Y \subset X$ and $i_1 \in Y$. The *minsup* of X, denoted as $minsup(X) = MIS(i_1) \ \ (= min(MIS(i_j)|\forall i_j \in X))$. Similarly, the *minsup* of Y, denoted as $minsup(Y) = MIS(i_1) \ \ (= min(MIS(i_j)|\forall i_j \in Y))$. If $S(X) \geq MIS(i_1)$ $(= minsup(X))$, then it is a frequent itemset. Since $Y \subset X$, it turns out that $S(Y) \geq MIS(i_1))$ as $S(Y) \geq S(X) \geq MIS(i_1)$ (Property 1). In other words, Y is also a frequent itemset. Hence proved.

Based on the above observations, we introduce a pattern-growth algorithm known as MSFP-growth (see Algorithm 1). Briefly, the working of MSFP-growth is as follows:

– We construct FP-tree [11] and measure the SD value.

– Since FP-tree is constructed in support descending order of items, it turns out that the MIS value of the suffix item will be the lowest MIS value among all other items in its prefix path. Therefore, considering each item in the FP-tree as suffix item (or 1-itemset), we calculate its MIS value and consider it as "conditional minimum support" ($Cminsup$). Next, we construct its conditional pattern base, then construct its (conditional) FP-tree, and perform mining recursively on such a *tree*. The pattern-growth is achieved by the concatenation of the suffix itemset with the frequent itemsets generated from a conditional FP-tree.

Algorithm 1 MSFP-growth (T: rating transactional database, β: user-specified constant and LS: least support)

```
1: Scan the transactional database $T$ and measure the support values of all items.
2: Prune the items having support values less than the $LS$ value and sort them in
   descending order of their support values. Let this sorted order of items be $L$. Every
   item in $F$ is a frequent item.
3: Let $\lambda$ be the support of an representative item in the database. It generally repre-
   sents mean or median of all supports of items in $L$. That is, $\lambda = mean(S(i_j)\forall i_j \in L)$
   or $\lambda = median(S(i_j)\forall i_j \in L)$.
4: Calculate the support difference ($SD$) between the support and $MIS$ of the rep-
   resentative item. That is, $SD = \lambda(1 - \beta)$.
5: Since support-difference methodology is an order preserving function, construct
   FP-tree, say $Tree$, as discussed in [11].
6: for each item $i$ in the header of the $Tree$ do
7:   calculate conditional $minsup$, $Cminsup_i = ((S(i) - SD) > LS)?(S(i) - SD)$ :
     $LS)$.
8:   if $i_j$ is a frequent item then
9:     generate itemset $\beta = i \cup \alpha$ with support = $i$.support (or $S(i)$);
10:    construct $\beta$'s conditional pattern base and $\beta$'s conditional FP-tree $Tree_\beta$.
11:    if $Tree_\beta \neq \emptyset$ then
12:      call MSFPGrowth($Tree_\beta$, $\beta$, $Cminsup_i$).
13:    end if
14:  end if
15: end for
```

Procedure 2 MSFPGrowth(Tree, α, $Cminsup_i$)

```
1: for each $i$ in the header of Tree do
2:   generate itemset $\beta = i \cup \alpha$ with support = $i$.support.
3:   construct $\beta$'s conditional pattern base and then $\beta$'s conditional FP-tree $Tree_\beta$.
4:   if $Tree_\beta \neq \emptyset$ then
5:     call MSFPGrowth($Tree_\beta$, $\beta$, $Cminsup_i$).
6:   end if
7: end for
```

3.2 An Improved Neighborhood-Restricted Rule-based Recommender System: NRRS*

The proposed NRRS* system is shown in Algorithm 3. It is a two phase algorithm involving offline and online phases. The working of NRRS* is as follows:

