Efficient Adaptive–Support Association Rule Mining for Recommender Systems

WEIYANG LIN weiyangl@microsoft.com
SVC-3/2955, Microsoft Corporation, 1065 La Avenida, Mountain View, CA 94043, USA

SERGIO A. ALVAREZ alvarez@cs.bc.edu
Department of Computer Science, Boston College, Chestnut Hill, MA 02467, USA

CAROLINA RUIZ ruiz@cs.wpi.edu
Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA 01609, USA

Abstract. Collaborative recommender systems allow personalization for e-commerce by exploiting similarities and dissimilarities among customers' preferences. We investigate the use of association rule mining as an underlying technology for collaborative recommender systems. Association rules have been used with success in other domains. However, most currently existing association rule mining algorithms were designed with market basket analysis in mind. Such algorithms are inefficient for collaborative recommendation because they mine many rules that are not relevant to a given user. Also, it is necessary to specify the minimum support of the mined rules in advance, often leading to either too many or too few rules; this negatively impacts the performance of the overall system. We describe a collaborative recommendation technique based on a new algorithm specifically designed to mine association rules for this purpose. Our algorithm does not require the minimum support to be specified in advance. Rather, a target range is given for the number of rules, and the algorithm adjusts the minimum support for each user in order to obtain a ruleset whose size is in the desired range. Rules are mined for a specific target user, reducing the time required for the mining process. We employ associations between users as well as associations between items in making recommendations. Experimental evaluation of a system based on our algorithm reveals performance that is significantly better than that of traditional correlation-based approaches.

1. Introduction

The ability to deliver personalized goods and services is becoming a key issue in determining the success of online businesses. Faced with an enormous variety of options, customers surfing the web gravitate toward sites that offer information tailored to their personal preferences. Collaborative recommender systems offer a technology that allows personalized recommendations of items of potential interest to users based on information about similarities and dissimilarities among different users' tastes (see e.g. (Shardanand and Maes, 1995), (Resnick et al., 1994), (Balabanovic and Shoham, 1997)). However, despite enormous interest in collaborative recommender technology, both the number of available published techniques and information about their performance are quite limited. Most available systems rely on simple methods to represent and measure similarities among users, in particular the linear correlation coefficient for a given pair of users. Recent work using machine learning techniques (Billsus and Pazzani, 1998) has shown that it is possible to obtain improved recommendation performance by using more sophisticated methods. Concurrently, and in a different direction, the field of data mining has given rise to effective methods for identifying patterns in large datasets. Conditions are ripe for the application of these new methods to electronic commerce, and in particular to recommender systems.

We are particularly interested here in the data mining paradigm of association rules (Agrawal et al., 1993). The technical core of this paradigm is the mining algorithm used to extract significant associations from available transaction data. Existing algorithms, such as the Apriori algorithm (Agrawal and Srikant, 1994), were designed with market basket analysis in mind. Market basket analysis is directed toward establishing "aggregate" customer behaviors; information about such behaviors allow optimized store layouts for example. As a result, market basket-oriented rule mining algorithms seek rules with arbitrary heads (consequents), a task which requires a great amount of time. In contrast, recommender systems seek to provide personalized recommendations for specific users. In terms of rules, this means that the head can be specified in advance. Many existing rule mining algorithms are unable to exploit this restriction in order to improve the efficiency of the mining process. There do exist classification-oriented algorithms such as CBA-RG (Liu et al., 1998) that are capable of mining rules with only one target attribute in the head. Such algorithms can achieve improved efficiency compared with those that mine unrestricted rules. Nonetheless, such algorithms were not designed for recommendation and so they do not take advantage of the fact that the mining task for recommender systems is even more specific than for classification in general. First of all, a specific value of the target attribute may be assumed. Also, for recommendation it is not necessary to mine an arbitrarily large ruleset containing all rules above a pre-specified minimum support level (see section 2.1 for definitions). Because of this, further work is needed in this area.

1.1. Contributions of this paper

In the present paper we describe collaborative recommendation using an association rule mining technique specially designed for this purpose. A significant feature and advantage of our mining technique compared with existing methods is that it does not require that the minimum support of the rules to be mined be specified in advance. Rather, our algorithm automatically adjusts the minimum support during mining so that the number of rules generated lies within the desired range. This provides enough rules for high quality recommendations, but without spending excessive time exhaustively mining all rules above a prespecified minimum support as standard mining algorithms would. We include experimental results that show that excellent results may be obtained using this technique. In particular, recommendation performance is significantly better than that obtained using traditional recommender system technologies.

1.1.1. *Recommendation based on Association Rules*

Our motivation to mine association rules for recommender systems is based on the observation that rules such as "90% of users who like article A and article B also like article C, 30% of all users like all of them" and "90% of articles liked by user A and user B are also liked by user C, 30% of all articles are liked by all of them" are very useful for recommendation purposes. We refer to rules of the first kind as *article associations* and rules of the second kind as *user associations*. We explore article associations and user associations on two levels (*like* and *dislike*) by using extensions of the basic association rules. One example of two level user associations is "90% of articles liked by user A and disliked by user B are liked by user C, 30% of all articles are liked by user A and C and disliked by user B". We have developed a system that can be configured to employ either user associations, article associations, or a combination of the two. Our recommendation strategy involves attaching a score to each article based on the confidence and support over the training data of the rules recommending the given article; the articles with the highest scores are recommended. This strategy is described in section 3.

Our approach has the following advantages.

1. The association rules framework provides two useful measures for evaluating the association expressed by a rule. The *confidence* of a rule measures the degree of the correlation among users or among articles, while the *support* of a rule measures the significance of the correlation among users or among articles. These terms are defined in the next section of the paper.

2. Instead of identifying users whose tastes are similar to that of a given user, we can use overlaps of other users' tastes to match the given user's taste. For example, the rule "90% of articles liked by user A and user B are also liked by user C" uses the overlap of user A's and user B's

tastes to match user C's taste. This allows making good recommendations to users whose tastes don't correlate strongly with those of other individual users.

3. One user's ratings might be useful for predicting another user's ratings even though they are not correlated. For example, negative ratings from user A might predict positive ratings of user B, even though user A's positive ratings do not imply negative ratings of user B. This is also applicable to relationships between articles.

1.1.2. *Efficient Association Rule Mining*

The association rule mining process employed by Apriori (Agrawal and Srikant, 1994) and similar algorithms is computationally very expensive. We have developed a new mining algorithm that achieves improved efficiency by exploiting the specialized nature of the classification task needed for recommender systems.

First, our algorithm focuses on mining rules for only one target user or article at a time. Since we need to mine user associations online, the efficiency of the process is of particular importance in this case. We are only interested in predicting articles that a target user would like, so we only need rules with the target user in the rule head. Also, by mining article associations for one article at a time we are able to obtain rules for articles that have only received a limited number of ratings, for example a new movie. This would not be possible if we mined article associations for all articles at once, because rules for new articles would fail to have the necessary support. Furthermore, a significant amount of runtime is saved by mining rules only over the subset of the transaction data that is related to the target user/article instead of over the whole data.

