5-2013

A Hybrid Recommendation System Based on Association Rules

Ahmed Alsalama
Western Kentucky University, ahmed.alsalama263@topper.wku.edu

A HYBRID RECOMMENDATION SYSTEM BASED ON ASSOCIATION RULES

A Thesis
Presented to
The Faculty of the Department of Computer Science
Western Kentucky University
Bowling Green, Kentucky

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science

By
Ahmed Alsalama

May 2013

A HYBRID RECOMMENDATION SYSTEM
BASED ON ASSOCIATION RULES

Date Recommended 03/26/13

[illegible]
Dr. Qi Li, Director of Thesis

[illegible]
Dr. Guangming Xing

[illegible]
Dr. Zhonghang Xia

[illegible] 5-8-13
Dean, Graduate Studies and Research Date

My thesis research is dedicated to my mother, Monirah, who prays for me every day. To my father, Mohammed. To my great wife, Masheil, and to my lovely kids, Turki and Wateen.

ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Qi Li, who has supported me during my thesis research. His knowledge and guidance were very helpful to me. I would like to express my appreciation and gratitude for his supervision and support. Without his continuous support and help, my thesis would not have been accomplished.

I gratefully thank my committee members, Dr. Guangming Xing and Dr. Zhonghang Xia, for their time to read my thesis and provide valuable comments and corrections.

I would also like to thank the Ministry of Higher Education in the Kingdom of Saudi Arabia for giving me the opportunity to study abroad. This opportunity has given me valuable knowledge.

Finally, I would like to thank the Saudi Arabian Cultural Mission to the U.S. for their support and advice during my study in the United States.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

A HYBRID RECOMMENDATION SYSTEM BASED ON ASSOCIATION RULES

Ahmed Alsalama May 2013 59 Pages

Directed by: Dr. Qi.Li, Dr. Guangming Xing, and Dr. Zhonghang Xia

Department of Computer Science Western Kentucky University

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 a hybrid framework recommendation system to be applied on two dimensional spaces *(User × 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.

Introduction

1.1 Overview

Today, most companies and corporations around the world use technology solutions for their work and business environment. Those organizations implement computer systems to deal with their business transactions. For example, banks and financial institutions allow their clients to make financial transactions such as making payments and transferring money through the World Wide Web. In addition, retail stores such as Wal-Mart use electronic devices to scan items that customers purchase, and all transactions' information are stored in the database. Amazon allows the customers to buy and sell their books and other items through its website, and customers can provide feedback in the form of ratings or comments. All the feedback that is provided by the customers is also stored in the database. A huge amount of this data is stored in data warehouses. Another example is Netflix, which allows the customers to rate the movies that they watch, and the feedback information is stored. The databases and data warehouses of such companies and corporations contain huge amounts of data. Analysis of this huge amount of data to gain useful information is a significant matter.

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 [10]. Also, Amazon can study a customer's profile and analyze the feedback that the customer provides in order 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 [4]. The task of recommendation systems is to predict the ratings of the items that the user has not seen or ranked before [5]. Based on that predicated rating, the recommendation system will be able to recommend other items to the user [5].

There are different approaches to recommendation systems that are used to serve in different contexts based on system needs [33]. The content-based approach deals with item profiles and user profiles, and it is designed to recommend text-based items [5]. The collaborative filtering approach is widely used in commercial areas. Amazon uses the collaborative filtering approach to recommend books and other products to its customers [5]. Recommendation systems based on collaborative filtering recommend items to a particular user based on the similar items that have been rated by some other users, and the target user and the other users share the same preferences of items or products [5]. The demographic approach recommender systems use demographic information such as the gender, age, and date of birth of respective users in order to recommend items [1]. The hybrid approach has been introduced to go over the limitations and drawbacks of the other recommendation systems approaches [5]. The hybrid approach combines two or more recommendation systems approaches together to eliminate the limitations of pure approaches [7]. Several studies show that hybrid approaches can provide more accurate recommendations than other approaches [5].

On the other hand, data mining techniques such as finding association rules and frequent patterns are used widely to analyze customer buying behavior [9]. For example, finding association rules and frequent itemsets are used by retailers to do market basket analysis to discover the buying habits of the customers to develop marketing strategies [9]. Recently, association rules mining has been extended to be used in recommendation systems. For example, Bendakir and Ameur have proposed a course recommendation system based on association rules [15]. Also, Xizheng has proposed a personalized recommendation system using association rule mining and classification in e-commerce [11].

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 $(User \times Item)$. We use the training and test datasets of MovieLens that is provided by GroupLens Research [12]. In addition, we use WEKA software [13] to generate association rules and do the data mining tasks that are required to implement and test the propsed framework.

1.2 Motivations

The Apriori algorithm requires scanning the database every time it generates the candidate itemsets in order to build the association rules [9]. 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. For example, in course recommendation systems, we have a large number of students and a limited number of courses. Also, in vacation recommendation systems, there are many tourists and some excellent destinations (we assume for example the tourists destinations are in a particular county such as the United States or European countries). Another example is restaurant recommendation systems in a city or town. In this kind of system, the data in the $(User \times Item)$ matrix is much less sparse than in movie or book recommendation systems. Also, the $(User \times Item)$ matrix is not big, so the performance of the Apriori algorithm will be more effective than other systems with a large $(User \times Item)$ matrix.

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.

1.3 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.

1.4 Thesis Organization

Chapter 2 is the background and related work, and it provides necessary concepts, methods, and algorithms of association rules mining and recommendation systems. Chapter 3 presents our approach in details, and it illustrates our proposed algorithm. Chapter 4 shows the experiments results of our proposed algorithm. Finally, the conclusion and the future work are presented in chapter 5.

2.1 Mining Frequent Patterns and Association Rules

Frequent patterns can be defined as patterns that appear frequently in the data set [9]. A set of items such as bread, butter, and milk that occur frequently together in a transaction is called frequent itemset [9]. Mining in a frequent itemset allows us to discover the associations and correlations among items in large transactional data sets [9]. For example, many retail stores collect and store huge amounts of data in their databases. These amounts of data can be mined to discover interesting correlation relationships among these database's records that can help the business managers to make decision such as cross-marketing, customer buying behavior analysis, and catalog design [9].

