An Improved Neighborhood-Restricted Association Rule-based Recommender System

R. Uday Kiran **Masaru Kitsuregawa**

Institute of Industrial Science,
The University of Tokyo,
Meguro-ku, Tokyo, Japan,
Email: {uday_rage, 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. To confront the problem, researchers have introduced Neighborhood-Restricted rule-based Recommender System (NRRS) using the concept of multiple *minsups*. We have observed that NRRS is computationally expensive to use and can recommend uninteresting products to the users. With this motivation, this paper proposes an improved NRRS using the relative support measure. We call the proposed system as NRRS++. Experimental results show that NRRS++ can provide better recommendations and is runtime efficient than NRRS.

Keywords: Recommender Systems, Association Rules and Coverage Problem.

1 Introduction

Association rule mining is an important knowledge discovery technique in data mining. It detects the association between items that are interesting to the user in a database. The classic application is market basket analysis, where (association) rule mining analyzes how the items purchased by customers are associated. An example of a rule is as follows,

$$bed \Rightarrow pillow \quad [support = 10\%,\ confidence = 75\%]$$

This rule says that 10% of customers have bought *bed* and *pillow* together, and those who bought *bed* have also bought *pillow* 75% of the time. The *support* and $confidence$ are the statistical measures used to access the "strength" of a rule. In particular, the $confidence$ of a rule measures the degree of correlation among the items, while the *support* of a rule measures the significance of the correlation among items. These terms are defined in the later parts of the paper.

Recently, rules were being used in developing recommender systems (Adomavicius et al. 2001, Smyth et al. 2005, Abel et al. 2008). 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 (Lin et al. 2002).

The general working of a rule-based recommender system is as follows:

1. (**Offline.**) Users (or items) are grouped based on their purchased history. A dataset is generated using the purchased history of each group. Frequent itemsets are discovered from each dataset. An itemset (or a set of items) is said to be frequent if it satisfies the user-defined minimum support (*minsup*) threshold. The *minsup* threshold controls the minimum number of transactions in which an itemset (or a rule) must occur in the database.
2. (**Online.**) The discovered frequent itemsets are stored in a tree-structure which preserves the confidence value of a rule generated from a frequent itemset. Recommendations are generated by performing depth-first search on the constructed tree-structure using the purchased history of the user.

The frequent itemset mining is a key step in a rule-based recommender system, because, it affects the computational cost and predictive accuracy of a recommender system.

A major limitation of a rule-based recommender system is the problem of reduced coverage (Lin et al. 2002, Gedikli & Jannach 2010). This phenomenon is generally caused due to the usage of a single global *minsup* threshold in the mining process, which leads to the effect that no rules involving rare items can be found.

To confront the problem, researchers have introduced Neighborhood-Restricted rule-based Recommender System (NRRS) (Gedikli & Jannach 2010) using multiple *minsups* framework (Liu 1999). The NRRS uses an Apriori-like algorithm known as Improved Multiple Support Apriori (IMSApriori) (Kiran & Reddy 2009) to discover frequent itemsets and a tree-structure known as Extended Frequent Itemset-graph (EFI-graph) to store the rules and recommend products to the users. The following performance issues were observed in NRRS:

- In the multiple *minsups* framework, each item in the database require a *minsup*-like constraint known as minimum item support (MIS). The methodology to specify items' MIS values is an open research problem in this framework.
- The frequent itemsets discovered with the multiple *minsups* framework do not satisfy the *anti-monotonic property*. This increases the search

space, which in turn increases the computational cost of mining the frequent itemsets. In addition, IMSApriori suffers from the performance problems involving the generation of huge number of candidate itemsets and multiple scans on the database. Thus NRRS is a computationally expensive recommender system.

- It has also been observed that NRRS can recommend uninteresting items to the users. Further details on this issue are discussed in later parts of this paper.

This paper makes an effort to improve the performance of NRRS. The key contributions are as follows:

1. An improved NRRS using the *relative support* measure (Yun et al. 2003) has been proposed to confront the coverage problem. We call it as NRRS++.
2. An improved EFI-graph structure, called EFI-growth++, has been introduced to improve the recommendations.
3. Experimental results on the real-world datasets demonstrate that NRRS++ can provide better recommendations and is runtime efficient than NRRS.

The rest of the paper is organized as follows. Section 2 describes background and related work on rule mining and rule-based recommender systems. Section 3 describes the NRRS. Section 4 describes the performance problems of NRRS and introduces the proposed NRRS++. The experimental evaluations of NRRS and NRRS++ have been reported in Section 5. Finally, Section 6 concludes the paper.