1. (**Offline phase.**) For each user u, top-k_1 and top-k_2, where $k_1 \geq k_2$, number of neighboring users who had similar purchased history are discovered. (The top-k_1 neighboring users will be used for learning rules, while the top-k_2 neighboring users will be used for prediction or recommendation of products to the user u.) A transactional database is generated using the purchased history of top-k_1 neighbors of the user u. Next, frequent itemsets are discovered from the database using the proposed MSFP-growth algorithm.
2. (**Online phase.**) The discovered user-specific frequent itemsets (UserFPs) of the target user and of the neighbors of the target user are used to calculate predictions using EFI-graph++ (lines 8 to 13 in Algorithm 3). The resulting confidence scores are weighted according to the similarity of the target user and the neighbor (line 14 to 16 in Algorithm 3). The similarity measure we use is Pearson correlation. These user-specific predictions are finally combined and sorted by the weighted confidence scores (line 16 and 17 in Algorithm 3). The construction of EFI-graph++ is given in [10].

Algorithm 3 NRRS* (U: set of users, $ratingDB$: rating database, k_1: learning neighborhood size, k_2: predicting neighborhood size λ: support of a representative item, β: user-specified constant and LS: least minimum item support)

```
 1: (Offline: Discovery of Frequent itemsets)
 2: for each $u \in U$ do
 3:    Let $k_1$-neighborhood$^u$ and $k_2$-neighborhood$^u$ be the set of top $k_1$ and $k_2$
       neighbors for the user $u$ in $ratingDB$. That is, $k_1$-neighborhood$^u$ $= u \cup$
       $findNeighbors(u, k_1, ratingDB)$;
 4:    Let $ratingDB^u$ be the rating database created using the ratings given by the
       each user in $k_1$-neighborhood$^u$.
 5:    Let $UserFPs$ be the of frequent itemsets discovered for $u$ in $ratingDB^u$ using
       MSFP-growth. That is, $UserFPs = MSFP\text{-}growth(ratingDB^u, \lambda, \beta, LS)$.
 6: end for
 7: (Online: Recommending items by constructing EFI-graph for each user)
 8: for each user $u \in U$ do
 9:    $recommendedItems = \emptyset$;
10:    for each user $u_i \in k_1$-neighborhood$^u$ do
11:       $userRecs = Recommend(u, EFIG++(UserFPs(u_i)))$;
12:       Scan $ratingDB^u$ to measure the support count of infrequent itemsets in EFI-
          graph of $u$;
13:       weightedUserRecs = adjustConfidenceScoresBySimilarity($userRecs, u, u_i$);
14:       $recommendedItems = recommendedItems \cup weightedUserRecs$;
15:    end for
16:    $recommendedItems = sortItemsByAdjustedScores(recommendedItems)$;
17:    output $recommendedItems$;
18: end for
```

4 Experimental Results

In this section, we evaluate the NRRS and proposed NRRS*. The programs are written in java and run with Ubuntu on a 2.66 GHz machine with 2GB memory. The experiments were pursued on "MovieLens" rating dataset consisting of 100,000 ratings provided by 943 users on 1,682 items.

In order to test NRRS and NRRS* frameworks also in settings with low data density, we varied the density level of the original data sets by using sub-samples of different sizes of the original data set. Four-fold cross-validation was performed for each data set; in each round, the data set was split into a 80% training set and a 20% test set.

To compare the predictive accuracy of NRRS and NRRS* systems, we use the following procedure. First, we determine the set of existing "like" statements (ELS) in the 20% test set and retrieve a top-N recommendation list with each system based on the data in the training set. The top-N recommendations are generated based on the confidence of the producing rule. The set of predicted like statements returned by a recommender shall be denoted as Predicted Like Statements (PLS). The standard information retrieval accuracy metrics are used in the 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. The averaged precision and recall values are then combined in the usual *F-score*. That is, $F = 2 \times \left(\frac{precision \times recall}{precision + recall}\right)$.