2.1.1 Association Rules

Let I = $\{I_1, I_2, I_3, ..., I_m\}$ be a set of items [9]. Let D be a set of transaction in a database where each transaction T is a set of items such that $T \subseteq I$ [9]. Each transaction in the database is associated with an identifier TID , and let A be a set of items [9]. A transaction T contains A if and only if $A \subseteq T$ [9]. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, and $A \cap B$ = $\varnothing$ [9]. The rule $A \Rightarrow B$ holds in the set of database transactions D with support s, where s is the percentage of transactions in D that contain $A \cup B$ which means the probability $P(A \cup B)$ indicates that a transaction contains the union of set A and set B [9]. In addition, the confidence c of the rule $A \Rightarrow B$ in the transaction set D is the percentage of transaction in D that containing A that also containing B too which means the conditional probability $P\ (B \mid A)$ [9]. Therefore, the rules that satisfy both a minimum support threshold and a minimum confidence threshold are called strong association rules [9].

The confidence c of rule $A \Rightarrow B$ can be derived from the support count (the number of transactions that contain the itemset) of A and $A \cup B$ as is shown by the following equation [9]:

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support - count(A \cup B)}{support - count(A)} \tag{2.1}$$

In general, finding all frequent itemsets and generate strong association rules are the main process of association rule mining [9].

2.1.2 The Apriori Algorithm

The Apriori algorithm is a well-known algorithm that is used for mining frequent itemsets for Boolean association rules [9]. It is an algorithm for efficient association rule discovery [24]. The algorithm was proposed by R. Agrawal and R. Srikant in 1994 [9]. 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* [9].

In order to generate the association rules, the set of one frequent itemsets can be found by scanning the database to accumulate the count for each item in the transactions; then, the algorithm collects the items that satisfy the minimum support [9]. The resulting set is denoted L_1 [9]. Next, L_1 is used to find L_2, which is the set of two frequent itemsets. Also, it is used to find L_3, and so on, until no more frequent *k-itemsets* can be found [9]. Finding each L_k is expensive work; it requires one full scan of the database [9]. Thus, the generation of candidate sets is costly in the Apriori algorithm [25]. The joint step is required to find L_k [9]. A set of candidate *k-itemsets* can be generated by joining $L_k - 1$ with itself [9]. This set of candidates is denoted C_k [9].

To illustrate the Apriori algorithm, let us assume that we have these five transactions in the database [9]:

Table 2.1: The transactions in the database

TID	$items - bought$
$T100$	$\{M, O, N, K, E, Y\}$
$T200$	$\{D, O, N, K, E, Y\}$
$T300$	$\{M, A, K, E\}$
$T400$	$\{M, U, C, K, Y\}$
$T500$	$\{C, O, O, K, I, E\}$

TID is the transaction ID and $items - bought$ are the items that are bought by the customers.

We use the Apriori algorithm to find frequent itemsets and generate the association rules that satisfy the minimum support s which is 60% and minimum confidence c which is 80%.

First, we generate all the candidates of one itemset C_1 as shown in the Table 2.2:

Table 2.2: The candidates of one itemset C_1

$Item$	$SupportCount$
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
O	3
U	1
Y	3

Next, we remove the items that do not satisfy the support count. Table 2.3 shows the frequent one itemset L_1:

Table 2.3: The frequent one itemset L_1

$Item$	$SupportCount$
E	4
K	5
M	3
O	3
Y	3

Next, we get all the candidates of two itemsets C_2 by applying the joint operation on L_1 $(C_2 = L_1 \boxtimes L_1)$. Then, we remove the itemsets that do not satisfy the support count as shown in Table 2.4:

Table 2.4: The frequent two itemset L_2

$Item$	$SupportCount$
E, K	4
E, O	3
E, O	3
K, M	3
K, O	3
K, Y	3

Then, we do the joint operation again on L_2 to get C_3:

$C_3 = L_2 \boxtimes L_2$

$= \{\{E, K\}, \{E, O\}, \{K, M\}, \{K, O\}, \{K, Y\}\} \boxtimes$

$\{\{E, K\}, \{E, O\}, \{K, M\}, \{K, O\}, \{K, Y\}\}$

$= \{\{E, K, O\}, \{K, M, Y\}\}$

Next, we look for the subsets that are frequent:

$\{E, K, O\}$:

- {E,K} is frequent
- {E,O} is frequent
- {K,O} is frequent

$\{K, M, Y\}$:

- {K,M} is frequent
- {K,Y} is frequent
- {M,Y} is NOT frequent

We remove $\{K, M, Y\}$ since it contains a subset that is not a frequent itemset.

The frequent three itemsets L_3 is showing in the Table 2.5:

Table 2.5: The frequent three itemsets L_3

$Item$	$SupportCount$
E, K, O	3

Since the L_3 contains only one set, we cannot do the joint operation on L_3. Thus, $C_4 = \varnothing$, so we stop.

Then, we can list the association rules for example in the form of:

$buy(X, item_1)buy(X, item_2) \Rightarrow buy(X, item_3)$ with its support s and confidence c as shown in Table 2.6:

Table 2.6: Association rules

$Item$	$Support$	$Confidence$
$E, K \Rightarrow O$	$3/5 = 60\%$	$3/4 = 75\%$
$E, O \Rightarrow K$	$3/5 = 60\%$	$3/3 = 100\%$
$K, O \Rightarrow E$	$3/5 = 60\%$	$3/3 = 100\%$

Finally, we list the strong association rules that satisfy the minimum support s which is 60% and the minimum confidence c which is 80%. Table 2.7 shows the strong association rules:

Table 2.7: The strong association rules

$Rule$
$E, O \Rightarrow K$
$K, O \Rightarrow E$

2.2 Recommendation: Concepts and Approaches

Recommendation systems are widely used in e-commerce applications. They attempt to predict unrated items for a particular user [5]. So, upon the predicted ratings, the system will be able to recommend items to the users. Many corporations such as Amazon and Netflix use recommendation systems to recommend items or products to their customers [5].