2 Background and Related Work

2.1 Association Rule Mining

Since the introduction of association rules by Agrawal et al. (1993), the problem of effective mining of rules from databases has received a great deal of attention (Han et al. 2007). An association rule mining algorithm generally involves the following two steps:

1. Discover all frequent itemsets that satisfy the user-defined minimum support (*minsup*) threshold.
2. Generate all strong rules that satisfy the user-defined minimum confidence (*minconf*) threshold using frequent itemsets.

The frequent itemset mining is a key step as it effectively reduces the search space and limits the number of rules getting generated. The popular algorithms to discover frequent itemsets are Apriori (Agrawal & Srikanth 1994) and FP-growth (Han et al. 2004). The Apriori uses "candidate generate-and-test approach," while FP-growth uses "pattern-growth technique" to discover the complete set of frequent itemsets. Generally, FP-growth performs better than Apriori as the latter suffers from the performance problems involving the generation of huge number of candidate itemsets and multiple scans on the database.

Since only a single *minsup* constraint is used for the entire dataset, the model implicitly assumes that all items in a database have uniform frequencies. However, this is often not the case in many real-world databases. In many real-world applications, some items appear very frequently in the data, while others rarely appear. If the items' frequencies in a database vary widely, we encounter the following issues:

i. If *minsup* is set too high, we will miss those rules that involve rare items.

ii. In order to find the rules involving both frequent and rare items, we have to set low *minsup*. However, this may cause combinatorial explosion, producing too many frequent itemsets, because those frequent items will combine with one another in all possible ways and many of them can be meaningless depending upon the user and/or application requirements.

This dilemma is called the *rare item problem* (Weiss 2004).

To confront the problem, the concept of finding frequent itemsets with multiple *minsups* framework has been introduced in the literature (Liu 1999). In this framework, each item in the database is specified with a *minsup*-like constraint known as the *minimum item support* (MIS), and *minsup* for an itemset is represented with the minimal MIS value among all its items. This framework can effectively confront the *rare item problem*. However, it has the following issues:

- This framework is computational expensive to use as the discovered frequent itemsets do not satisfy the anti-monotonic property.
- An open problem with this framework is the methodology to determine the items' MIS values.

Efforts are being made in the literature to address the above two issues (Kiran & Reddy 2009, 2010, 2011).

In the literature, researchers have also introduced other measures to discover frequent itemsets involving both frequent and rare items. Examples include *all-confidence*, *any-confidence*, *bond*, *lift*, *relative support* and χ^2 (Brin et al. 1997, Omiecinski 2003, Yun et al. 2003). Unlike the support measure, these measures assess the interestingness of an itemset with respect to the frequencies of items within itself. Each measure has a selection bias that justifies the significance of a knowledge itemset. As a result, there exists no universally acceptable best measure to judge the interestingness of an itemset for any given dataset. Researchers are making efforts to suggest a right measure depending upon the user and/or application requirements (Tan & Kumar 2002, Akshat et al. 2010).

The *relative support* measure was introduced in the literature to discover the frequent itemsets involving both frequent and rare items effectively (Yun et al. 2003) . An Apriori-like algorithm known as Relative Support Apriori (RSA) has been proposed to discover the itemsets. The performance issues of RSA are as follows:

- The rule model used in RSA requires the user's classification of items into either frequent or rare items. This classification can be difficult for the user in real-world applications.
- The frequent itemsets discovered with the relative support measure **do not satisfy the anti-monotonic property**. Thus, mining itemsets with RSA is a computationally expensive process.

Recently, it was shown in the literature that the measure *relative support* satisfies the *anti-monotonic*

property if items within an itemset are considered as an ordered set with respect to their supports (Kiran & Kitsure 2012). This property is known as the *convertible anti-monotonic property* (Pei & Han 2002). In addition, a simplified model which do not require the classification of items into either frequent or rare items has been proposed along with a pattern-growth algorithm, called Relative Support Frequent Pattern-Growth (RSFP-growth). The model is as follows:

