World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:9, No:1, 2015

A Hybrid Recommendation System Based On Association Rules

Ahmed Mohammed K. Alsalama

International Science Index, Computer and Information Engineering Vol:9, No:1, 2015 waset.org/Publication/10000147

***Abstract*—Recommendation systems are widely used in e-commerce applications. The engine of a current recommendation system recommends items to a particular user based on user preferences and previous high ratings. Various recommendation schemes such as collaborative filtering and content-based approaches are used to build a recommendation system. Most of current recommendation systems were developed to fit a certain domain such as books, articles, and movies. We propose[1] a hybrid framework recommendation system to be applied on two dimensional spaces $(User \times Item)$ with a large number of Users and a small number of Items. Moreover, our proposed framework makes use of both favorite and non-favorite items of a particular user. The proposed framework is built upon the integration of association rules mining and the content-based approach. The results of experiments show that our proposed framework can provide accurate recommendations to users.

***Keywords*—Data Mining, Association Rules, Recommendation Systems, Hybrid Systems.

I. Introduction

IN places such as Amazon and Netflix, analyzing the feedback data like ratings provides useful information for those companies and their customers at the same time. For example, Netflix analyzes the movie ratings of customers in certain ways to recommend other movies [1]. Also, Amazon can study a customer's profile and analyze the feedback that the customer provides, to recommend books and other items to him or her. All of these kinds of recommendations are done by what is called recommendation systems. The goal of recommendation systems is to suggest items to a particular user. A user and an item are the basic entities that appear in a recommendation system. A user is a person who utilizes the recommender system providing her/ his opinion (i.e., rating) about various items. Then, a user receives recommendations about new items from the system based on her/his opinion [2]. The task of recommendation systems is to predict the ratings of the items that the user has not seen or ranked before [3]. Based on that predicated rating, the recommendation system will be able to recommend other items to the user [3]. There are different approaches to recommendation systems that are used to serve in different contexts based on system needs [4]. Our work is to combine association rules mining and content-based approach to provide a framework of a hybrid recommendation system on two dimensional spaces $(User \times Item)$.

Ahmed M. Alsalama is with the Computer Center of Exploration and Petroleum Engineering Center, Saudi Aramco Company, Dhahran, Saudi Arabia e-mail: (ahmed.alsalama@aramco.com).

[1]The research findings in this paper were originally from the author master's degree thesis[23].

A. Motivations

The Apriori algorithm requires scanning the database every time it generates the candidate itemsets to build the association rules [5]. This issue affects the performance of the Apriori algorithm and causes scalability problems, especially when the transactions in the database are large in number. In e-commerce applications of recommendation systems, the size of the $(User \times Item)$ matrix is big, and in most commercial recommendation systems this matrix suffers from the sparsity problem which means there is a substantial number of items in the systems have not yet been rated. Thus, applying the Apriori algorithm to a sparse matrix can lead to irrelevant information and can cause poor recommendations to the target user. Our proposed hybrid framework is to apply the Apriori algorithm on two dimensional spaces $(User \times Item)$ with a large number of users and a small number of items. Moreover, most current recommendation systems consider the items that have been rated highly by the users and recommend similar items to the target user. The current recommendation systems focus on items that are liked by the user and, most of the time, discard the items that the user did not like. The assumption in our work here is: Even if a user did not like an action movie that she/ he watched, our system may still recommend another action movie to the user.

B. Our Approach

Our approach is divided into the following steps:

- The first step is to apply the Apriori algorithm to a $(User \times Item)$ matrix to generate the association rules.
- The second step is to divide the items that have been rated by users into two categories: *Favorite Items* and *Non-Favorite Items*.
- The third step is to use the generated association rules to discover the frequent itemsets of *Favorite Items* set and find the correlations among those items to recommend new items to a target user.
- The last step is to apply the content based approach into *Non-Favorite Items* set to recommend some new items to a target user.

II. Background

A. Mining Frequent Patterns and Association Rules

Frequent patterns can be defined as patterns that appear frequently in the data set [5]. A set of items such as bread, butter, and milk that occur frequently together in a transaction is called frequent itemset [5]. Mining in a frequent itemset allows us to discover the associations and correlations among items in large transactional data sets [5]. In general, finding all frequent itemsets and generate strong association rules are the main process of association rule mining [5].