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.

2.2.1 Basic Concepts

To provide more formal definition of recommendation systems, let U be a set of all possible users, and let I be a set of all possible items [5]. In many e-commerce applications, the space U and I can be very large. Let f be a utility function that measures the usefulness of an item i to a user u such as $U \times I \rightarrow R$ where R is an ordered set of non-negative integers or real numbers [5]. Then, for each user $u \in U$, we want to choose an item $i_u \in I$ to maximize the user's utility as shown below[5]:

$$\forall u \in U, i_u = arg\, max\, f(u, i) \tag{2.2}$$

In the context of recommendation systems, the utility of an item is usually represented by a rating. For example, James gave the movie *Spider Man* a rating of 4 (out of 5) [5]. Basically, the utility of an item indicates how a particular user liked a particular item [5]. Table 2.8 shows an example of $(User \times Item)$ rating matrix:

Table 2.8: A $(User \times Item)$ rating matrix

$User/Item$	$SpiderMan$	$DieHard\,I$	$DieHard\,II$	$TheFlight$	$BadBoys\,II$
$James$	5	7	6	2	ϕ
$Jessica$	2	ϕ	5	ϕ	9
$John$	7	ϕ	3	5	1
$Zack$	4	6	10	ϕ	ϕ
$Sara$	ϕ	7	8	3	ϕ

The table above shows the ratings for each movie that the users have watched (out of 10). ϕ indicates the movie has not been rated yet by the user. Therefore, the goal of recommendation systems is to predict unrated items [5]. Based on that predicated ratings, the recommendation systems will be able to select some items with highest predicted ratings and recommend them to the user [5].

2.2.2 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 [5]. The system then recommends the items that have a high degree of similarity to the user's preferences and tastes [5]. 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 [5]. Then, the system recommends the movies that have a high degree of similarity to the user's preferences [5].

A content-based approach usually deals with item profile $ItemProfile(i)$ and user profile $UserProfile(u)$ [5].

$ItemProfile(i)$ is a profile containing a set of attributes that describes an $item\, i$ [5]. These attributes or features are extracted from the item's content, and the content of an $item\, i$ usually is described with keywords [5]. $UserProfile(u)$ is a profile of a $user\, u$, and it contains the preferences of this user [5]. $UserProfile(u)$ can be defined as a vector of weights $(w_{i1}, w_{i2}, ..., w_{ik})$, where each weight w_{ui} represents the importance of the keyword k_i to the $user\, u$ [5]. Then, the utility function $f(u, i)$ can be defined as [5]:

$$f(u, i) = score(ItemProfile(i), UserProfile(u))$$

The utility function $f(u, i)$ is represented in the information retrieval literature by a scoring heuristic that is defined in terms of vectors $\vec{w_u}$ and $\vec{w_i}$, and the scoring for example can be computed by cosine similarity measure [5]. Below, we provide more details about the item and user profiles, and how we measure and compute the keyword weight and weights vector.

Item Profile:

The items that can be recommended to the user by the system are represented by a set of attributes or features [6]. For example, in movie recommendation systems, each movie can be described by some features or attributes such as directors, actors, and genres, etc [6].

Content-based approaches are designed to recommend text-based items [5]. Techniques from information retrieval are used to specify the content (keywords) [5].

Measure for Specifying Keyword Weight:

The term frequency/inverse document frequency (TF-IDF) measure is one of the best measures used for specifying keyword weight [5]. This measure is defined as follows: let N be the total number of documents that can be recommended to the users, and let k_i be the keyword that appears in n_i of the documents [5]. Also, let $f_{i,j}$ be the number of times the keyword k_i appears in the document d_j [5]. Then, $TF_{i,j}$ the term frequency of keyword k_i in document d_j is defined as [5]:

$$TF_{i,j} = \frac{f_{i,j}}{max_z f_{z,j}} \tag{2.3}$$

The inverse document frequency for keyword k_i is defined as [5]:

$$IDF_i = log\frac{N}{n_i} \tag{2.4}$$

Then, the TF-IDF weight for keyword k_i in document d_j is defined as [5]:

$$W_{i,j} = TF_{i,j} \times IDF_i \tag{2.5}$$

The content of document d_j is defined as [5]:

$$Content(d_j) = (w_{1j}, w_{2j}, ..., w_{kj})$$

User Profile:

In a content-based recommendation system, a user profile contains the user's preferences of items [5]. A user profile can be obtained by analyzing the content of all rated items [5]. Specifically, this profile is constructed by using the content (keyword) that has been analyzed using the methods that are mentioned in the item profile section [37]. Each item in the user's profile has a weight that denotes the importance of keyword k_i to the user [5]. This weight can be computed using average approach through a variety of techniques such as Rocchio algorithm, Bayesian classifier, Winnow algorithm, and cosine similarity measure [5].

A Bayesian classifier can be used to classify unrated items into two classes C_1 (relevant) or C_2 (irrelevant) [38]. For example, in an article recommendation system using a content approach, the user profile contains preferences of the user such as the user is interested in business articles. These articles have terms such as market, stock, business, money, etc. The system computes the weight of the terms using methods that are mentioned in the item profile section. Each article is represented as a vector, and each vector contains such terms with their respective weights. In order to recommend an article to the user, the system uses the similarity measure such as cosine similarity or classification techniques such as Naïve Bayesian classifier to recommend an article with high weight to the user [5]. Naïve Bayesian classifier is used to estimate the probability that an article belongs to a certain class C_1 (relevant) or C_2 (irrelevant) by giving a set of keywords $k_{1,j}$, ..., $k_{n,j}$ for that article[5]:

$$P(C_i|k_{1,j}, ..., k_{n,j}) \tag{2.6}$$

Experimental results show that the Naïve Bayesian classifier provides a highly accurate classification [5].