Let $I = \{i_1, i_2, \cdots, i_n\}$ be a set of items, and DB be a database that consists of a set of transactions. Each transaction T contains a set of items such that $T \subseteq I$. Each transaction is associated with an identifier, called TID. Let $X \subseteq I$ be a set of items, referred as an itemset (or a *pattern*). An itemset that contains k items is a k-itemset. A transaction T is said to contain X if and only if $X \subseteq T$. The frequency (or support count) of an itemset X in DB, denoted as $f(X)$, is the number of transactions in DB containing X. The support of X, denoted as $S(X)$, is the ratio of its frequency to the DB size, i.e., $S(X) = \frac{f(X)}{|DB|}$. The relative support of an itemset X, denoted as $Rsup(X)$, is the ratio of its support to the minimum support of an item (or 1-itemset) within itself. That is, $Rsup(X) = \frac{S(X)}{min(S(i_j)|\forall i_j \in X)}$. An itemset X is said to be **frequent** if its support and relative support are no less than the user-defined minimum support ($minsup$) and minimum relative support ($minRsup$) thresholds, respectively. An association rule is an implication of the form, $A \Rightarrow B$, where $A \subset X$, $B \subset X$ and $A \cap B = \emptyset$. The confidence of a rule $A \Rightarrow B$, denoted as $C(A \Rightarrow B)$, represents the number of transactions in T that support A also support B. That is, $C(A \Rightarrow B) = \frac{S(A \cup B)}{S(A)}$. The rules which satisfy $minsup$, $minRsup$ and $minconf$ thresholds are called **strong rules**.

Example 1 *Consider the transactional database of 20 transactions shown in Table 1. The set of items* $I = \{a, b, c, d, e, f, g\}$. *The set of items 'a' and 'b', i.e.,* $\{a, b\}$ *is an itemset. It is a 2-itemset. For simplicity, we write this itemset as "ab". It occurs in 8 transactions (tids of* $1, 2, 7, 10, 11, 13, 16$ *and* 19*). Therefore, the support count of "ab," i.e.,* $f(ab) = 8$. *The support of ab is* $0.4 \left(= \frac{8}{20}\right)$. *The relative support of ab is* $0.88 \left(= \frac{0.4}{min(0.55, 0.45)}\right)$. *If the user-specified* $minsup = 0.3$ *and* $minRsup = 0.65$*, then ab is a frequent itemset because* $S(ab) \geq minsup$ *and* $Rsup(ab) \geq minRsup$. *The association rules generated from this itemset are* $a \Rightarrow b$ *and* $b \Rightarrow a$. *The confidence of* $a \Rightarrow b$*, i.e.,* $C(a \Rightarrow b) = \frac{0.4}{0.55} = 0.72$. *Similarly, the confidence of* $b \Rightarrow a$*, i.e.,* $C(b \Rightarrow a) = \frac{0.4}{0.45} = 0.88$. *If the user-specified* $minconf = 0.8$*, then only* $b \Rightarrow a$ *is a strong rule.*

In this paper, we use this model to discover frequent itemsets from the dataset generated using the purchased history of each user's group.

2.2 Association Rule-based Recommender System

Since the introduction of recommender systems in (Resnick et al. 1995), they have received a great deal of attention from both industry and academia (Adomavicius et al. 2001). The popular approaches to improve the performance of recommender systems involved the usage of machine learning techniques (Billsus & Pazzani 1998) and data mining techniques (Lin et al. 2002). In this paper, we focus on the usage of data mining techniques to improve the performance of recommender systems.

Table 1: Transactional database.

TID	Items	TID	Items
1	a, b	11	a, b
2	a, b, e	12	a, c, f
3	c, d	13	a, b, e
4	e, f	14	b, e, f, g
5	c, d	15	c, d
6	a, c	16	a, b
7	a, b	17	c, d
8	e, f	18	a, c
9	c, d,g	19	a, b
10	a, b	20	c, d, f

Association rule mining has been widely used in developing recommender systems (Adomavicius et al. 2001, Smyth et al. 2005). Adomavicius & Tuzhilin. (2005) have been employed association rules to build behavioral profiles of individual users. Shaw et al. (2010) have used association rules to address cold-start problem in the recommender systems. The association rule model used in most of these works is based on the single *minsup* framework, where only a single *minsup* constraint is used to discover frequent itemsets from the entire database. Since the single *minsup* framework cannot effectively capture the non-uniform frequencies of items in a database, these recommender systems suffer from the problem of reduced coverage.

To confront the problem, "adaptive-support" framework has been proposed in (Lin et al. 2002). In this framework, a *minsup* threshold is determined individually for each user's (or item's) database. This framework can still suffer from the coverage problem if items' frequencies in the database vary widely.

The concept of rule mining with multiple *minsups* framework was introduced in (Gedikli & Jannach 2010) to confront the coverage problem. In addition, NRRS was proposed to recommend the products to the users. It has been observed that NRRS is computationally expensive to use and can recommend uninteresting items to the users. In this paper, we address these two issues and propose an improved NRRS.