B. The Apriori Algorithm

The Apriori algorithm is a well-known algorithm that is used for mining frequent itemsets for Boolean association rules [5]. It is an algorithm for efficient association rule discovery [6]. The algorithm was proposed by R. Agrawal and R. Srikant in 1994 [5]. The approach that is used in the the Apriori algorithm is known as a level-wise search, where *k-itemsets* are used to explore *(k+1)-itemsets* [5].

C. Recommendation Systems Approaches

This section reviews the basic concepts and approaches of recommendation systems. The approaches include content-based, collaborative filtering, demographic, and hybrid approaches. The limitations of the current recommendation approaches are also described in this section.

1) Content-Based Approaches: In content-based recommendation systems, the user rates the items, and the recommender system should understand the common characteristics among the items that the user has rated in the past [3]. The system then recommends the items that have a high degree of similarity to the user's preferences and tastes [3]. For example, in a movie recommendation system, a content-based approach tries to understand the common characteristics such as actors, directors, genres, etc among the movies that the user has given high ratings in the past [3]. Then, the system recommends the movies that have a high degree of similarity to the user's preferences [3]. A content-based approach usually deals with item profile $ItemProfile(i)$ and user profile $UserProfile(u)$ [3].

2) Collaborative Filtering Approaches: Collaborative filtering approaches are widely used in e-commerce [7]. They have been successful in many e-commerce applications such as Amazon and Netflix [20]. It is a popular technique used to reduce information overload [20]. Amazon recommends books to their customers using the collaborative filtering approach [3]. A recommendation system based on collaborative filtering recommends items to a particular user based on the similar items that have been rated by some other users [3]. The system finds items for other users that have similar preferences as a query user [3]. For example, in movie recommendation systems that are based on the collaborative filtering approach, the system finds a group of users that have similar preferences as a query user. Then, the system recommends the movies that they have rated highly in the past by those users to the target user [3].

Collaborative filtering approaches are grouped into two general categories [3]:

- Memory-Based Approaches: They use the entire collection of the rated items to make recommendations or predictions [3].
- Model-Based Approaches: They allow systems to learn to recognize patterns in the data sets to make recommendations or predictions [9].

3) Demographic Based Approach: A demographic-based recommender system recommends items to the user based on the user's demographic information such as gender, age, and date of birth [10]. The demographic approach puts the users into groups based on their demographic characteristics [10].

4) Hybrid Approach: The hybrid approach has been introduced to avoid the limitations of the content-based and collaborative filtering approaches [3]. Several recommendation systems combine two or more approaches to gain better performance and eliminate some of the drawbacks of the pure recommendation systems approaches [11]. Currently, many recommendation systems combine the collaborative approach with some other approaches such as content-based approach and demographic approach [12]. Combining collaborative filtering and content-based approaches is mostly used today in the industry [3]. Several studies show that recommendation systems based on hybrid approaches can provide more accurate recommendations than the pure approaches [3] that are mentioned above.

III. Proposed Framework

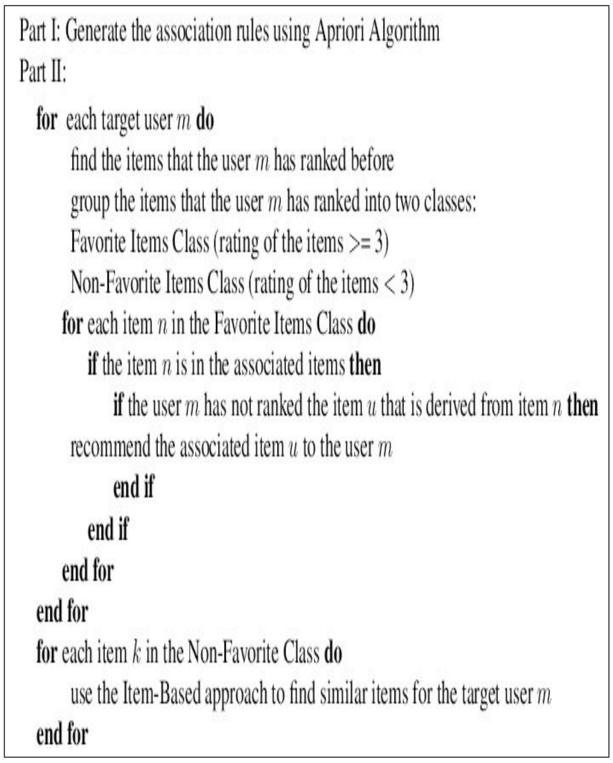
In this section, we provide the details and description of our proposed framework. We illustrate the use of the algorithm and how it works on the context of a recommendation system.