Limitations of Content-Based Approaches:

- Over-Specialization: A content-based approach tends to recommend items that are similar to the items rated before by the same user [5]. For example, a user who is interested in business articles will hardly receive a recommendation for an article in sports or technology [27].
- New User Problem: When a new user enters the system, she/he has no user profile and there are no rated items yet. Thus, the system will not be able to provide accurate recommendations [5].

2.2.3 Collaborative Filtering Approaches

Collaborative filtering approaches are widely used in e-commerce [39]. They have been successful in many e-commerce applications such as Amazon and Netflix [23]. It is a popular technique used to reduce information overload [23]. Amazon recommends books to their customers using the collaborative filtering approach [5]. 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 [5]. The system finds items for other users that have similar preferences as a query user [5]. 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 [5].

Collaborative filtering approaches are grouped into two general categories [5]:

- Memory-based approaches: They use the entire collection of the rated items in order to make recommendations or predictions [5].
- Model-based approaches: They allow systems to learn to recognize patterns in the data sets in order to make recommendations or predictions [8].

Memory-Based Approaches:

In a memory-based approach, it is important to measure the similarities between users or items [8]. There are many different similarity measures that are used to compute the similarities between users or items [8]. For example, the Pearson correlation measures [5]:

$$sim(a,b) = \frac{\sum_{s \in S_{ab}} (r_{a,s} - \bar{r_a})(r_{b,s} - \bar{r_b})}{\sqrt{\sum_{s \in S_{ab}} (r_{a,s} - \bar{r_a})^2} \sqrt{\sum_{s \in S_{ab}} (r_{b,s} - \bar{r_b})^2}} , \tag{2.7}$$

where a and b are the users, S_{ab} is the set of all items co-rated by both users a and b, and $\bar{r_a}$ is the average rating of the related rated items of the ath user.

Cosine measure is also widely used to compute the similarity between items [8]:

$$w_{i,j} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i}.\vec{j}}{\|\vec{i}\| * \|\vec{j}\|} \tag{2.8}$$

After computing the similarities between users or items, and in order to provide recommendations or predictions, computation methods of recommendations or predictions must be used such as Weighted Sum of Others' Ratings and Simple Weighted Average [8].

Then, the popular algorithm K-Nearest Neighbors is applied to get a subset of nearest neighbors of the target user based on the similarity that is computed using the above methods [8]. Finally, Top N recommendations are provided to the target user [8].

Model-Based Approaches:

In model-based approaches, classification, clustering, and regression algorithms can be used [8]. For example, the Bayesian classification and K-Means clustering algorithm are used in model based of collaborative filtering approach [8].

Limitation of Collaborative Filtering Approaches:

- New User Problem: Collaborative filtering has the same problem as the content-based approach which is new users entering the system [5]. In order to make recommendations to a user, the system needs to know the user's preferences from the ratings that the user makes [5]. Since the user is new in the system, she/he has not rated items yet. Thus, the system will not be able to provide accurate recommendations [5].

- New Item Problem: The systems should contain rated items in order to recommend some items to the users. When a new item enters the systems, the item has not rated by users yet. Therefore, the systems will not be able to recommend it to the users [5].

- Sparsity: Sparsity is a major problem for collaborative filtering approach [28]. The total number of ratings is important in the recommendation system. In order to provide accurate recommendations by the recommendation systems, sufficient number of ratings should exist in the systems [5]. For example, in movie recommendation systems, there are many movies that have been rated by only a few people [5]. The systems will rarely recommend these movies [5]. The utility matrix $(User \times Item)$ that is used in the collaborative filtering approach will be sparse [36]. Thus, providing accurate recommendations is challenging [8].

- Scalability: In many practical collaborative filtering recommendation systems, the number of users and items increase rapidly in the system [8]. Therefore, the system needs to provide more and complicated computational process, and this leads the computational resources going beyond the acceptable levels [40].

2.2.4 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 [1]. The demographic approach puts the users into groups based on their demographic characteristics [1]. For example, the system will put the users who belong to a certain zip code into one group. Also, the users of ages ranging from 18 to 25 years-old will be in one group. The recommendation systems based on demographic approaches assume that the users in the same group or category share the same interests and preferences [1]. The demographic system tracks the buying or rating behavior of the users within the same group or category [29]. If there is a new user entering the system, the system first will place the user into a particular group based on the user's demographic information. Then, the system will recommend products or items to the user based on the buying or rating behavior of the other users in the group [2].

An early example of a recommendation system based on demographic information was Grundy [17]. The purpose of the system is to recommend books to library visitors based on their personal information that is gathered from them through an interactive dialogue [17]. Another recent example of a recommendation system based on demographic groups is LIFESTYLE FINDER [18]. The system uses demographic groups from marketing research to recommend a range of products and services, and it gathers the data from users through a short survey [17].The advantage of the demographic based approach is: the system does not require maintaining a history of user ratings like in content based and collaborative filtering approaches [17].

There are some limitations of the demographic-based approach. The first limitation that the demographic system suffers from is how to identify the group or category that the user belongs to when the user is new to the system [1]. The second limitation is how to identify the interests and preferences of users within the same group [1]. The third limitation of the demographic approach is the demographic system works well when the demographic data is available to the system [1]. But, this kind of data is not easy to collect [3].

Therefore, few recommendation systems use the demographic approach due to the limitations of the demographic approach [3]. Moreover, the accuracy of recommendation systems based on demographic data is less than those recommendation systems based on content or collaboration filtering [3].

2.2.5 Hybrid Approach

Content-based and collaborative filtering approaches have been widely used in commercial and research areas. But, they have many limitations mentioned in the previous sections. Therefore, the hybrid approach has been introduced to avoid the limitations of the content-based and collaborative filtering approaches [5]. Several recommendation systems combine two or more approaches to gain better performance and eliminate some of the drawbacks of the pure recommendation systems approaches [34].