3 Neighborhood-Restricted Rule-Based Recommender System

The NRRS uses the concept of mining frequent itemsets with the multiple *minsups* framework to address the coverage problem that persists in the traditional rule-based recommender system (Gedikli & Jannach 2010). In particular, it employs user-based collaborative filtering technique and uses two neighborhood sizes for recommending items to the users. Briefly, 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, the complete set of frequent itemsets are discovered from the database using IMSAprori algorithm. The items' MIS values are specified using Equation 1 (Kiran & Reddy 2009).

$$MIS(i_j) = max(S(i_j) - SD,\ LS) \quad (1)$$

where, $MIS(i_j)$ is the minimum item support for an item i_j, $S(i_j)$ refers to support of an item i_j, SD refers to the support difference and LS refers to the lowest minimum item support allowed for an item. The parameter SD can be either user-specified or derived using Equation 2

$$SD = \lambda(1 - \beta) \quad (2)$$

where, λ represents the parameters like mean, meadian, or mode of items supports in a database and $\beta \in [0,\ 1]$ is a user-specified constant.

2. (**Online phase.**) For each user u, the discovered frequent itemsets of top-k_2 number of neighboring users are stored in a lexical order in a tree-structure, called Extended Frequent Itemset Graph (EFI-graph). Recommendations to the user u are generated by traversing EFI-graph using depth-first search.

We now describe and illustrate the structure and working of EFI-graph for the product recommendation.

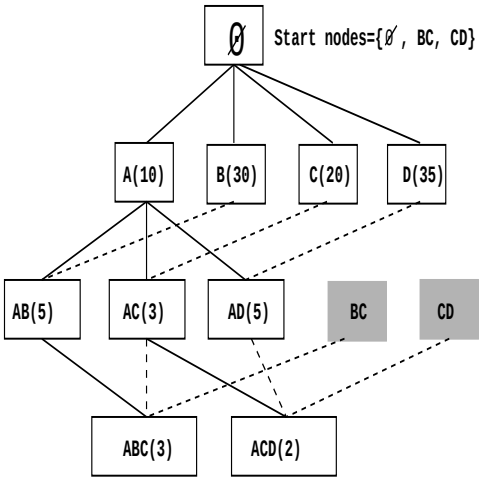

Figure 1: Extended Frequent Itemset Graph approach.

Lin et al. (2002) have introduced Frequent Itemset Graph (FI-graph) to make real-time recommendations without explicitly learning rules derived from the frequent itemsets of single *minsups* framework. Thus, FI-graph can be used to store only those frequent itemsets which satisfy the anti-monotonic property. The frequent itemsets discovered with the multiple *minsups* framework do not satisfy the anti-monotonic property. Therefore, FI-graph cannot be used to store the frequent itemsets discovered with the multiple *minsups* framework. To address this problem, Yun et al. (2003) have extended the FI-graph to store the frequent itemsets discovered with the multiple *minsups* framework. It is called Extended Frequent Itemset Graph (EFI-graph). In EFI-graph (see Figure 1), nodes represent both frequent and infrequent itemsets that are arranged in lexicographical order, and organized in a tree structure such that the size of the itemsets gets increased on each level. The nodes representing frequent itemsets preserve their respective support value. The nodes representing infrequent itemsets cannot preserve their respective support value as they were pruned by the pattern discovery algorithm (or IMSApriori). If a parent node represents a frequent itemset, then the links connecting the respective parent node and child nodes will preserve the confidence value of an association rule. If a parent node represents an infrequent itemset, then the links connecting the respective parent node and child nodes will not preserve the confidence of an association rule. The recommendations to the user are made by employing depth-first search on the EFI-graph.