A. Overview

We propose a hybrid recommendation framework that integrates association rule mining with a content-based approach, based on an assumption that the $(User \times Item)$ space has a large number (e.g., larger than 1000) of users but a small number (e.g., less than 50) of items. We use the Apriori algorithm [5] to generate a set of association rules. The Apriori algorithm mines over the frequent sets to discover association rules. The most important parameters in the Apriori algorithm are minimum support count and minimum confidence [13]. Generated association rules play an important role in our proposed recommendation framework.

B. The Hybrid Recommendation Algorithm

Our proposed framework consists of two parts. The first part is to generate a set of association rules using the Apriori algorithm. The second part is to apply the generated association rules to recommend items for a user. Specifically, the proposed framework addresses the recommendation of *Favorite* and *Non-Favorite* items. For *Favorite* items, the framework straightly applies the generated association rules to offer recommendations for the user; for *Non-Favorite* items, the framework applies a content-based approach to offer recommendations. Basically, our proposed algorithm considers all the items that are rated by a user even if the ratings are low. Below is the proposed algorithm:



```
Part I: Generate the association rules using Apriori Algorithm
Part II:
  for each target user $m$ do
      find the items that the user $m$ has ranked before
      group the items that the user $m$ has ranked into two classes:
      Favorite Items Class (rating of the items >= 3)
      Non-Favorite Items Class (rating of the items < 3)
    for each item $n$ in the Favorite Items Class do
      if the item $n$ is in the associated items then
          if the user $m$ has not ranked the item $u$ that is derived from item $n$ then
      recommend the associated item $u$ to the user $m$
          end if
      end if
    end for
  end for
  for each item $k$ in the Non-Favorite Class do
      use the Item-Based approach to find similar items for the target user $m$
  end for
```

Fig. 1. The Algorithm of the Proposed Recommendation Framework

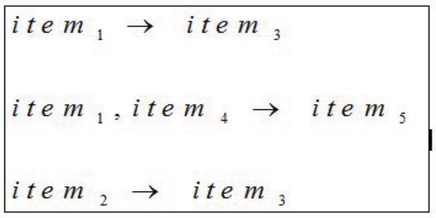
1) *Association Rule Generation:* First of all, to apply our proposed algorithm, we first need to obtain the required association rules via the Apriori algorithm. The inputs of the Apriori algorithm are: the transactions file, minimum support, and minimum confidence. The transactions file in our context is basically the ratings matrix as shown in the Table I.

TABLE I
THE TRANSACTIONS FILE IN THE FORM OF A BINARY RATINGS MATRIX

$User/Item$	$item_1$	$item_2$	$item_3$	$item_4$	$item_5$	......	$item_n$
$user_1$	1	1	0	0	1	......	1
$user_2$	0	1	0	1	1	......	1
$user_3$	0	0	1	0	0	......	0
.....	.....	.....	.....	.....	.....	......	.....
$user_m$	1	0	0	0	1	......	0

In the above matrix, 0 means the $user_m$ has not yet ranked the $item_n$. 1 means the $user_m$ has ranked the $item_n$.
After running the Apriori algorithm, and based on the minimum support and minimum confidence, a list of strong association rules is obtained. For example, a list of association rules is shown in the Fig. 2.

2) *Favorite and Non-Favorite Item Distinction:* Once the strong association rules are generated, we will distinguish an item as either favorite or non-favorite. First, for each $user_m$ an array of rated items by the user will be created. Then, the table will be divided into two classes, *Favorite Items* and *Non-Favorite Items* based on the ratings of the items. Rating of the items $>= 3$ is considered *Favorite Items*, and rating of the items < 3 is considered *Non-Favorite Items*. This information



$item_1 \rightarrow item_3$

$item_1, item_4 \rightarrow item_5$

$item_2 \rightarrow item_3$

Fig. 2. Example of strong association rules

can be obtained from original ratings matrix as shown in the Table II.