Currently, many recommendation systems combine the collaborative approach with some other approaches such as content-based approach and demographic approach [7]. Combining collaborative filtering and content-based approaches is mostly used today in the industry [5].

There are different ways to combine collaborative filtering and content-based approaches [5]:

- Recommendation systems can be developed by implementing content-based and collaborative filtering methods separately and combining their predictions [35].
- Adding content-based characteristics to collaborative filtering models [5].
- Adding collaborative filtering characteristics to content-based models [5].
- Constructing a general unifying model that incorporates both content-based and collaborative filtering characteristics [5].

Several studies show that recommendation systems based on hybrid approaches can provide more accurate recommendations than the pure approaches [5] that are mentioned above.

2.3 Recommendation Systems based on Association Rule Mining

There are some recommendation systems that use association rules mining techniques have been introduced in the literature. They are applied to various application areas in the real world such as e-Learning systems, e-Commerce systems, and course recommendation systems.

Chellatamilan and Suresh presented an idea for building a recommendation system for the e-Learning system using Association Rules Mining to provide students with the best selection of learning materials and e-Learning resources [14]. Their idea is to gather data from students using a survey questionnaire in area of educational background, IT experience, technology accessibility, frequency of their study patterns, demographics data, etc. In addition, the system analyzes students' logs of a Learning Management System (LMS) Moodle. Then, they apply data mining tools such as association rules to find frequent itemsets. Association rule mining, distance metrics such as Jaccard measure, and cosine of the angle are used to construct the recommendation system [14]. This system is required to gather personal and background data from the users in the form of a survey questionnaire. This is a major step in this system, and it can be considered as a disadvantage of the proposed recommendation system. Recommendation systems that require gathering data such as demographic data work well only if the data is available [1]. Thus, failure to provide such data can cause poor recommendations [1].

Our proposed framework does not require gathering information from users, such as demographic information, in order to provide recommendations which is an advantage over the system proposed by Chellatamilan and Sures.

Bendakir and Aïmeur proposed a course recommendation system based on association rules mining [15]. The system incorporates a data mining process with user ratings in recommendations [15]. Specifically, the architecture of the system is divided into two phases: an off-line phase which consists of a data mining process, and an on-line phase for the interaction of the systems with its users [15]. The off-line phase is used to extract association rules from the data, and the on-line phase uses the rules to infer course recommendations [15]. The advantage of this system is to allow the user (student) to evaluate the previous recommendations, so the system can be enhanced, and the rules are updated as more evaluations of the previous recommendations are provided by the students [15]. But, this system has disadvantages; it does not make use of a student's academic background [15]. Additionally, this system was developed to fit a certain context of recommendation systems, which is a course recommendation system.

Proposed Framework

In this chapter, 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. Also, we give comparisons of the proposed framework with other recommendation methods.

3.1 Overview

As we mentioned in the literature review chapter, recommendation systems are widely used in e-commerce applications. The goal of recommendation systems is to recommend items to a user. Different approaches of the recommendation systems are introduced in the literature such as content-based, collaborative filtering, demographic, and hybrid approaches.

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 [9] 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 [26]. Generated association rules play an important role in our proposed recommendation framework, as illustrated in Figure 3.1.