Example 2 *Let the set of frequent itemsets discovered from the transactional database of the user u be* $\{\{A : 10\}, \{B : 20\}, \{C : 30\}, \{D : 35\}, \{AB : 5\}, \{AC : 3\}, \{AD : 5\}, \{ABC : 3\}, \{ACD : 2\}\}$ *(i.e.,* $\{$*itemset, support count*$\}$*). Figure 1 shows an EFI-graph constructed for the discovered frequent itemsets. The numbers in brackets of a node stand for the support count of the respective frequent itemset in the database. The nodes with no support count represent the infrequent itemsets. In other words, the nodes* $\{BC\}$ *and* $\{CD\}$ *in EFI-graph represent the infrequent itemsets whose supersets are frequent. The parent node* $\{A, D\}$ *represents a frequent itemset. Therefore, the link containing the* $\{A, D\}$ *and* $\{A, C, D\}$ *nodes preserve the confidence of the rule* $\{A, D\} \Rightarrow \{C\}$*, i.e.,* $C(\{A, D\} \Rightarrow \{C\}) = \frac{S(\{A,C,D\})}{S(\{A,D\})} = \frac{2}{5}$*. The parent node* $\{C, D\}$ *represents an infrequent itemset. Therefore, the link connecting* $\{C, D\}$ *and* $\{A, C, D\}$ *do not preserve the confidence of the rule* $\{C, D\} \Rightarrow \{A\}$*. Given a set of past transactions* $T = \{A, D\}$ *of user u, recommendations are produced by traversing the tree in depth-first order and looking for (single-element) supersets of* $\{A, D\}$ *in the graph that has high confidence. That is, C could be recommended to u if the recommendation score of item C is high enough. On the other hand, if the set of past transactions of user u represent an infrequent itemset, say* $T = \{C, D\}$*, then the item A will be recommended by simply traversing the tree in depth-first order looking for supersets of* $\{C, D\}$ *without any confidence value. Please note that the solid arrows in the figure indicate how the graph would be traversed in depth-first order if past transactions of a user represent* $\{A, D\}$*.*

In the next section, we discuss the performance problems of NRRS and describe the proposed NRRS++.

4 Improved Neighborhood Restricted Association Rule-based Recommender System

In this section, we first discuss the performance issues of NRRS. Next, we describe the proposed NRRS++.

4.1 Performance Problems of NRRS

The NRRS uses the frequent itemsets discovered from the multiple *minsups* framework to recommend products to the users. Since the multiple *minsups* framework suffers from the open problem of determining items' MIS values, NRRS is difficult to be used in real-world applications.

The frequent itemsets discovered with multiple *minsups* do not satisfy the anti-monotonic property.

Therefore, IMSApriori is a computationally expensive algorithm. Since NRRS uses IMSApriori to discover frequent itemsets for each user's database, it is straight forward to say that NRRS is a computationally expensive recommender system.

In EFI-graph, the links connecting the parent node of an infrequent itemset to their respective child nodes do not preserve the confidence of a rule. It has been observed that this may lead to recommend uninteresting items to the user.

Example 3 *Continuing with the Example 2, NRRS recommends item A for the user u by simply traversing down the node of the infrequent itemset* $\{C, D\}$. *If the support count of an infrequent itemset* $\{C, D\}$ *is 14, then confidence of the rule,* $\{C, D\} \Rightarrow \{A\} = 0.14\ (= \frac{2}{14})$, *is very low. Thus, it can be arguable that NRRS may be suggesting an uninteresting item A for the user u.*

Algorithm 1 NRRS++ (U: set of users, $ratingDB$: rating database, k_1: learning neighborhood size, k_2: predicting neighborhood size $minsup$: user-specified minimum support threshold, $minRsup$: user-specified minimum relative support threshold)

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 RSFP-growth. That is, $UserFPs = RSFP\text{-}growth(ratingDB^u, minsup, minRsup)$.
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 = adjustConfidenceScores-BySimilarity($userRecs, u, u_i$);
14: $recommendedItems = recommendedItems \cup weightedUserRecs$;
15: **end for**
16: $recommendedItems = sortItemsByAdjusted\text{-}Scores(recommendedItems)$;
17: output $recommendedItems$;
18: **end for**

4.2 An Improved Neighborhood-Restricted Rule-based Recommender System: NRRS++

To address the open problem of specifying items' MIS values in NRRS, we use the relative support measure to discover frequent itemsets involving both frequent and rare items. The advantages of using the relative support measure are as follows:

- The measure relative support satisfies the convertible anti-monotonic property. Pei & Han (2002) have theoretically shown that for a pattern-growth algorithm the search space of a measure satisfying the convertible anti-monotonic property is same as that of the anti-monotonic property. Thus, frequent itemsets can be effectively discovered using the RSFP-growth.
- This measure satisfies the null-invariance property. This property facilitates the measure relative support to disclose genuine correlation relationships without being influenced by the object co-absence in a database.

To improve the predictions of NRRS, we have modified the structure of EFI-graph so that the links connecting the parent nodes of infrequent itemsets to their respective child nodes can preserve the confidence of a rule. We now describe the working of NRRS++.