TABLE II
THE ORIGINAL RATINGS MATRIX

$User/Item$	$item_1$	$item_2$	$item_3$	$item_4$	$item_5$	......	$item_n$
$user_1$	5	1	ϕ	3	ϕ	......	3
$user_2$	4	1	ϕ	ϕ	1	......	4
$user_3$	ϕ	3	ϕ	4	5	......	4
.....	.....	.....	.....	.....	.....	......	.....
$user_m$	ϕ	4	5	2	1	......	1

The next step in our proposed algorithm is for each $item_n$ in the Favorite Items table, we check if the $item_n$ is in the left hand side of the generated association rules, and we check if the item $item_u$ that is in the right hand side is not rated by the user. Then, we can recommend the $item_u$ to the user.

For example, the $user_1$ has given the $item_1$ rating of 5 which is classified as a favorite item, and the $item_1$ is in the left hand side of the generated association rules as shown in the Fig. 2.

Next, from the favorite items table of the $user_1$, we see that the $item_3$ has not rated yet by the $user_1$, and according to our proposed algorithm, we can recommend the $item_3$ to the $user_1$.

3) *Non - Favorite Items:* In the first part of our proposed framework, we evaluate *Favorite Items* that a user has seen in the past, and based on those items, the system uses an association rule mining technique to recommend new items to a user. The second part of the proposed framework is to address *Non-Favorite Items* that have been seen by a user. The framework evaluates those items, and it recommends new items to a user. Note that most current recommendation systems only address the items that users have highly rated in the past and recommend similar items to a target user. The current recommendation system focus on items that are the favorites of users and generally discards the non-favorite ones. Our proposed framework overcomes this limitation. For example, in the context of a movie recommendation, our framework may still recommend an action movie to a user even though the user has already rated some other action movie as a non-favorite item.

To implement this part, an item-based approach is used in our proposed framework. The technique is to find similar items to those items that are considered *Non-Favorite Items*. For example, if a user has watched a movie *Die Hard I*, and she/he did not like it. The system will find a similar movie to the *Die Hard I* and recommend it to the user. The words that describe an item are the main features to decide the similarity among items. For example, to decide if two movies are similar to each other, we consider the genres of movies such as action, classic, drama etc. In our proposed framework, the genres of each movie are considered as a vector. The vector represents the genres in binary values 0 or 1. For example, the movie *Toy Story* can be represented as a vector with the following binary values: $(0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$. It means that the movie *Toy Story* is an animation, a children, and a comedy movie.

In our proposed framework, we use Jaccard coefficient to measure the similarity between two items [8]. The Jaccard coefficient is used to compute the similarity between two binary vectors, and it is defined in the following formula [14]:

$$Jaccard(i, j) = \frac{|S(i) \cap S(j)|}{|S(i) \cup S(j)|}\ , \tag{1}$$

where S denotes the sample set of items i and j.

So, the Jaccard coefficient is defined as the size of the intersection of the sample sets of the items i and j and is divided by the size of the union of the same sample sets and items [14]. Since the Jaccard coefficient is used to measure the similarity between two binary vectors, for simplicity, it can be illustrated in the following formula [15]:

$$Jaccard = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}\ , \tag{2}$$

where M_{01} is the number of attributes where object i was 0 and object j was 1, M_{10} = the number of attributes where object i was 1 and object j was 0, M_{00} = the number of attributes where object i was 0 and object j was 0, and M_{11} is the number of attributes where object i was 1 and object j was 1 [15].

IV. Experiments and Results

This section presents an experimental study of our proposed framework. It describes the experimental setup, presents the experiment results, and finally it summarizes our observation.

A. Dataset

We use the dataset of MovieLens, provided by GroupLens Research [16]. It is a public dataset. It consists of 100,000 movies ratings in a scale of 1-5 from 943 users on 1,682 movies. The dataset is already cleaned up. There is no need to preprocess the datasets. But, we have reformatted the dataset files to fit into our implementation of the proposed algorithm.

B. Hardware and Software

This section provides information about the hardware and software that are used to conduct the experiments.

1) Hardware:

- Processor Type: Intel Core i3 CPU
- Processor Speed: 2.53 GHz
- Available Ram: 4.00 GB

2) Software: To generate the association rules, we have used WEKA software [17]. WEKA software provides machine learning algorithms to implement several data mining tasks. It is open source software. Additionally, we used Java with Eclipse IDE [18] to implement our proposed algorithm, and to write several associated functions.

C. Validation

In the experiment, we have used five fold cross validation. When the algorithm generates an associated movie for a particular user, the rating of the movie is predicted by getting the ratings of the associated movie from other users that they have rated the movie and average the ratings.