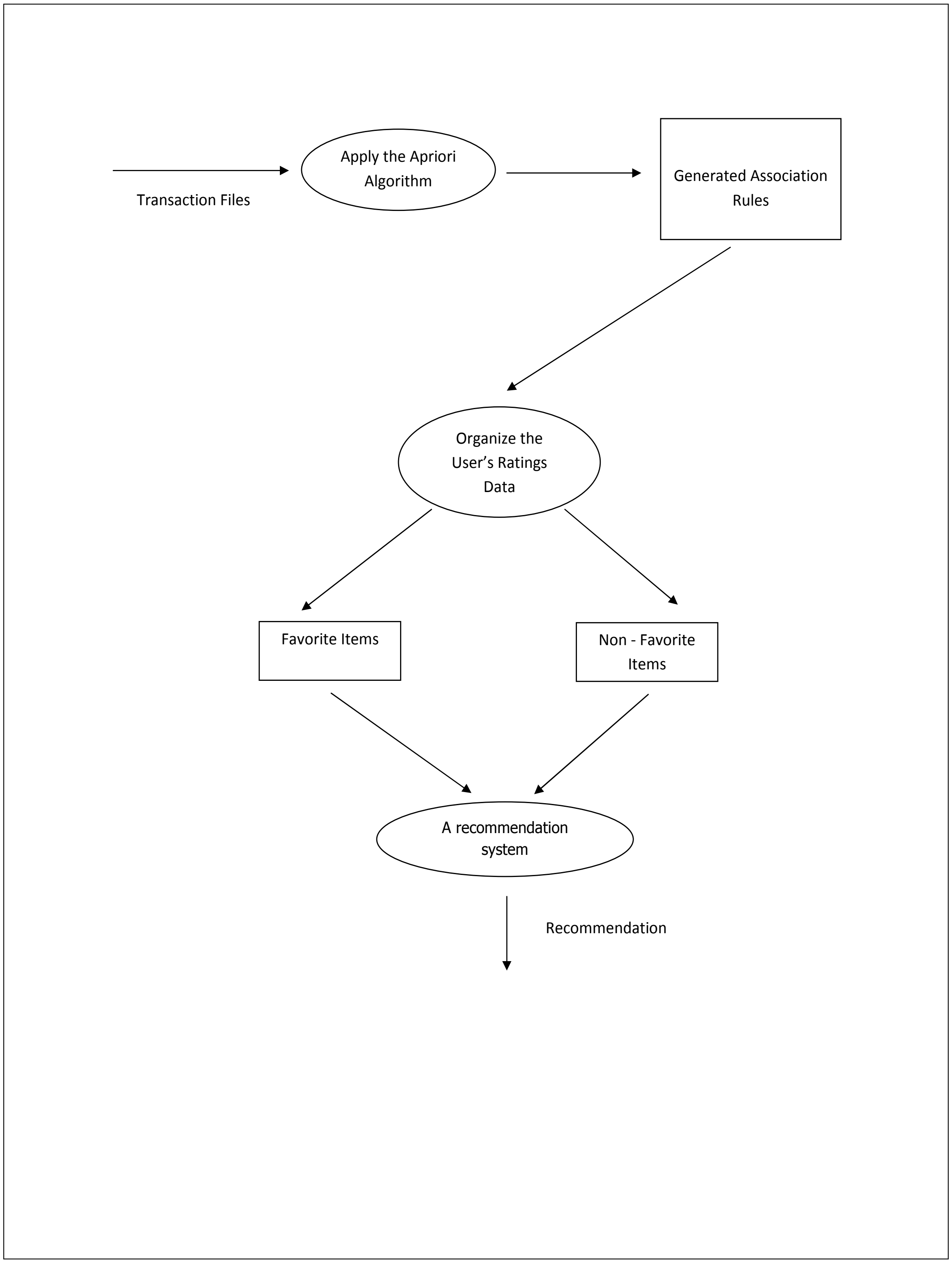


Figure 3.1: The diagram of the framework

3.2 A Hybrid Recommendation Framework

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:

Algorithm 1 The Proposed Recommendation Framework

```
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
```

3.2.1 Association Rule Generation

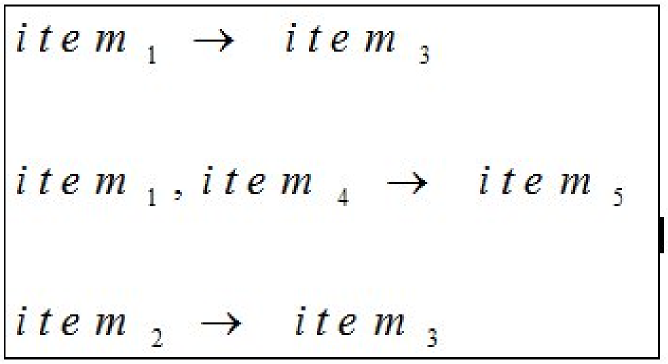
First of all, in order to apply our proposed algorithm, we first need to obtain the required association rules via the Apriori algorithm. The inputs of the Apriori algorithm are: the transactions file, minimum support, and minimum confidence. The transactions file in our context is basically the ratings matrix as shown in the Table 3.1:

Table 3.1: 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 Figure 3.2:



$item_1 \rightarrow item_3$

$item_1, item_4 \rightarrow item_5$

$item_2 \rightarrow item_3$

Figure 3.2: Example of strong association rules

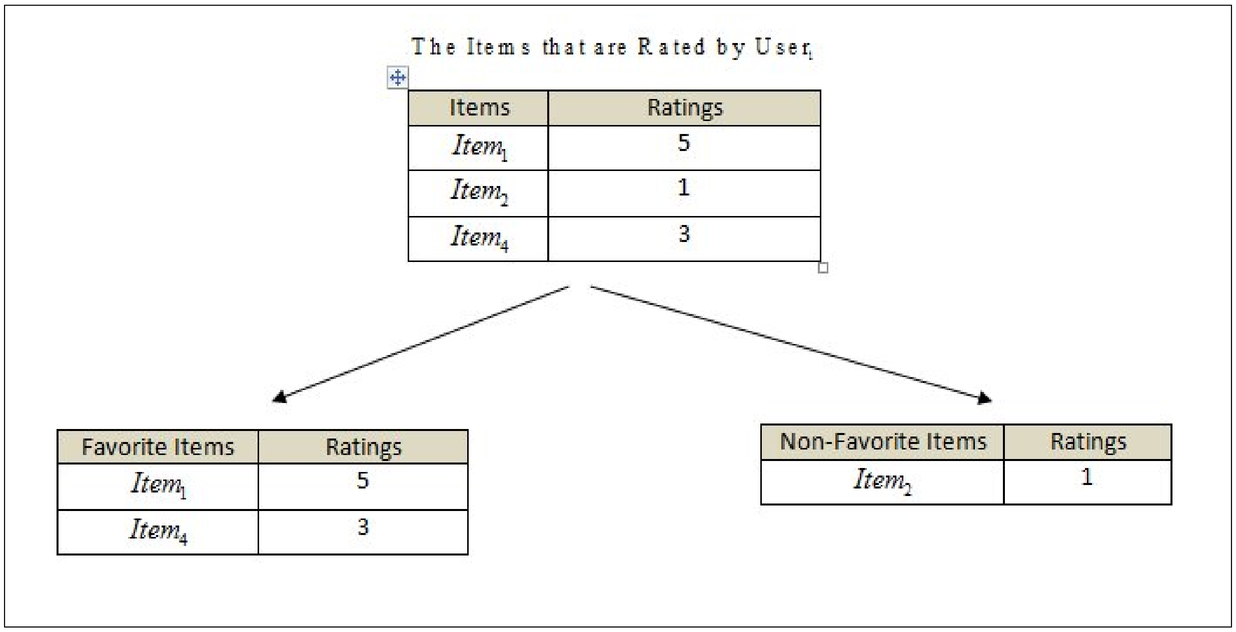
3.2.2 Favorite and Non-Favorite Item Distinction

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 can be obtained from original ratings matrix as shown in the Table 3.2:

Table 3.2: 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 Figures 3.3, 3.4, and 3.5 show how the data of each user is organized in the framework:

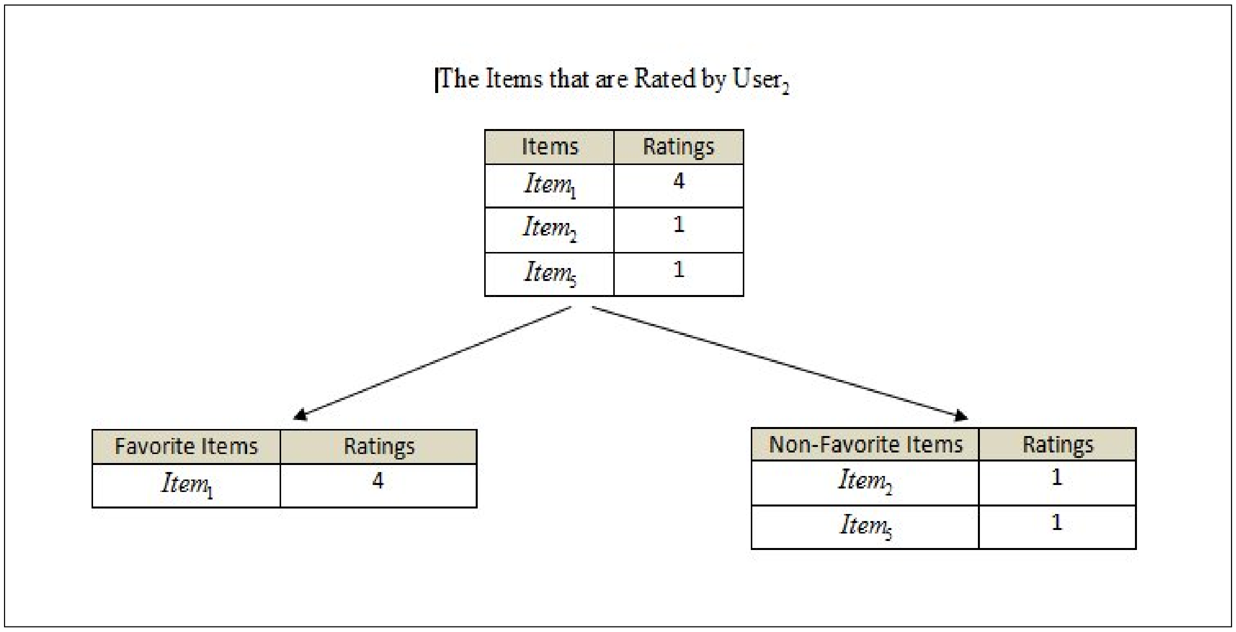

The Items that are Rated by $User_1$

Items	Ratings
$Item_1$	5
$Item_2$	1
$Item_4$	3

Favorite Items	Ratings
$Item_1$	5
$Item_4$	3

Non-Favorite Items	Ratings
$Item_2$	1

Figure 3.3: Rated items of the $user_1$

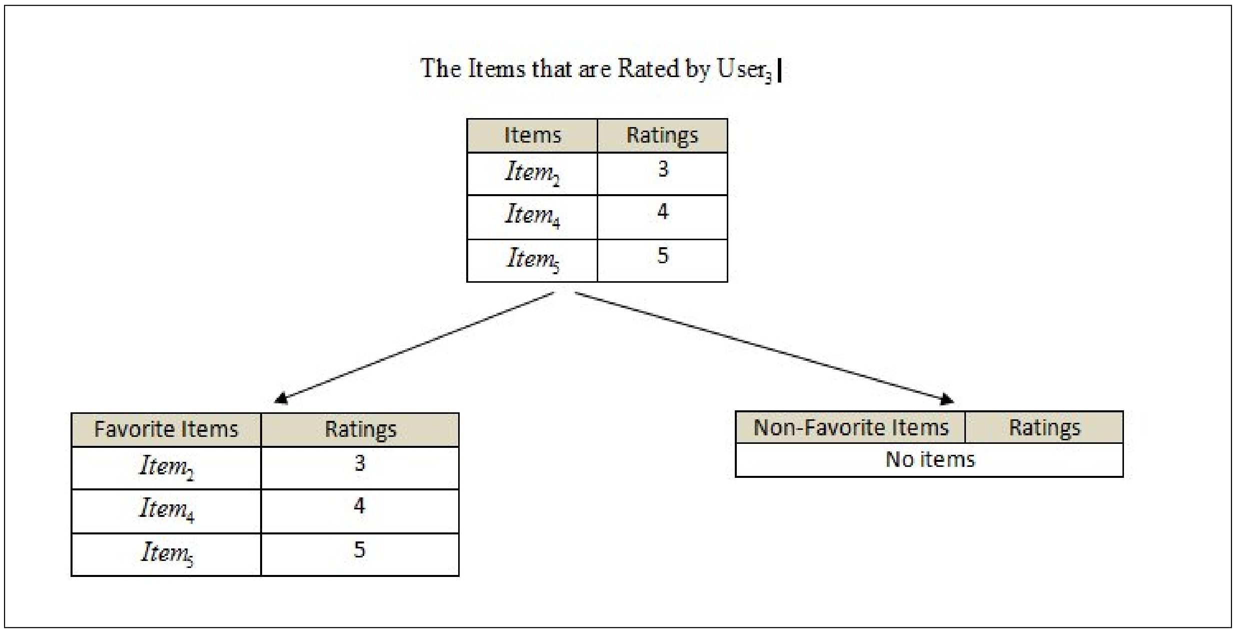

The Items that are Rated by $User_2$

Items	Ratings
$Item_1$	4
$Item_2$	1
$Item_5$	1

Favorite Items	Ratings
$Item_1$	4

Non-Favorite Items	Ratings
$Item_2$	1
$Item_5$	1

Figure 3.4: Rated items of the $user_2$



The Items that are Rated by $User_3$

Items	Ratings
$Item_2$	3
$Item_4$	4
$Item_5$	5

Favorite Items	Ratings
$Item_2$	3
$Item_4$	4
$Item_5$	5

Non-Favorite Items	Ratings
No items	

Figure 3.5: Rated items of the $user_3$

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 thus classified as a favorite item, and the $item_1$ is in the left hand side of the generated association rules as shown in the Figure 3.6:

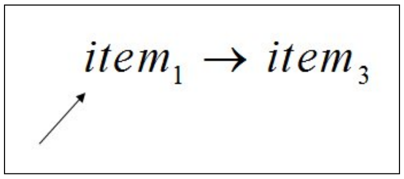


Figure 3.6: The $item_1$ appears in the left hand side of an association rule

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 $user_1$.