Perhaps more significantly, we observe that the nature of recommendation suggests addressing a modified version of the usual association rule mining problem, in which instead of specifying a minimum support, a range for the desired number of rules is given before the mining process. We have designed and implemented an algorithm to mine association rules which solves this modified version of the rule mining problem. Our algorithm automatically selects the minimum support so that the mining process produces an appropriate number of rules for each target item. This property of our algorithm allows recommender systems that rely on the rules mined by it to achieve very good performance both in terms of response time and accuracy of the predictions.

1.2. Relation to other work

Collaborative recommendation has been the subject of research over the past decade. Now-classical work in this area includes the papers by Shardanand and Maes (Shardanand and Maes, 1995) and Resnick et al (Resnick et al., 1994). We will briefly describe some of the more widely used techniques for collaborative recommendation below. We also describe the relation of our association rule mining framework to other rule mining approaches.

1.2.1. *Linear correlation-based method*

(Shardanand and Maes, 1995) and (Resnick et al., 1994) put forth several variants of a technique based on the statistical (Pearson) correlation between two users' ratings. This technique has since become widely used. The correlation formula predicts that the rating given by user U to item I will differ from user U's mean rating by a linear combination of the deviations from the mean of other users' ratings for item I, with weights proportional to the correlations between those users' ratings and those of the target user U. Variants of this approach that already appear in (Shardanand and Maes, 1995) include threshold values for the correlations, so that only users whose correlation with the target user exceeds the threshold are considered in the linear combination that gives the predicted rating for the target user.

A disadvantage of the correlation-based techniques described above lies in the simple linear formula that they use to make predictions. Users whose preferences are very clearly related, except not via a linear relation, may not be correlated at all and thus the predictive information present in their behavior patterns will be ignored altogether by correlation-based methods. Other drawbacks of the correlation approach are mentioned in (Billsus and Pazzani, 1998): the significance of the correlations between users is not measured, and, most importantly, if two users do not rate articles in common, they can not be similar under the correlation method even if they share common interests. Our approach can possibly overcome these drawbacks.

1.2.2. *Bayesian Classifier and Bayesian Network Model*
Breese, Heckerman and Kadie (Breese et al., 1998) list and test several algorithms for collaborative recommendation. They propose a new approach for finding dependences among articles by using a Bayesian classifier and a Bayesian network model. The idea of this approach is similar to our article associations. But (1) they need to calculate the conditional probabilities of all the possible ratings for an article given all possible ratings for other articles, which is computationally expensive and (2) they can not estimate how good a prediction they made is. Our approach (1) only needs to find some significant dependences among articles, which is above a certain minimum support and (2) we can evaluate a prediction according to the support and confidence of the rule. Since we are only concerned with recommending a certain number of interesting articles, not predicting the ratings for all articles, the significant dependencies are good enough.

1.2.3. *Neural Networks Paired with Feature Reduction Techniques*
Billsus and Pazzani (Billsus and Pazzani, 1998) present a framework for applying machine learning algorithms paired with feature reduction techniques, such as singular value decomposition (SVD) or information gain, for collaborative recommendations. They use feature reduction techniques to reduce the dimension of the rating data, and then neural networks are applied to the simplified data to construct a model for recommendation. In section 4 we compare our approach with this approach as well as the correlation-based method under some similar experimental conditions.

1.2.4. *Association Rules*
A large variety of association rule frameworks and algorithms have been published in the literature, including Apriori (Agrawal and Srikant, 1994), GUHA (Hájek et al., 1966), and DIS (Brin et al., 1997). One extension of the basic binary association rules, called quantitative association rules (Srikant and Agrawal, 1996), finds associations between attributes with categorical values. Quantitative association rules have the potential to extend association rules to general classification domains. Some results of adapting those rules to classification tasks are shown in (Liu et al., 1998; Hájek and Havranek, 1977). (Liu et al., 1998) presents the CBA-RG algorithm (which is based on the Apriori algorithm) and a good framework to perform the so-called associative classification.

Association rules have previously been used in web mining. They have been used to mine path traversal patterns and to facilitate the best design and organization of web pages (Cooley et al., 1997a; Cooley et al., 1997b; Cooley et al., 1999; Chen et al., 1998). For the domain of recommender systems, Fu, Budzik and Hammond (Fu et al., 2000) have recently developed a system for recommending web pages using the Apriori algorithm to mine association rules over users' navigation histories. Sarwar (Sarwar, 2001) describes his approach to using a traditional association rule mining algorithm to find rules for Top-N recommendation. He deals with the complexity of the process by pre–selecting a reduced number of collaborative users who are the closest ones to the target user.

However, most previously proposed association rule mining algorithms are not suitable for the intended domains as described above. A significant reason for this is that previously proposed algorithms do not provide a mechanism to choose a proper minimum support for the given minimum confidence and the desired range for the number of rules. This often leads to either too many or too few rules, and thus to either excessive computation time or else poor classification performance. Recent work (Webb, 2000; Agarwal et al., 2000) has begun to address some of these concerns by means of judicious search techniques. The present paper offers a different approach that is closer in spirit to the two-stage Apriori mining algorithm but that achieves significantly improved efficiency.

2. Association Rule Mining

This section describes our new framework for association rule mining that is specially designed for use in collaborative recommendation. We begin with some background knowledge on association rules.

The framework of association rules was introduced into the data mining community at large by Agrawal et al. (Agrawal et al., 1993). Their work was motivated by the desire to mine associations over sales transactions for market basket analysis and stressed the algorithmic aspects of the mining process. Earlier, P. Hájek et al. (Hájek et al., 1966; Hájek and Havranek, 1977) had anticipated many of the same concepts and approaches, focusing on the representational power of association rules, with a view toward their application in scientific discovery.

2.1. Definitions

We now introduce some of the basic terminology of association rules. A *transaction* is a set of items; in the original market basket scenario described in (Agrawal et al., 1993), the items of a transaction represent items that were purchased concurrently by a user. An *association rule* is a rule of the form X $\Rightarrow$ Y, where X and Y are sets of items (or *itemsets*); X and Y are called respectively the *body* and the *head* of the rule. The intended meaning of this rule is that the presence of (all of the items of) X in a transaction implies the presence of (all of the items of) Y in the same transaction with some probability.

Each association rule X $\Rightarrow$ Y has two measures relative to a given set of transactions: its *confidence* and its *support*. The confidence of the rule is the percentage of transactions that contain Y among transactions that contain X; The support of the rule is the percentage of transactions that contain both X and Y among all transactions in the input data set. In other words, the confidence of a rule measures the degree of the correlation between itemsets, while the support of a rule measures the significance of the correlation between itemsets.

As an example, assume that we have a database of transactions as listed in Table I, for association rule "{A} $\Rightarrow$ {B, C}", the confidence of the rule is 66%, and the support of the rule is 50%.