We measure the accuracy by using two different evaluation metrics:

Mean Absolute Error (MAE): Mean absolute error (MAE) is a statistical accuracy metric that is used to measure the average absolute deviation between a predicted rating and the user's actual rating of an item [20]. MAE is widely used in evaluating the accuracy of a recommendation system [19]. MAE can be computed by the following equation:

$$MAE = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N} \tag{3}$$

where p_i is the predicted rating, r_i is the actual rating, and N is the total number of the ratings.

Root Mean Squared Error (RMSE): Shani and Gunawardana [21] state that the Root Mean Squared Error (RMSE) is perhaps the most popular metric used in evaluating accuracy of predicted ratings in recommendation systems [21]. It measures the quality of predicted ratings [22]. It can be computed as the following:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (p_i - r_i)^2}{N}} \tag{4}$$

D. Experiments

In this section, we illustrate the details of the experiments that are done on the proposed algorithm's parts which are *Favorite Items* and *Non-Favorite Items*, and the results that are extracted from those experiments.

TABLE III
THE RATING MATRIX ($User \times Item$)

$User/Item$	$item_1$	$item_2$	$item_3$	......	$item_{1682}$
$user_1$	1	1	0	......	1
$user_2$	0	1	0	......	1
$user_3$	0	0	1	......	0
.....	.....	.....	.....	......	.....
$user_{943}$	1	0	0	......	0

E. Experiments on Favorite Item Recommendation

In the generating association rules part, we consider each row in the ratings matrix ($User \times Item$) as a transaction to run the Apriori algorithm and obtain the association rules. Table III shows the ratings matrix. In the above matrix, 0 means for example the $user_1$ has not yet ranked the $item_3$. 1 means the $user_1$ has ranked the $item_1$.

1) The Apriori Algorithm on the Entire Rating Matrix: To apply the Apriori algorithm on WEKA, we need to update the above binary rating matrix (Table III) to a Boolean rating matrix, as shown in Table IV. $False$ means the movie has not rated yet, and $True$ means the movie has rated by the user.

TABLE IV
BOOLEAN RATING MATRIX ($User \times Item$)

$User/Item$	$item_1$	$item_2$	$item_3$	......	$item_n$
$user_1$	$True$	$True$	$False$	......	$True$
$user_2$	$False$	$True$	$False$	......	$True$
$user_3$	$False$	$False$	$True$	......	$False$
.....	.....	.....	.....	......	.....
$user_m$	$True$	$False$	$False$	......	$False$

At the beginning of this experiment, we tried to run the Apriori algorithm on the entire dataset that contains 943 users and 1,682 items. The WEKA crashed and failed to produce any association rules due to a lack of memory issue. The Apriori algorithm scans the database each time that the algorithm mines over the dataset, and it produces a large number of candidate itemsets [5]. The Apriori algorithm is not efficient to work on two dimensional space ($User \times Item$) with a huge number of items in the space. The algorithm takes an insufficient amount of time to generate the association rules. Additionally, in a machine with a limited memory size and a huge dimensional space, software like WEKA will not be able to generate the association rules due to a memory issue.

One solution that we have tried is to reduce the items in the training dataset and keep the number of users as is. We generated a training dataset with items (movies) that have been rated by at least 100 users. Thus, the number of items has been reduced from 1,682 to 117. With this dataset, we ran the Apriori algorithm in WEKA software with parameters of 50 % minimum support count, 90 % confidence, and 50,000 rules. WEKA crashed, and it was not able to produce the association rules due to a memory issue. To solve the problem, we reduced the number of rules to 5,000. WEKA was able to produce the 5,000 association rules. But, the FALSE value dominated the results of the association rules, and it provided irrelevant information.

The reason for getting this irrelevant information is because of the sparsity of the data in the ratings matrix ($User \times Item$). As we mentioned earlier in the literature review chapter, the total number of ratings is important in the recommendation systems. To provide accurate recommendations, a sufficient number of ratings should exist in the system [3]. Therefore, and to reduce the rate of sparsity, we generated a training dataset with items (movies) that were rated by at least 320 users. This operation produced 12 items, and the number of users was kept the same 943. WEKA was able to produce 16 rules that were considered relevant information.