Table 1 shows the performance of NRRS and NRRS* systems on MovieLens dataset with varying densities. It can be observed that the performance of NRRS* is relatively better than NRRS, independent of the density. The reason is that EFI-graph++ facilitated NRRS* to recommended only those items that had high confidence value although the past transactions of a user represented an infrequent itemset.

Table 1. Performance comparison of NRRS and NRRS* at different density levels.

Density		10%	30%	50%	70%	90%
Movie Lens	F1	35.75	47.75	57.89	60.91	62.72
	Precision	45.9%	60.23%	63.61%	64.7%	63.8%
	Recall	29.3%	39.56%	53.12%	57.54%	61.69%

Fig. 1 shows the runtime taken by NRRS and NRRS* systems to generate recommendations to the users with varying dataset sizes for learning the rules (i.e., k_1). It can be observed that although both algorithms are linearly scalable with increase in dataset size, NRRS* is relatively more efficient than NRRS system. It is because the MSFP-growth discovered frequent itemsets for each users' dataset more effectively than IMSApriori. Another important observation is that NRRS* is more scalable than NRRS with increase in k_1 neighborhood (or dataset) size for learning the rules.

5 Conclusions

This paper has proposed an effective and efficient rule-based recommender system, called NRRS*, to confront the coverage problem that persists in the conven-

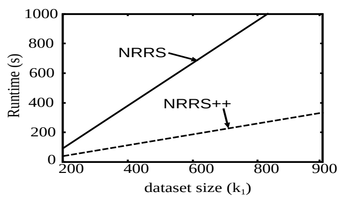

Fig. 1. Runtime comparison of mining frequent itemsets for each user in NRRS and NRRS⋆. The k_1 represents a users' dataset size (or neighborhood size).

tional rule-based recommender systems. It has also shown that frequent itemsets discovered with the multiple *minsups* framework satisfy the convertible anti-monotonic property, and therefore, can be effectively discovered using the proposed pattern-growth algorithm, called MSFP-growth. Experimental results demonstrate that NRRS⋆ is efficient than NRRS.

References

1. Rakesh Agrawal, Tomasz Imieliński and Arun Swami: Mining association rules between sets of items in large databases. In: SIGMOD, pp. 207–216 (1993).
2. Gediminas Adomavicius and Alexander Tuzhilin: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. In: IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 6, pp. 734–749 (2005).
3. Barry Smyth, Kevin McCarthy, James Relly, Derry O'Sullivan, Lorraine McGinty, D. C. Wilson: Case Studies in Association Rule Mining for Recommender System. In: IC-AI, pp. 809–815 (2005).
4. Weiyang Lin, Sergio A. Alvarez, Carolina Ruiz: Efficient Adaptive-Support Association Rule Mining for Recommender Systems. In: Data Mining and Knowledge Discovery. Vol. 6, No. 1, 83–105 (2002).
5. Fatih Gedikli, Dietmar Jannach: Neighborhood-Restricted Mining and Weighted Application of Association Rules for Recommenders. In: WISE, pp. 157–165 (2010).
6. Bing Liu, Wynne Hsu, Yiming Ma: Mining association rules with multiple minimum supports. In: KDD, pp. 337–341 (1999).
7. Ya-Han Hu, Yen-Liang Chen: Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism. Decision Support Systems, Volume 42, Issue 1, pp. 1–24 (2006).
8. R. Uday Kiran, P. Krishna Reddy: An Improved Multiple Minimum Support Based Approach to Mine Rare Association Rules. In: Proceedings of CIDM, pp. 340–347 (2009).
9. R. Uday Kiran, P. Krishna Reddy: Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. In: Proceedings of EDBT, pp. 11–20 (2011).
10. R. Uday Kiran, P. Krishna Reddy: An Improved Neighborhood-Restricted Association Rule-based Recommender System Using Relative Support. To be Appeared in ADC (2013).
11. Jian Pei and Jiawei Han: Constrained frequent pattern mining: a pattern-growth view. SIGKDD Explor. Newsl. Vol. 4, No. 1, 31–39 (2002).