Table I. Sample Transactions

Transaction Id	Purchased Items
1	{A, B, C}
2	{A, D}
3	{A, B, C, E}
4	{B, E, F}

2.2. Traditional Association Rule Mining Problem Definition

The traditional association rule mining problem definition is: *given a set of transactions, a user-specified minimum support and minimum confidence, find all association rules that are above the user-specified minimum support and minimum confidence.* This problem definition is well suited to uncovering aggregate customer behaviors as required in traditional market basket analysis. However, for reasons explained below, we believe that this problem definition should be modified if the goal is to mine association rules for the purpose of recommending items to specific target users.

2.3. Our New Association Rule Mining Problem Definition

We propose the following new problem definition for mining association rules for classification. In keeping with the usual requirements of recommender systems, we assume that a single *target item* is specified in advance. In the case of user associations, the target item is simply the user for whom recommendations are to be made. However, in order to cover both user associations and article associations, we use the term target item here to denote a user in the case of user associations and an article in the case of article associations.

Given: a transaction dataset, a target item, a specified minimum confidence and a desired range [minNumrules,maxNumrules] for the number of rules,

Find: association rules with the target item in the heads of the rules such that the number of rules is in the given range (unless fewer than minNumrules exist for the given minimum confidence), the rules satisfy the minimum confidence constraint, and the rules have the highest possible support (i.e. no rule outside S with confidence greater than or equal to the minimum confidence has a higher support than a rule in S).

In classification tasks, this new problem definition has the advantage that it focuses on mining rules for only one target item at a time. Such rules could be mined more efficiently than rules with arbitrary heads. Also, significant amount of runtime is saved by mining rules only over the subset of the transaction data that is related to the target item instead of over the whole data. When the application domain requires mining association rules online, the efficiency of the process is of great importance. Furthermore, by mining rules for one item at a time we are able to obtain classification rules even for items that appear only in a limited number of transactions. This would not be possible if we mined associations for all items at once, because rules for infrequent items would fail to have the necessary support.

2.4. Our Association Rule Mining Algorithm

We have designed an *Adaptive–Support Association Rule Mining (ASARM)* algorithm to solve the above problem. It consists of two parts: an outer loop ASARM1 and an inner loop ASARM2. A high level description of our algorithm is:

- Main process (ASARM1): Given a minimum confidence constraint:
  - it chooses an initial minimum support according to the like ratio of the target user/article and calls ASARM2;
  - while the number of rules that satisfy the minimum support and confidence constraints is not in the given range, it adjusts the minimum support and call the mining process again.
- Mining process (ASARM2): Given minimum confidence and support constraints:

- ASARM2 mines rules for the target user/article satisfying those constraints. If fewer than the required minimum number of rules are found, it returns those rules and raises *belowMinNumrulesFlag*. In contrast, if the mining process reaches the required maximum number of rules plus one, ASARM2 halts and returns the first maxNumrules rules mined and raises *aboveMaxNumrulesFlag*.

2.4.1. *ASARM1*

In order to mine only a given number of most promising rules for each target item, we use ASARM1 to control the minimum support count and find the rules with the highest supports. The minimum support count is the minimum number of transactions that must satisfy a rule in order to make that rule frequent, i.e., it is the multiplication of the minimum support and the whole number of transactions. The overall process is shown in detail in Figure 1.

Input: Transactions *T, targetItem, minConfidence, minNumrules, maxNumrules.*

Output: Set S of association rules for *targetItem* such that number of rules in S is in [*minNumrules, maxNumrules*] (unless fewer than minNumrules exist for the given minimum confidence), confidence of rules in S are greater than or equal to *minConfidence*, and no rule outside S with confidence greater than or equal to the *minConfidence* has a higher support than a rule in S.

```
Set initial minsupportCount based on targetItem's frequency;
R = ASARM2(T, targetItem, minConfidence, minNumrules, maxNumrules, minsupportCount);
while (R.aboveMaxNumrulesFlag) or (R.belowMinNumrulesFlag) do {
    if (R.aboveMaxNumrulesFlag) then {
        if minsupportCount = |T| then return (R.rules);
        minsupportCount++;
        R1 = R;
        R = ASARM2(T, targetItem, minConfidence, minNumrules, maxNumrules, minsupportCount);
        if (R.belowMinNumrulesFlag) then
            return(maxNumrules rules with the highest support from R.rules ∪ R1.rules);
    }
    else (i.e. if (R.belowMinNumrulesFlag)) {
        if minsupportCount = 0 then return (R.rules);
        minsupportCount- -;
        R1 = R;
        R = ASARM2(T, targetItem, minConfidence, minNumrules, maxNumrules, minsupportCount);
        if (R.aboveMaxNumrulesFlag) then
            return(maxNumrules rules with the highest support from R.rules ∪ R1.rules);
    }
}
return(R.rules);
```

Figure 1. The ASARM1 Algorithm

The ASARM1 algorithm works as follows:

1. First, ASARM1 initializes the minimum support count according to the target item's like ratio, and calls ASARM2 to mine rules. The *like ratio* of a user is the percentage of articles that the user likes among those that he/she has rated. Similarly, the like ratio of an article is the percentage of users who have liked the article among those who have rated it.

2. If the number of rules mined by ASARM2 exceeds maxNumrules (this would be signaled by a raised *aboveMaxNumrulesFlag*) which implies that the minimum support count is too low, ASARM1 increases the minimum support count and calls ASARM2 again. If the number of rules mined by ASARM2 is below minNumrules (this would be signaled by a raised *belowMinNumrulesFlag*) due to the minimum support count being too high, ASARM1 decreases the minimum support count and calls ASARM2 again. Finally, if the number of rules is within the target range or the minimum support count cannot be changed further to obtain a rule count within the desired range, ASARM1 halts and returns the set of mined rules.

2.4.2. *ASARM2*

ASARM2 is a variant of CBA-RG (Liu et al., 1998) and therefore of the Apriori algorithm (Agrawal and Srikant, 1994) as well. It is worth mentioning that the Apriori algorithm uses a two-stage mining process. In the first stage, frequent itemsets (i.e., sets of items having support above the user-specified threshold) are found. The second stage then generates rules from frequent itemsets, keeping only the rules that satisfy the minimum confidence constraint. In contrast, our ASARM2 algorithm generates frequent itemsets and rules simultaneously.

ASARM2 is a variant of CBA-RG in the sense that instead of mining rules for all target classes, it only mines rules for one target item. It differs from CBA-RG in that it will only mine a number of rules within a certain range. If it detects that it has generated *maxNumrules + 1* rules already then it simply terminates its execution and returns the first *maxNumrules* rules it has mined.

We follow the notation of (Liu et al., 1998) to describe our algorithm. *k-condset* denotes a set of items of size k which could form a rule: *k-condset* $\Rightarrow$ *targetItem*. The support count of the *k-condset* (called *condsupCount*) is the number of transactions that contain the *k-condset*. The support count of the corresponding rule (also called *rulesupCount* of this *k-condset*) is the number of transactions that contain the *condset* as well as the target item.