2) Results: The ratings matrix ($User \times Item$) with 943 users and 12 items (that have been rated by at least 320 users) has been used in this experiment for all five fold cross validation. When the algorithm generates an associated movie (recommended movie) for a particular user, the rating of the movie is predicted by getting the ratings of the associated movie from other users who have rated the movie and average the ratings. In most cases, the computed-predicted ratings will be decimal numbers (e.g. 3.7256), and the actual ratings that are provided by the users are positive integers. This can cause variation in the results. Therefore, we have evaluated the results in three different cases:

- Case I: Evaluate the results of the computed-predicted ratings in decimal form directly.
- Case II: Apply the ceiling function to the predicted ratings and evaluate them.
- Case III: Apply the floor function to the predicted ratings and evaluate them.

The Table V shows the results of the computed- predicted ratings in decimal form. The Table VI shows the results after applying the floor function to the computed-predicted ratings. The Table VII shows the results after applying the ceiling function to the computed-predicted ratings.

TABLE V
ACCURACY MEASURED IN TERMS OF MAE AND RMSE

$Fold\ Cross\ Validation$	MAE	$RMSE$
$1st$	0.669324874	0.871004639
$2nd$	0.691398246	0.869712575
$3rd$	0.815557926	1.037908492
$4th$	0.559849015	0.677475326
$5th$	0.696621347	0.93701284
$Mean$	0.6865502804	0.8786227744

TABLE VI
ACCURACY MEASURED IN TERMS OF MAE AND RMSE (AFTER APPLYING THE FLOOR FUNCTION)

$Fold\ Cross\ Validation$	MAE	$RMSE$
$1st$	0.685106383	0.915586081
$2nd$	0.705069124	0.908231684
$3rd$	0.84375	1.120825589
$4th$	1.516129032	1.616447718
$5th$	0.710526316	0.973328527
$Mean$	0.892116171	1.10688392

Now, we can summarize the results on the following table (Table VIII) and chart (Fig. 3).

TABLE VII
ACCURACY MEASURED IN TERMS OF MAE AND RMSE (AFTER APPLYING THE CEILING FUNCTION)

Fold Cross Validation	*MAE*	*RMSE*
1st	0.774468085	1.177809001
2nd	0.746543779	1.148029769
3rd	1.11875	1.57916117
4th	0.602150538	0.789718883
5th	0.789473684	1.235441536
Mean	0.806277217	1.186032072

TABLE VIII
EVALUATION OF EXPERIMENT'S RESULTS

Evaluation	*MAE*	*RMSE*
In Decimal Numbers	0.6865502804	0.8786227744
After Appl. Floor Func.	0.892116171	1.10688392
After Appl. Ceiling Func.	0.806277217	1.186032072

From the chart and table above, it clearly appears that the predicted ratings in decimal form provide more accurately-predicted ratings.

F. Experiments on Non-Favorite Items Recommendation

To implement the second part of our proposed framework, we consider the attributes that describe the item. Each movie is described by its genre. The genre of each movie represents binary values. Thus, each movie can be represented as a vector. The Table IX shows how we represent a movie based on its genre.

TABLE IX
THE REPRESENTATION OF A MOVIE

Movies/Genres	Action	Adventure	Animation	....
$Movie_1$	0	1	0	.....
$Movie_2$	1	0	0	.....
$Movie_3$	0	1	0	.....
$Movie_4$	1	1	0	.....
.....	.....	.....	.....	.....
$Movie_n$	1	0	1	.....

In the experiment, we had 19 attributes that represent the genres of each movie: *Unknown, Action, Adventure, Animation,*

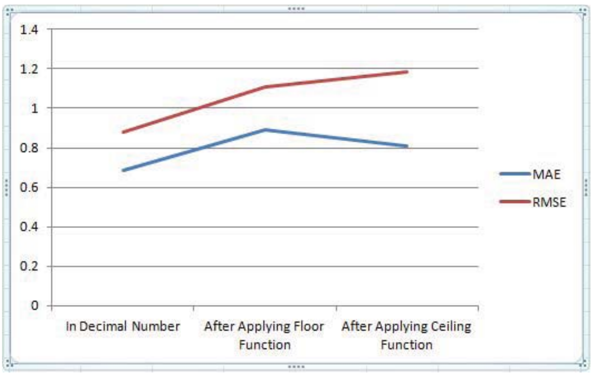


Fig. 3. The results of the evaluation of the three cases

Children, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, and Western. For each movie that is in the *Non-Favorite Items* category of a user, we applied the Jaccard coefficient to measure the similarity between the movie that was not liked (in the *Non-Favorite Items* category) by a user and other movies that have not been seen yet, and return most similar movies to the user.