The proposed NRRS++ system is shown in Algorithm 1, and summarized as follows. The input parameters to NRRS++ are the set of users (U), the ratings database of each user ($ratingDB$), the two neighborhood sizes for rule learning and prediction phases (k_1 and k_2), user-specific minimum support ($minsup$) constant, user-specific minimum relative support ($minRsup$) constant. The parameters $minsup$ and $minRsup$ are used in RSFP-growth algorithm to discover frequent itemsets. The NRRS++ 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. 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 RSFP-growth algorithm (Kiran & Kitsure 2012).
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 the proposed Extended Frequent Itemset Graph, i.e., EFI-graph++ (lines 8 to 13 in Algorithm 1). The resulting confidence scores are weighted according to the similarity of the target user and the neighbor (line 14 to 16 in Algorithm 1). 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 1).

Now, we introduce EFI-graph++ structure.

The EFI-graph++ (see Figure 2) is an extension of EFI-graph to store the support counts of both frequent and infrequent itemsets. The structure of EFI-graph++ resembles that of EFI-graph. However, the major difference is that the nodes representing the infrequent itemsets will also store their respective support counts. The construction of EFI-graph++ is shown in Algorithm 2.

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 2 and 3 in Algorithm 2). Lines 4 to 14 in Algorithm 2 connect each $k-1$ subset s of a k-frequent itemset with its superset f. If subset s is an infrequent itemset, i.e., $s \notin F$, then add s into the list of infrequent itemset IF, create a new

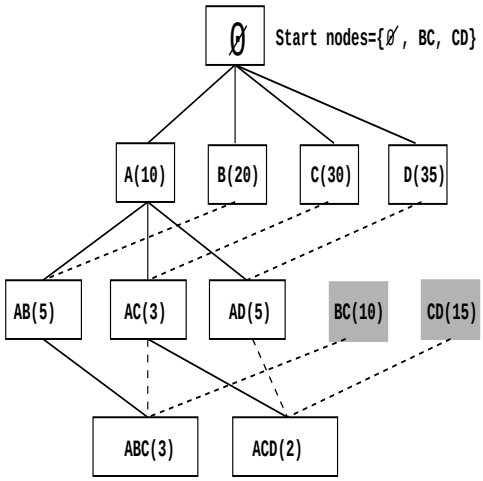

Figure 2: Proposed Extended Frequent Itemset Graph approach.

node with s and support count equal to zero and add it to the 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 infrequent itemsets whose supersets are frequent itemsets. (Lines 5 to 9 in Algorithm 2). If the subset s exists in the frequent itemset, then a node with s and corresponding support count is created and linked to the node f (Lines 10 to 12 in Algorithm 2). Finally, the support counts of the nodes representing the infrequent itemsets are updated by performing another scan on the database (Lines 17 and 18 in Algorithm 2).

Algorithm 2 EFIG++(F: the set of frequent itemsets sorted in descending order of the number of items within each itemset.)

1: Let n be the number of frequent itemsets discovered by CFG. Let IF be the set of infrequent itemsets whose supersets exists in F. Initialize $IF = \emptyset$.
2: **for** $k = n$ to 1 **do**
3: **for** all frequent k-itemsets f **do**
4: **for** all $k-1$ subsets s of f **do**
5: **if** $s \notin F$ **then**
6: create a new node $(s, 0)$;
7: add new edge (s, f);
8: add new start node s
9: $IF = IF \cup s$; // add infrequent itemset to the set of IF;
10: **else**
11: create a new node (s, *support*);
12: add new edge (s, f);
13: **end if**
14: **end for**
15: **end for**
16: **end for**
17: Scan the database and find the support of every infrequent itemset in IF.
18: In EFI-graph++, update the support of every infrequent itemset node.

Example 4 *Figure 2 shows the EFI-graph++ constructed for the user u discussed in Example 2. It can be observed that EFI-graph++ stores the support of infrequent itemsets, while EFI-graph (shown in Figure 1) do not store the support of infrequent itemsets. The storing of support values for infrequent itemsets facilitates the proposed NRRS++ framework to improve the recommends to the users by pruning the recommendations of those items that have low confidence value.*

5 Experimental Results

In this section, we evaluate the NRRS and proposed NRRS++. The programs were written in java and run with Ubuntu on a 2.66 GHz machine with 2GB memory. In the following, we will summarize the findings of this evaluation.