Association rules are generated by making multiple passes over the transaction data. The first pass counts the *rulesupCounts* and the *condsupCounts* of all the single items and finds the frequent 1-*condsets*. For pass $k > 1$, it generates the candidate frequent *k-condsets* by using the frequent $(k-1)-$*condsets*; then it scans all transactions to count the *rulesupCounts* and the *condsupCounts* of all the candidate *k-condsets*; finally, it goes over all candidate *k-condsets*, selecting as *frequent* those whose *rulesup* is above the minimum support, and at the same time generating rules *k-condset* $\Rightarrow$ *targetItem*, if the confidence of the rule is above the minimum confidence. The algorithm is presented in Figure 2.

2.5. Algorithm Implementation

We have implemented our algorithm in C++. In order to speed up the mining process, we have chosen data structures that efficiently support the key operations of our algorithm: (1) subset test: how to find all candidate *condsets* that are contained in one transaction; (2) candidate generation - join step: how to find frequent *condsets* that could be joined together; (3) candidate generation - prune step: how to test if any (k-1)-subset of a candidate *k-condset* is a frequent (k-1)-*condset*. As described in (Agrawal and Srikant, 1994), using a hash-tree to store candidate itemsets and a bitmap to store a transaction could speed up the support counting process. In addition to using bitmaps and hash–trees, we use a data structure that we call *set–tree*, which we have designed and implemented to facilitate the join and prune operations on candidate itemsets.

Input: Transactions *T, targetItem, minConfidence, minNumrules, maxNumrules, minsupportCount.*

Output: Set S of association rules for *targetItem* s.t. rules in S satisfy the *minConfidence*, and the *minsupportCount* constraints, number of rules in S is at most *maxNumrules*, *S.aboveMaxRulenumFlag* (resp. *S.belowMinRulenumFlag*) is raised if more (resp. less) than *maxNumrules* (resp. *minNumrules*) exist for the given *minConfidence* and *minsupportCount*.

```
F1 = { frequent 1–condsets } ;
R = genRules(F1, targetItem); /* generates at most maxNumrules rules, and
                                 sets R.aboveMaxRulenumFlag if more rules exist */
for (k = 2; (F_{k-1} ≠ ∅) and (not R.aboveMaxRulenumFlag); k++) do {
    C_k = candidateGEN(F_{k-1});
    for each transaction t ∈ T do {
        C_t = all candidate condsets in C_k contained in t;
        for each candidate c ∈ C_t do {
            c.condsupCount++;
            if t contains targetItem then c.rulesupCount++;
        }
    }
    F_k = {c ∈ C_k | c.rulesupCount ≥ minsupportCount};
    R1 = genRules(F_k, targetItem);
    if (R.numrules + R1.numrules > maxNumrules) or (R1.aboveMaxRulenumFlag) then
        Set R.aboveMaxRulenumFlag;
    R = maxNumrules rules with the highest support from R.rules ∪ R1.rules;
}
if (R.numrules < minNumrules) then set R.belowMinNumrulesFlag;
return(R);
```

Figure 2. The ASARM2 Algorithm

3. Collaborative Recommendation using our Mining Algorithm

We now describe how our algorithm to mine association rules may be used for collaborative recommendation. The same mining process may be used to mine a certain number of rules for each user/article for both user associations and article associations. The main differences between the implementation of these two association types are that we use different transaction data to mine the association rules, and we use quite different recommendation strategies for the two association types.

A desirable feature of recommender systems is short response time. Our algorithm achieves a reduced response time for the following reasons:

- We mine rules off-line for article associations;

- The training data to mine rules for one target user is only a small subset of all the ratings, namely the ratings from users in the collaborative group and the target user for articles that the target user has rated. So the training data size is small;

- The mining process AR-CRS-2 will stop after it mines *maxRulenum* rules. If *maxRulenum* is small, it is very fast;
- In the main process, we choose an initial minimum support count according to users' like ratios. For most users, the main process only needs to call the mining process two or three times. We can switch to article associations for users who need more iterations.

3.1. Mapping Ratings to Transactions

Association rules are mined from a set of "transactions". For collaborative recommendation, we assume that we are given users' ratings for certain articles [1]. The conversion from item ratings available for recommendation tasks to "transactions" as required for association rule mining is determined by what kind of associations and how many levels of associations we want to discover. We map the numeric ratings for an item into two categories: *like* and *dislike* according to whether the rating for the item is greater than or less than some chosen threshold value. Then we convert the chosen *like* and *dislike* ratings into transactions:

- In order to obtain *like* associations among users, we have each user correspond to an "item" and each article rated by users correspond to a "transaction". If a user likes an article, then the transaction corresponding to the article contains the item corresponding to the user liking the article; If the user dislikes or did not rate the article, then the corresponding transaction does not contain the corresponding item. The mined rules will then be of the following form: "90% of articles liked by user A and user B are also liked by user C, 30% of all articles are liked by all of them", or, in simpler notation,

  "$[user_a : like]$ AND $[user_b : like] \Rightarrow [user_c : like]$ with confidence 90% and support 30%".

- In order to mine *like* and *dislike* associations among users, we extend each user, say $user_k$, to two "items", one item corresponding to [$user_k$: like] and another item corresponding to [$user_k$: dislike]. If $user_k$ likes an article, then the corresponding transaction contains the item [$user_k$: like]; If $user_k$ dislikes the article, the corresponding transaction contains the item [$user_k$: dislike]; If $user_k$ did not rate the article, the corresponding transaction does not contain either of these two items. From here, we can mine rules like "90% of articles liked by user A and disliked by user B are liked by user C, 30% of all articles are liked by user A and C and disliked by user B", or simply

  "$[user_a : like]$ AND $[user_b : dislike] \Rightarrow [user_c : like]$ with confidence 90% and support 30%".

- In order to mine *like* associations among *articles*, we proceed by analogy with the case of user associations described above. Namely, we have each article correspond to an "item" and each user who rated the target item correspond to a "transaction". If a user likes an article, then the transaction corresponding to the user contains the item corresponding to the article; If the user dislikes or didn't rate the article, then the corresponding transaction does not contain the corresponding item. From here, we can mine *like* associations among articles. For example, Table II gives an example of the training data to mine *like* article associations for a target article. In the table, [A1: L] means $[article_1 : like]$, and the first column corresponds to the user ID. So from the transaction data we could mine rules like:

  "$[article_1 : like]$ AND $[article_4 : like] \Rightarrow [target\_article : like]$" with confidence 100% and support 40%.

[1] techniques that infer ratings from passive user feedback may first be applied if explicit ratings are not available.