1) Results: In this experiment, we have used five fold cross validation, and we used the same evaluating measures in the *Favorite Items* part. We repeated the experiment with each training and test dataset. The predicted ratings of the returned list of similar movies are computed using the same *Favorite Items* part. The following tables (Table X, Table XI, and Table XII) show the results after applying the Jaccard coefficient with similarity of 50 % and up among the movies.

TABLE X
ACCURACY MEASURED IN TERMS OF MAE AND RMSE

Fold Cross Validation	*MAE*	*RMSE*
1st	0.8900545064	1.094071759
2nd	0.953473449	1.148272952
3rd	0.764562879	0.95668796
4th	0.784482212	0.992369605
5th	0.967204261	1.333494909
Mean	0.871955461	1.104979437

TABLE XI
ACCURACY MEASURED IN TERMS OF MAE AND RMSE (AFTER APPLYING THE FLOOR FUNCTION)

Fold Cross Validation	*MAE*	*RMSE*
1st	1.075396825	1.334820599
2nd	1.298969072	1.572967515
3rd	0.842281879	1.139586644
4th	1.205211726	1.522897975
5th	1.385135135	1.800900676
Mean	1.161398927	1.474234682

TABLE XII
ACCURACY MEASURED IN TERMS OF MAE AND RMSE (AFTER APPLYING THE CEILING FUNCTION)

Fold Cross Validation	*MAE*	*RMSE*
1st	1.345238095	1.698505412
2nd	1.020618557	1.282984065
3rd	1.104026846	1.441242939
4th	0.973941368	1.206659048
5th	2.081081081	2.53089024
Mean	1.304981189	1.632056341

The Table XIII provides the summary of the results of the evaluation of the *Non-Favorite Items* part. The results of the

TABLE XIII
SUMMARY OF EXPERIMENT'S RESULTS OF NON-FAVORITE ITEMS

Evaluation	*MAE*	*RMSE*
In Decimal Numbers	0.871955461	1.104979437
After Appl. Floor Func.	1.161398927	1.474234682
After Appl. Ceiling Func.	1.304981189	1.632056341

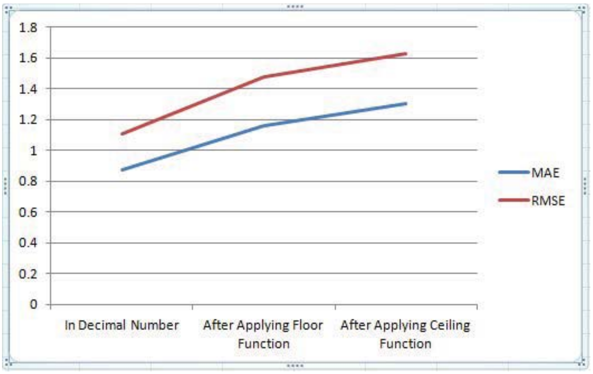


Fig. 4. Experiment's results of Non-Favorite items

experiment on *Non-Favorite Items* from the table (Table XIII) and chart (Fig. 4) show the predicted ratings in decimal form, which provides more accurate predicted ratings than the other floor and ceiling functions.

V. Observations

From the experiments that we have conducted, we observed the following points:

- The Apriori algorithm is not efficient to work on two dimensional space $(User \times item)$ with a large number of items in the space. The algorithm takes insufficient time to generate the association rules.
- When the item space contains a large number of items, the Apriori algorithm can generate many association rules that are irrelevant to the user.
- In a machine with a limited memory size and a huge dimensional space, software like WEKA will not be able to generate the association rules due to the memory issue.
- The proposed algorithm works well and fits on two dimensional spaces $(User \times item)$ with items that are significantly fewer than users. Thus, we can apply our algorithm on two dimensional space such as $(Student \times Course)$, $(Tourist \times VacationPlace)$, or $(Person \times Restaurant)$.
- The computed-predicted ratings as decimal numbers provide more accurately-predicted ratings.