5.1 Experimental Setup and Evaluation Metrics

5.1.1 Data sets.

The experiments were pursued on the following datasets: (i) MovieLens rating dataset consisting of 100,000 ratings provided by 943 users on 1,682 items and (ii) Yahoo!Movies dataset containing 211,231 ratings provided by 7,642 users on 11,915 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 sets were split into a 80% training set and a 20% test set.

5.1.2 Accuracy metrics.

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.

In the evaluation procedure, we use "top-10," i.e., the list of top ten movies for a test user with predicted ratings values above the user's mean ratings, 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. That is,

$$F = 2 \times \left(\frac{precision \times recall}{precision + recall} \right). \quad (3)$$

5.1.3 Algorithm details and Parameters.

The sensitivity of parameters was analyzed through multiple experiments on the MovieLens and Yahoo! data set with different density levels. The neighborhood sizes, k_1 and k_2, for both datasets and all density levels was empirically determined to be 900 and 60, respectively. For both the datasets, λ represented mean of all frequent items, $\beta = 0.25$ and $LS = 9\%$.

5.2 Results

Table 2 shows the performance of NRRS and NRRS++ systems on MovieLens and Yahoo!Movies datasets 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.

Fig. 3 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 that CFG discovered frequent itemsets for each user more effectively than that of IMSApriori. Another important observation is that NRRS++ is more scalable than NRRS with increase in neighborhood size for learning the rules.

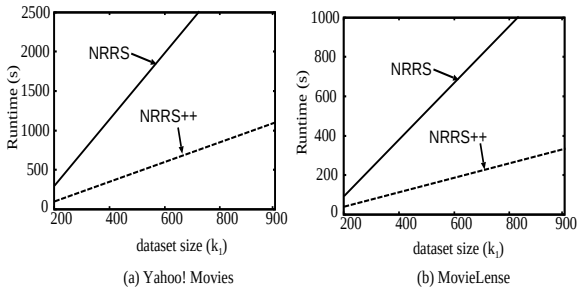

Figure 3: Runtime comparison of NRRS and NRRS++ Systems.

6 Conclusion

This paper proposed an effective and efficient rule-based recommender system known as NRRS++ to confront the coverage problem that persists in the conventional rule-based recommender systems. It has also shown that frequent itemsets discovered with the relative support measure satisfy the convertible anti-monotonic property, and therefore, can be effectively discovered using CFG algorithm. Furthermore, an efficient tree structure has been proposed to improve the recommendations to the users if the discovered itemsets satisfy the convertible anti-monotonic property. Experimental results demonstrate that NRRS++ is efficient than NRRS.

References

Abel, F., Bittencourt, I. I., Henze, N., Krause, D. & Vassileva, J. (2008), A Rule-Based Recommender System for Online Discussion Forums, *in* Proceedings of the 5th international conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 12–21.

Adomavicius, G. & Tuzhilin, A. (2005), Expert-Driven Validation of Rule-Based User Models in Personalization Applications, *Data Mining and Knowledge Discovery*, **5**(1-2), 33–58.

Adomavicius, G. & Tuzhilin, A. (2005), Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions,

Table 2: Performance comparison of NRRS and NRRS++ systems at different density levels in MoviesLense and Yahoo! Movies datasets.

		Density	10%	20%	30%	40%	50%	60%	70%	80%	90%
Movie Lens	F1	NRRS	35.75	41.61	47.75	52.71	57.89	60.85	60.91	61.66	62.72
		NRRS++	37.30	44.46	48.93	54.9	59.46	62.56	63.5	63.95	64.64
	Precision	NRRS	45.9%	54.64%	60.23%	62.32%	63.61%	64.7%	64.7%	64.7%	63.8%
		NRRS++	47.8%	56.37%	61.56%	64.44%	65.61%	66.71%	66.8%	66.8%	66.71%
	Recall	NRRS	29.3%	33.61%	39.56%	45.67%	53.12%	57.43%	57.54%	58.91%	61.69%
		NRRS++	30.58%	36.71%	40.61%	47.82%	54.38%	58.9%	60.52%	61.34%	62.71%
Yahoo! Movies	F1	NRRS	13.89	19.04	25.4	31.3	36.33	39.46	42.16	44.44	45.95
		NRRS++	15.21	21.25	26.15	32.78	37.19	40.55	43.03	45.6	46.95
	Precision	NRRS	15.6%	22.76%	28.91%	38.12%	43.21%	47.84%	51.65%	53.86%	56.82%
		NRRS++	17.01%	25.12%	30.1%	39.71%	44.65%	49.16%	51.98%	56.73%	57.31%
	Recall	NRRS	12.52%	16.37%	22.65%	26.55%	31.34%	33.58%	35.62%	37.83%	38.58%
		NRRS++	13.76%	18.42%	23.12%	27.92%	31.87%	34.51%	36.72%	38.13%	39.76%

IEEE Transactions on Knowledge and Data Engineering, **17**(6), 734–749.