Table II. Training Data for *Like* Article Associations

User ID	[A1: L]	[A2: L]	[A3: L]	[A4: L]	[A5: L]	[A6: L]	[Target_A: L]
1	1	0	0	0	1	0	0
2	0	1	0	0	1	0	1
3	1	0	0	1	1	0	1
4	0	1	1	1	0	1	0
5	1	0	0	1	0	0	1

3.2. Recommendation Strategy

3.2.1. *User associations*

In the case of user associations, the rules mined are akin to $[user_1 : like]$ AND $[user_2 : dislike]$ $\Rightarrow [target\_user : like]$. For a test article of the target user, if $user_1$ likes this article and $user_2$ dislikes this article, then we say this rule *fires* for this article. We associate each rule with a score, which is the product of the support and the confidence of the rule. We use this product because the support and the confidence measure the rule's quality, which will directly influence the accuracy of the recommendations made by this rule. We also assign a score to each article, which is the sum of the scores of all the rules that fire for that article.

$$score_{article_i} = \sum_{\text{all rules that recommend article}_i} (support_{rule} * confidence_{rule}).$$

If $score_{article_i}$ is greater than a threshold, then we recommend $article_i$ to the target user. Obviously, the score threshold determines the number of articles that are recommended to a user, and so it is of great importance to select it wisely. We have done so based on extensive experimentation with our system.

3.2.2. *Article associations*

For article associations, the rules we have are of the form: $[article_1 : like]$ AND $[article_2 : like]$ AND $[article_3 : like] \Rightarrow [target\_article : like]$. For a test article of the target user, if the user likes $article_1$, $article_2$, and $article_3$ (which could be determined from the training articles of the user), then we say this rule fires for the target article.

Our recommendation strategy for article associations is different than for user associations. We only recommend articles whose rules' supports are above a support cutoff. After we have determined what value of the support cutoff is best during the system tuning process, we can restrict the mining process to rules whose support is above the support cutoff. This may seem similar to Apriori or CBA in the sense that a minimum support is specified in the mining process, but the difference is that we only mine rules with the highest possible support for one article at a time and the number of rules obtained lies within the specified range. Our mining process has the following advantages:

- By mining article associations for one article at a time, only ratings related to the target article are used for mining, which is often only a small subset of the whole rating data. The support of a rule is calculated over the small subset of the whole rating data, which enables us to obtain rules for articles that have only received a limited number of ratings, for example a new movie. This would not be possible if we mined article associations for all articles at once, because rules for new articles would fail to have the necessary support over the whole rating data.

– A significant amount of runtime is saved by mining rules only over the subset of the rating data that is related to the target article instead of over the whole data. Systems that mine rules with unrestricted heads such as IBM's Intelligent Miner can easily take several days to mine article associations for all articles at once.

3.2.3. *Combined associations mode*

In our experiments (described in the next section of the paper), we found that when a target user's minimum support as determined by the mining process is very low, it takes a long time to mine rules for this user and at the same time the performance of using those rules for recommendation is poor. So we use the following strategy to combine user and article associations: If a user's minimum support is greater than a threshold, then we use user associations for recommendation, otherwise we use article associations for recommendation.

4. Experimental Evaluation

In this section, we describe experimental results obtained with a collaborative recommendation method based on our association rule mining techniques. We include results for user associations, article associations, and a combination of the two.

4.1. Training and Test Data

We use the EachMovie Dataset as the test-bed of our approaches. The EachMovie data set is an online data source provided by Compaq's Systems Research Center (McJones, 1997). It contains ratings from 72,916 users for 1,628 movies. User ratings were recorded on a numeric six–point scale (0.0, 0.2, 0.4, 0.6, 0.8, 1.0).

4.2. Collaborative Users and Target Users

We performed three groups of experiments. In the first group of experiments, we chose the first 1000 users in the EachMovie dataset who have rated more than 100 movies as collaborative users, and the first 100 users whose userIDs are greater than 70,000 and who have rated more than 100 movies as target users. By choosing collaborative users who have rated over 100 movies, we ensure that enough movies are available for our cross-validation tests. In order to compare our approach with other approaches, we performed two additional groups of experiments for which we chose the first 2000 users in the database as the collaborative user group. In one case we chose as target users 91 random users whose like ratios are less than 0.75 and who have rated 50 to 100 movies. For the final tests, 20 target users were selected at random from the set of users with like ratios less than 0.75.

4.3. Experimental Protocol

4.3.1. *Parameters*

The main parameters of our system that were adjusted during the experiments reported in this paper are the following:

– the minimum confidence of the association rules;

– the *like* and *dislike* threshold for the ratings;

– the maximum number of terms per rule; minRulenum and maxRulenum;

- the target range for the number of rules to be mined;
- the score threshold for recommendation.

Results for different parameter values are described in the next section. We use a *like* threshold of 0.7, i.e., if a user's rating for an article is greater than 0.7, then we assume that the user likes the article. With this *like* threshold, the ratio of the number of movies liked to the total number of movies rated among all the test users is 0.45.

4.3.2. *4-fold cross-validation*

During our tests, for each given target user we employ the *4-fold* cross-validation approach. First, we randomly divide all the articles this user has rated into 4 groups. Then we run four rounds of tests, each time choosing one group of articles as test data and the other three groups as training data. So every article this user rated will be used as a test article once.

4.4. Performance Measurement

We use *accuracy*, a commonly used performance measure in machine learning, together with two standard information retrieval measures, *precision* and *recall*. Accuracy is the percentage of correctly classified articles among all those classified by the system; Precision is the percentage of articles recommended to a user that the user likes; Recall is the percentage of articles liked by a user that are recommended to him/her. More precisely,

$$accuracy = \frac{correctly\_classified\_articles}{total\_articles\_classified}$$

$$precision = \frac{correctly\_recommended\_articles}{total\_recommended\_articles}$$

$$recall = \frac{correctly\_recommended\_articles}{total\_articles\_liked\_by\_users}$$

For recommendation tasks, precision is perhaps most significant because we are more concerned about making high quality recommendations than about recommending a large number of items. Nonetheless, the choice of a performance measure may depend on the application domain. Our approach includes several adjustable parameters that may be tuned to optimize a particular measure.

4.5. Performance Results

4.5.1. *Maximum Rule Length*

We use rule length to refer to the number of the items present in the body of a rule. Table III lists the performance for different maximum rule lengths. Here we choose minimum confidence as 100%, and the number of rules in the range [5,100].

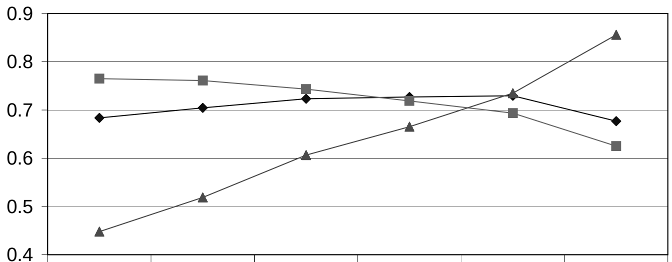
Table III. Performance for Different Maximum Rule Length

Rule Length	2	4	6	8	10
Accuracy	0.693712	0.696117	0.695676	0.69759	0.694547
Precision	0.704357	0.724006	0.733482	0.737896	0.736086
Recall	0.572606	0.545425	0.528625	0.528411	0.520813

From Table III we could see that when the maximum rule length is around 8, we get the best performance. Given these results and the fact that longer rules are in more danger of overfitting the data, we have chosen the maximum rule length to be 8 for all the remaining experiments.