VI. Conclusion

Our research has proposed a hybrid framework recommendation system to be applied on two dimensional spaces $(User \times Item)$ with a large number of *Users* and a small number of *Items*. Our proposed framework makes use of both *favorite* and *non-favorite* items of a particular user. The proposed framework is built upon the integration of association rules mining and the content-based approach. Our proposed framework is divided into two parts: In the first part, we evaluate *Favorite Items* that a user has seen in the past, and based on those items, the system uses association rules mining technique to recommend new items to a user. The second part of the proposed framework is to consider *Non-Favorite Items* that a user has seen before, and apply item-based approach to find similar items to those on the *Non-Favorite Items* category. We have done experiments on the proposed algorithm's part which are *Favorite Items* and *Non-Favorite Items*, and the results that are extracted from those experiments show that our proposed framework can provide accurate recommendations to users.

References

[1] B., Shneiderman (2008). Copernican challenges face those who suggest that collaboration, not computation are the driving energy for socio-technical systems that characterize Web 2.0. Science, 319, 1349-1350.
[2] E.,Vozalis, and K. G.,Margaritis (2003, September). Analysis of recommender systems algorithms. In Proceedings of the 6th Hellenic European Conference on Computer Mathematics and its Applications (HERCMA-2003), Athens, Greece.
[3] G.,Adomavicius, and A.,Tuzhilin (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on, 17(6), 734-749.
[4] Z.,Huang, D.,Zeng, and H., Chen (2004). A unified recommendation framework based on Probabilistic Relational Models. In Fourteenth Annual Workshop on Information Technologies and Systems (WITS) (pp. 8-13).
[5] J.,Han, and M.,Kamber (2006). Data mining: concepts and techniques (2nd ed.). Amsterdam: Elsevier .
[6] M.,Hegland (2007). The apriori algorithma tutorial. Mathematics and Computation in Imaging Science and Information Processing, 11, 209-262.
[7] G.,Linden, B.,Smith, and J.,York (2003). Amazon. com recommendations: Item-to-item collaborative filtering. Internet Computing, IEEE, 7(1), 76-80.
[8] A.,da Silva Meyer, A. F.,Garcia, A. P.,de Souza, and C. L.,de Souza (2004). Comparison of similarity coefficients used for cluster analysis with dominant markers in maize (Zea mays L.). Genetics and Molecular Biology, 27, 83-91.
[9] X.,Su, and T. M.,Khoshgoftaar (2009). A survey of collaborative filtering techniques. Advances in Artificial Intelligence, 2009, 4.
[10] B.,Amini, R.,Ibrahim, and M.S.,Othman (2011). Discovering the impact of knowledge in recommender systems: A comparative study. arXiv preprint arXiv:1109.0166.
[11] M. A.,Ghazanfar, and A.,Prugel-Bennett (2010, January). A scalable, accurate hybrid recommender system. In Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on (pp. 94-98). IEEE.
[12] T.,Tran, and R.,Cohen (2000, July). Hybrid recommender systems for electronic commerce. In Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press.
[13] R.,Perego, S.,Orlando, and P.,Palmerini (2001). Enhancing the apriori algorithm for frequent set counting. Data Warehousing and Knowledge Discovery, 71-82.
[14] B.,Sigurbjrnsson, and R.,Van Zwol (2008, April). Flickr tag recommendation based on collective knowledge. In Proceedings of the 17th international conference on World Wide Web (pp. 327-336). ACM.
[15] P.,Tan, M.,Steinbach, and V.,Kumar (2005). Introduction to data mining. Boston: Pearson Addison Wesley.
[16] MovieLens Data Sets. (2011, August 8). GroupLens Research. Retrieved November 18, 2012, from http://www.grouplens.org/node/73
[17] Weka 3 - Data Mining with Open Source Machine Learning Software in Java . (n.d.). Machine Learning Group at University of Waikato . Retrieved November 18, 2012, from http://www.cs.waikato.ac.nz/ml/weka
[18] About the Eclipse Foundation. (n.d.). Eclipse. Retrieved November 18, 2012, from http://www.eclipse.org/
[19] B.,Sarwar, G.,Karypis, J.,Konstan,and J.,Riedl (2001, April). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295). ACM.
[20] J. L.,Herlocker, J. A.,Konstan, L. G.,Terveen, and J. T.,Riedl (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), 5-53.
[21] G.,Shani, and A.,Gunawardana (2011). Evaluating recommendation systems. Recommender Systems Handbook, 257-297.

[22] Y.,Koren (2008, August). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 426-434). ACM.

[23] Alsalama, Ahmed (2013). A Hybrid Recommendation System Based on Association Rules. Masters Theses and Specialist Projects. Paper 1250. http://digitalcommons.wku.edu/theses/1250