Agrawal, R., Imielinski, T. & Swami, A. (1993), Mining association rules between sets of items in large databases, *in* 'ACM SIGMOD International Conference on Management of Data', Vol. 22, ACM Press, Washington DC, USA, pp. 207–216.

Agrawal, R. & Srikanth, R. (1994), Fast Algorithms for Mining Association Rules in Large Databases, *in* Proceedings of VLDB, pp. 487–499.

Akshat, S. & Kiran, R. U. & Reddy, P. K. (2010), Selecting Right Interestingness Measures for Mining Rare Association Rules *in* Proceedings of COMAD, pp. 115 – 124.

Billsus, D., Pazzani, M. J., (1998), Learning Collaborative Information Filters. *in* Proceedings of the Fifteenth Conference on Machine Learning, pp. 46–54.

Brin, S., Motwani, R. & Silverstein, C., (1997), Beyond market baskets: generalizing association rules to correlations, *in* Proceedings of ACM SIGMOD, pp. 265–276.

Gedikli, F. & Jannach, D. (2010), Neighborhood-Restricted Mining and Weighted Application of Association Rules for Recommenders, *in* Proceedings of International Conference on Web Information System Engineering, pp. 157–165.

Han, J., Pei, J., Yin, Y. & Mao, R. (2004), Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, *Data Mining Knowledge Discovery*, **8**(1), 53–87.

Han, J., Cheng, H., Xin, D., & Yan, X. (2007), Frequent Pattern Mining: Current Status and Future Directions. *Data Mining Knowledge Discovery*, **15** (1), 55–86.

Kiran, R. U. & Reddy, P. K. (2009), An Improved Multiple Minimum Support Based Approach to Mine Rare Association Rules, *in* Proceedings of CIDM, pp. 340–347.

Kiran, R. U. & Reddy, P. K. (2010), Mining Rare Association Rules in the Datasets with Widely Varying Items' Frequencies, *in* Proceedings of DASFAA, pp. 49–62.

Kiran, R. U. & Reddy, P. K. (2011), Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms, *in* Proceedings of EDBT, pp. 11–20.

Kiran, R. U. & Kitsure, M. (2012), Towards Efficient Discovery of Frequent Patterns with Relative Support *in* To be appeared in the Proceedings of COMAD.

Lin, W., Alvarez, S. A. & Ruiz, C. (2002), Efficient Adaptive-Support Association Rule Mining for Recommender Systems, *Data Mining and Knowledge Discovery*, **6**(1), 83–105.

Liu, B. (1999), Mining association rules with multiple minimum supports, *in* Proceedings of Knowledge Discovery in Databases, pp. 337–341.

Omiecinski, E. R., (2003), Alternative interest measures for mining associations in databases, *IEEE Trans. on Knowl. and Data Eng.*, **15**(1), 57–69.

Pei, J. & Han, J. (2002), Constrained frequent pattern mining: a pattern-growth view, *SIGKDD Explor. Newsl.*, **4**(1), 31–39.

Resnick, P., Iacovou. N., Suchak, M., Bergstrom, P., (1995), GroupLens: an Open Architecture for Collaborative Filtering of Netnews. *in* Proceedings of the Conference on Computer Supported Cooperative Work, pp. 175–186.

Shaw, G., Xu, Y. & Geva, S. (2010), Using Association Rules to Solve the Cold-Start Problem in Recommender Systems, *in* Proceedings of PAKDD 2010, pp. 340–347.

Smyth, B., McCarthy, K., Relly, J., O'Sullivan, D., McGinty, L., & Wilson, D. C. (2005), Case Studies in Association Rule Mining for Recommender System, *in* Proceedings of IC-AI 2005, pp. 809–815.

Tan, P. & Kumar, V. (2002), Selecting the Right Interestingness Measure for Association Patterns, *in* Proceedings of KDD, pp. 32–41.

Weiss, G. M. (2004), Mining with rarity+ a unifying framework, *SIGKDD Explor. Newsl.*, **6**(1), 7–19.

Yun, H., Ha, D., Hwang, B. & Ryu, K. H. (2003), Mining association rules on significant rare data using relative support, *Journal of Systems and Software*, **67**(3), 181–191.