4.5.2. *Minimum Confidence*

We believe that the minimum support and minimum confidence of the mined association rules are the two most important factors in determining recommendation performance. The minimum support is automatically adjusted for each user by our algorithm during the mining process. We evaluated the recommendation performance as a function of the minimum confidence. The results are shown in Figure 3 and are summarized below.



	1	0.95	0.9	0.85	0.8	0.75
Accuracy	0.683552	0.704511	0.723359	0.727041	0.729348	0.676991
Precision	0.764803	0.76103	0.743149	0.718678	0.693465	0.625
Recall	0.447833	0.518673	0.606528	0.665383	0.734724	0.855856

Minimum Confidence

Figure 3. Performance for Different Minimum Confidence Values

- The minimum confidence has a significant impact on the performance: the higher the minimum confidence, the higher the precision but the lower the recall. We achieved the highest precision of 0.76 with a recall of 0.45 for a minimum confidence of 100%.
- When the minimum confidence is varied, a tradeoff between the precision and the recall becomes evident. The highest *accuracy* is obtained not with minimum confidence 100% but with minimum confidence from 80% to 90%.

We use a minimum confidence of 0.9 for the remaining experiments.

4.5.3. *Range for the Number of Rules*

In order to decide what is the appropriate range of number of rules, we ran experiments with different ranges. As shown in Figure 4, we achieve quite similar performance for ranges [10,100], [20,200], and [50,500]. Our experiments verify that a limited number of rules is desirable for making recommendations to a user. Too many rules do not improve recommendation performance and lead to an unnecessarily large run time.

4.5.4. *Like and Dislike Associations*

In order to obtain both *like* and *dislike* associations, we map ratings into *like* and *dislike* by using two thresholds: the *like* threshold and the *dislike* threshold. The *like* associations we discussed before correspond to choosing the *dislike* threshold as 0. Figure IV gives the comparison of the performance for different *dislike* thresholds.

There is no significant difference between the performance for different *dislike* thresholds. So employing *like* and *dislike* associations does not outperform employing *like* associations only.

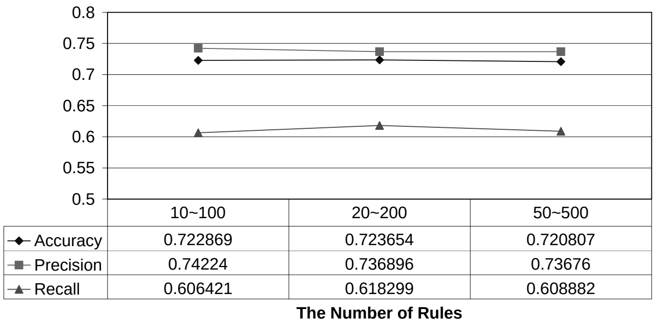


Figure 4. Performance for Different Rule Set Sizes

Table IV. Performance for Like and Dislike Associations. like_threshold = 0.7

Dislike Threshold	0	0.3	0.7
Accuracy	0.721936	0.724341	0.718549
Precision	0.725214	0.726137	0.717452
Recall	0.634029	0.640663	0.637453

4.5.5. *Score Threshold*

The score threshold for recommendation is also an important parameter of our approach. We have decided to use a threshold that is a linear function of the number of rules. The base value and the slope of this linear function must be selected; Figure 5 and Figure 6 give the performance for different values of these subparameters of the score threshold.

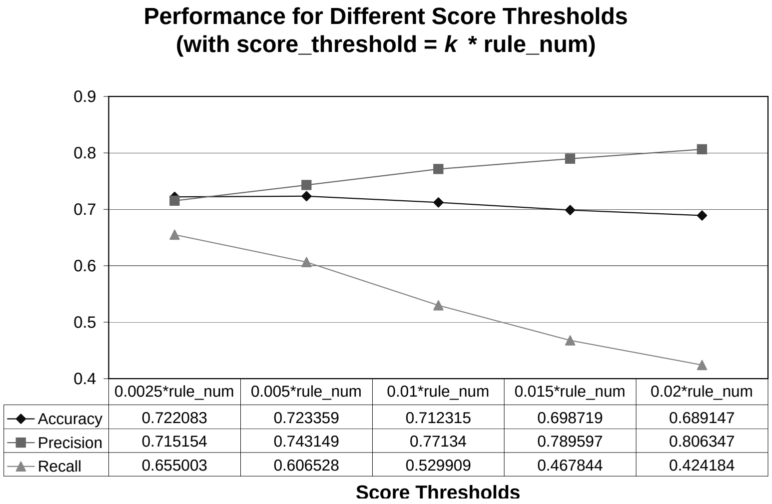
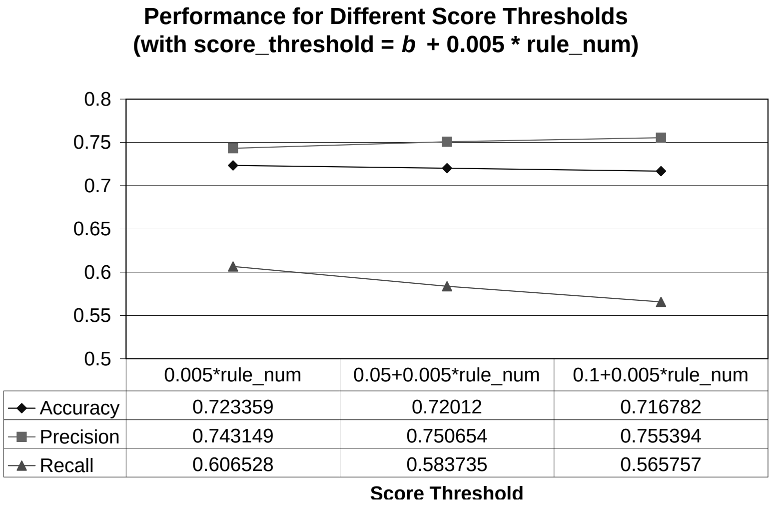


Figure 5. Performance for Different Score Thresholds I

From these two figures, we observe that the score threshold has a similar impact on the performance as the minimum confidence, i.e., the higher the score threshold, the higher the precision but the lower the recall. However, by simultaneously adjusting the minimum confidence we can



	0.005*rule_num	0.05+0.005*rule_num	0.1+0.005*rule_num
Accuracy	0.723359	0.72012	0.716782
Precision	0.743149	0.750654	0.755394
Recall	0.606528	0.583735	0.565757

Figure 6. Performance for Different Score Thresholds II

increase both precision and recall: in Figure 5, we achieve a precision of 0.77 and a recall of 0.53 for a score threshold of $0.01 * num\_rules$ (and minimum confidence of 90%); c.f. the results for minimum confidence 100% and score threshold 0.005 (Section 4.5.2).

4.5.6. *Performance Distribution*

Figure 7 presents the distribution of precision and recall over the set of test users for a score threshold of $0.02 * num\_rules$, and Figure 8 presents the distribution for a score threshold of $0.005 * num\_rules$. Notice that for a score threshold of $0.02 * num\_rules$, some users receive no recommendations. Some tuning is required to find a suitable threshold value.

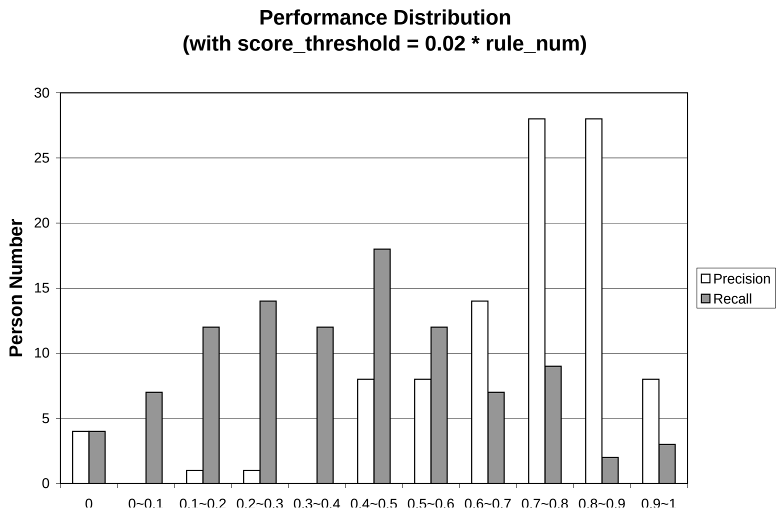


Figure 7. Performance Distribution for Score Threshold = 0.02*num_rule

4.5.7. *Different Like Ratios*

It is easier to do recommendation if a user's prior probability of liking an article is high. A system that recommends all articles to such a user will achieve good results. To understand how this phenomenon affects performance, we computed the average precision of users with different ranges

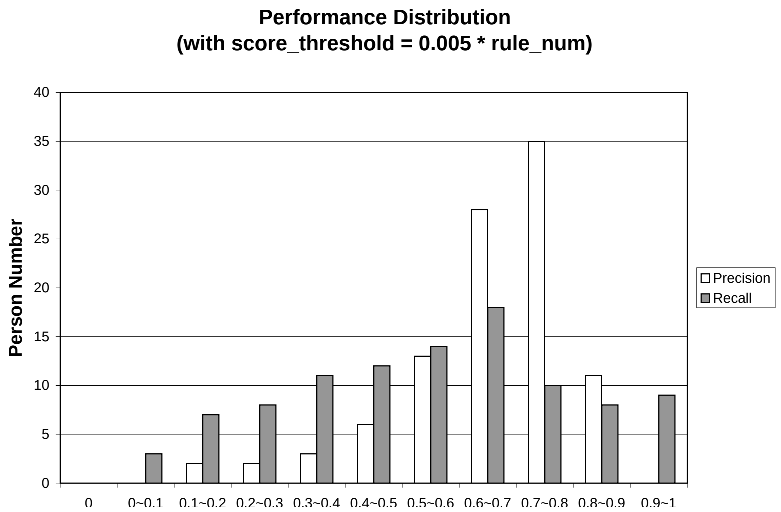


Figure 8. Performance Distribution for Score Threshold = 0.005*num_rule

of like ratios, which reflect a user's prior probability of liking a movie. The results, shown in Figure 9, confirm that precision is an increasing function of the like ratio. However, note also

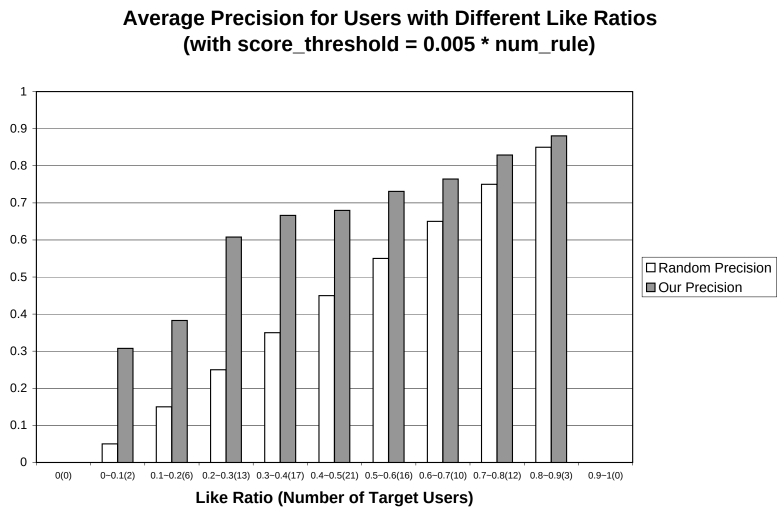


Figure 9. Precision Distribution as a function of Like Ratio

that our recommendations are always better than random recommendations, whose probability of success is equal to the user's prior probability of liking an article.

4.5.8. *Article Associations*

Performance for article associations is shown in Figure 10. As the figure shows, performance degrades slightly relative to the user associations mode. However, the running time is improved considerably. Running times shown are in seconds on a 463 MHz Pentium family PC with 128 MBytes of RAM. We point out that running times will be significantly lower than those shown if the size of the training set is reduced. For example, for a training set containing an average of 53 training movies per user, the running time for user associations (with a collaborative user group of 2000 users) is $0.55s$ instead of the value $14.2s$ reported in the table.

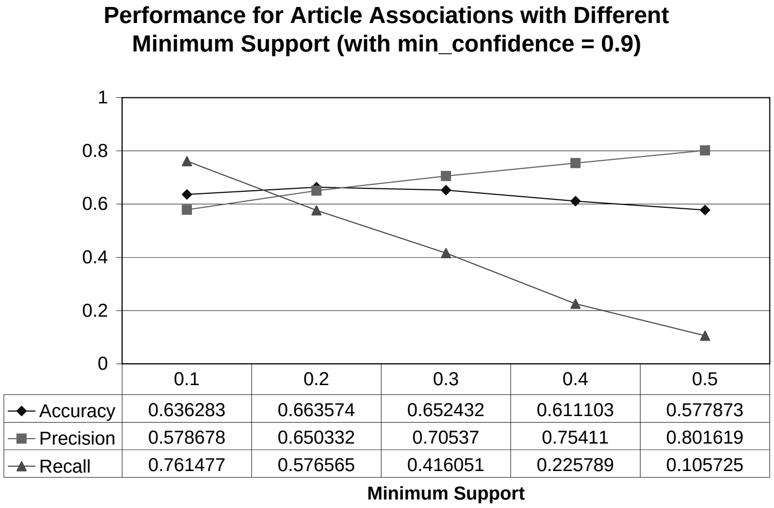


Figure 10. Performance for Article Associations

4.5.9. *Combining User and Article Associations*

Table V lists the performance for user associations, article associations, and the combination of the two as described in the previous section of the paper. We can see that when the two types of associations are combined, performance degrades a little bit, but we achieve a much faster response time.

Table V. Combining User and Article Associations

	User Assoc	Article Assoc	Combined	Combined
Threshold	N/A	N/A	0.075	0.1
Accuracy	0.720	0.611	0.717	0.712
Precision	0.751	0.754	0.745	0.723
Recall	0.584	0.226	0.582	0.602
Avg. Runtime	14.2s	0.06s	5.2s	4.6s

4.6. Comparison with other systems

Billsus and Pazzani (Billsus and Pazzani, 1998) tested three collaborative recommendation techniques on the EachMovie dataset: a correlation-based method, neural networks paired with Information Gain, and neural networks paired with Singular Value Decomposition. In their experiments, they choose the first 2000 users as the collaborative users, another 20 random users whose like ratios are less than 0.75 as target users, and 50 random movies as training movies for each target user. The resulting accuracies are listed in Table VI.

In order to compare our approach with those approaches, we tested our approach under similar experimental conditions. For the collaborative user group, we tried both the first 1000 users who have rated more than 100 movies, as well as the first 2000 users. We chose 91 random users who have rated 50 to 100 movies as target users and whose like ratios are below 0.75. We employed the *4-fold* cross-validation approach for our tests, which results in that the average number of training

Table VI. Accuracy for Four Collaborative Approaches

	Correlation	InfoGain/ANN	SVD/ANN	Our Approach
Accuracy	0.644	0.67	0.679	0.682

movies for each user is 53. The accuracy achieved for the user associations mode of our system varied between 0.67 and 0.69 depending on what sets of training users were selected (as well as the values of system parameters). In all cases, the performance of our approach was superior to that of the correlation-based method tested in (Billsus and Pazzani, 1998). With a randomly chosen set of 20 target users whose like ratios are less than 0.75, our approach achieved an accuracy of 0.682, slightly higher than the highest accuracy reported in (Billsus and Pazzani, 1998) under essentially the same conditions, as shown in Table VI.

5. Conclusions

We have described a new collaborative recommendation technique based on a specialized algorithm for mining association rules of classification type. Unlike most existing association rule mining algorithms, which require that the minimum support of the rules to be mined be specified in advance, our mining algorithm adjusts the minimum support during the mining process so that the number of rules generated lies within a specified range. This reduces the running time and provides enough rules for good recommendation performance.

Some salient observations may be summarized as follows:

- Our experiments verify our assumption that a limited number of rules is desirable for making recommendations to a user. Mining a large number of rules does not improve recommendation performance and unnecessarily increases the running time.

- The minimum confidence of rules has a great impact on recommendation quality. This is not surprising because the confidence of a rule corresponds to the average precision of using this rule to recommend training articles to the user.

- We can achieve a fast response time while maintaining good recommendation quality by combining user associations and article associations.

- The recommendation performance obtained through our approach is significantly better than that of traditional correlation-based methods.

References

Agarwal, R. C., C. C. Aggarwal, and V. Prasad: 2000, 'Depth First Generation of Long Patterns'. In: *Proc. of the Sixth ACM SIGKDD Conference on Knowledge Discover y and Data Mining*. Boston, MA, pp. 108–118.

Agrawal, R., T. Imielinski, and A. Swami: 1993, 'Mining Association Rules between Sets of Items in Large Databases'. In: *Proc. of the ACM SIGMOD Conference on Management of Data*. Washington, D.C., pp. 207–216.

Agrawal, R. and R. Srikant: 1994, 'Fast Algorithms for Mining Association Rules'. In: *Proc. of the 20th VLDB Conference*. Santiago, Chile, pp. 487–499.

Balabanovic, M. and Y. Shoham: 1997, 'Fab: Content-Based, Collaborative Recommendation'. *Communications of the ACM* **40**(3), 66–72.

Billsus, D. and M. J. Pazzani: 1998, 'Learning Collaborative Information Filters'. In: *Proc. of the Fifteenth International Conference on Machine Learning*. Madison, Wisconsin, Morgan Kaufmann Publishers.

Breese, J., D. Heckerman, and C. Kadie: 1998, 'Empirical Analysis of Predictive Algorithms for Collaborative Filtering'. In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. Madison, WI.

Brin, S., R. Motwani, J. D. Ullman, and S. Tsur: 1997, 'Dynamic Itemset Counting and Implication Rules for Market Basket Data'. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. New York, pp. 255–264, ACM Press.

Chen, M.-S., J. S. Park, and P. S. Yu: 1998, 'Efficient Data Mining for Path Traversal Patterns'. *IEEE Trans. on Knowledge and Data Engineering* **10**(2), 209–221.

Cooley, R., B. Mobasher, and J. Srivastava: 1997a, 'Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns'. Technical Report TR 97-021, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, USA.

Cooley, R., J. Srivastava, and B. Mobasher: 1997b, 'Web Mining: Information and Pattern Discovery on the World Wide Web'. In: *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*.

Cooley, R., P.-N. Tan, and J. Srivastava: 1999, 'WebSIFT: The Web Site Information Filter System'. In: *Proc. of the Workshop on Web Usage Analysis and User Pr ofiling (WebKDD99)*. Available at http://www.acm.org/sigkdd/proceedings/webk dd99/.

Fu, X., J. Budzik, and K. J. Hammond: 2000, 'Mining navigation history for recommendation'. In: *Proceedings of the 2000 international conference on intelligent user interfaces*. New Orleans, LA, pp. 106–112.

Hájek, P., I. Havel, and M. Chytil: 1966, 'The GUHA Method of Automatic Hypotheses Determination'. *Computing* **1**, 293–308.

Hájek, P. and T. Havranek: 1977, 'On Generation of Inductive Hypotheses'. *Int. J. Man-Machine Studies* **9**, 415–438.

Liu, B., W. Hsu, and Y. Ma: 1998, 'Integrating Classification and Association Rule Mining'. In: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*. New York, pp. 80–86.

McJones, P.: 1997, 'EachMovie collaborative filtering data set'. http://www.research.compaq.com/SRC/eachmovie. Compaq Systems Research Center.

Resnick, P., N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl: 1994, 'GroupLens: an Open Architecture for Collaborative Filtering of Netnews'. In: *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW94)*. pp. 175–186.

Sarwar, B.: 2001, 'Sparsity, Scalability, and Distribution in Recommender Systems'. Ph.D. thesis, University of Minnesota.

Shardanand, U. and P. Maes: 1995, 'Social Information Filtering: Algorithms for Automating "Word of Mouth"'. In: *Proceedings of the Conference on Human Factors in Computing Systems (CHI95)*. pp. 210–217.

Srikant, R. and R. Agrawal: 1996, 'Mining Quantitative Association Rules in Large Relational Tables'. In: *Proc. of the 1996 ACM SIGMOD Conference on Management of Data*. Montreal, Canada.

Webb, G. I.: 2000, 'Efficient Search for Association Rules'. In: *Proc. of the Sixth ACM SIGKDD Conference on Knowledge Discover y and Data Mining*. Boston, MA, pp. 99–107.