3.2.3 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. In other words, 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.

As we mentioned in the literature review chapter, there are different methods that are used to compute the similarity among items. For example, techniques from information retrieval such as the term frequency/inverse document frequency are used to specify the content (keywords) of an item. Specifying the content of items leads to compute the similarity among those items. In addition, the cosine of an angle and, the Jaccard coefficient are used to compute the similarity as well.

In our proposed framework, we use Jaccard coefficient to measure the similarity between two items [32]. The Jaccard coefficient is used to compute the similarity between two binary vectors, and it is defined in the following formula [21]:

$$Jaccard(i, j) = \frac{|S(i) \cap S(j)|}{|S(i) \cup S(j)|}\ , \tag{3.1}$$

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 [21]. Since the Jaccard coefficient is used to measure the similarity between two binary vectors, for simplicity, it can be illustrated in the following formula [22]:

$$Jaccard = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}\ , \tag{3.2}$$

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 [22].

3.3 Comparison of our Framework with other Systems

As we mentioned in the literature review chapter, Chellatamilan and Suresh presented an idea for building a recommendation system for the e-Learning system using Association Rules Mining to provide students with the best selection of learning materials and e-Learning resources [14]. Their idea is to gather data from students using a survey questionnaire in an area of educational background, IT experience, technology accessibility, frequency of their study patterns, demographics data, etc. In addition, the system analyzes students' logs of a Learning Management System (LMS) Moodle. Then, they apply data mining tools such as association rules to find frequent itemsets. Association rule mining, distance metrics such as Jaccard measure and cosine of the angle are used to construct the recommendation system [14]. This system is required to gather personal and background data from the users in the form of a survey questionnaire. This is a major step in this system, and it can be considered as a disadvantage of the proposed recommendation system. Recommendation systems that require gathering data such as demographic data work well only if the data is available [1]. Thus, failure to provide such data can cause poor recommendations [1]. Our proposed framework does not require gathering information from users, such as demographic information, in order to provide recommendations which is an advantage over the system proposed by Chellatamilan and Sures.

Bendakir and Aïmeur proposed a course recommendation system based on association rules mining [15]. The system incorporates a data mining process with user ratings in recommendations [15]. Specifically, the architecture of the system is divided into two phases: an off-line phase, which consists of a data mining process, and an on-line phase for the interaction of the systems with its users [15]. The off-line phase is used to extract association rules from the data, and the on-line phase uses the rules to infer course recommendations [15]. The advantage of this system is to allow the user (student) to evaluate the previous recommendations, so the system can be enhanced, and the rules are updated as more evaluations of the previous recommendations are provided by the students [15]. But, this system was developed to fit a certain context of recommendation systems which is a course recommendation. Our proposed framework can fit different contexts of recommendation systems such as $(Student \times Course)$, $(Tourist \times VacationPlace)$, or $(Person \times Restaurant)$.

Experiments

This chapter presents an experimental study of our proposed framework. The first section describes the experimental setup. The second section presents the experiment results. The last section summarizes our observation on the experiment results.

4.1 Experimental Setup

4.1.1 Dataset

We use the dataset of MovieLens, provided by GroupLens Research [12]. 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.

4.1.2 Hardware and Software

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

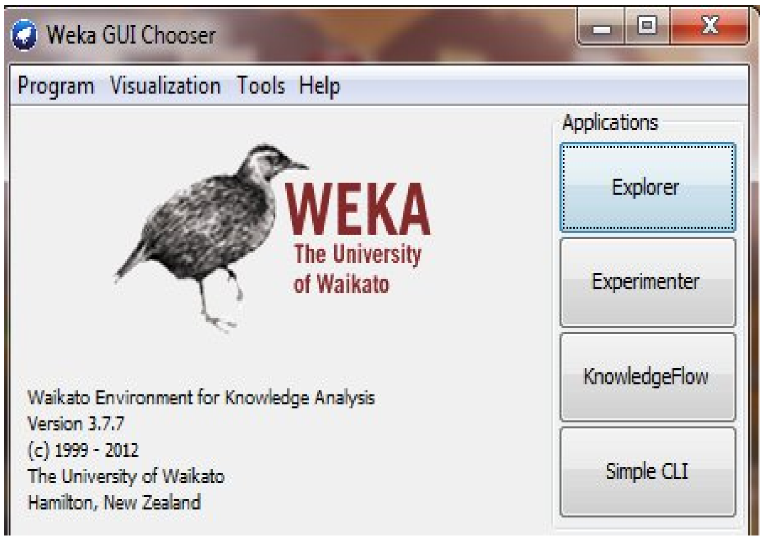
4.1.2.1 Hardware

- Processor Type: Intel Core i3 CPU
- Processor Speed: 2.53 GHz

- Available Ram: 4.00 GB

4.1.2.2 Software

In order to generate the association rules, we have used WEKA software [13]. WEKA software provides machine learning algorithms to implement several data mining tasks. It is open source software.



Waikato Environment for Knowledge Analysis
Version 3.7.7
(c) 1999 - 2012
The University of Waikato
Hamilton, New Zealand

Figure 4.1: The WEKA interface

Additionally, we used Java with Eclipse IDE [16] to implement our proposed algorithm, and to write several associated functions.

4.1.3 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 [30]. MAE can be computed by the following equation:

$$MAE = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N} \tag{4.1}$$

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 [19] state that the Root Mean Squared Error (RMSE) is perhaps the most popular metric used in evaluating accuracy of predicted ratings in recommendation systems [19]. It measures the quality of predicted ratings [31]. It can be computed as the following:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(p_i - r_i)^2}{N}} \tag{4.2}$$

4.2 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.

4.2.1 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 in order to run the Apriori algorithm and obtain the association rules. Table 4.1 shows the ratings matrix:

Table 4.1: 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

In the above matrix, 0 means, the $user_1$ has not yet ranked the $item_3$. 1 means the $user_1$ has ranked the $item_1$.

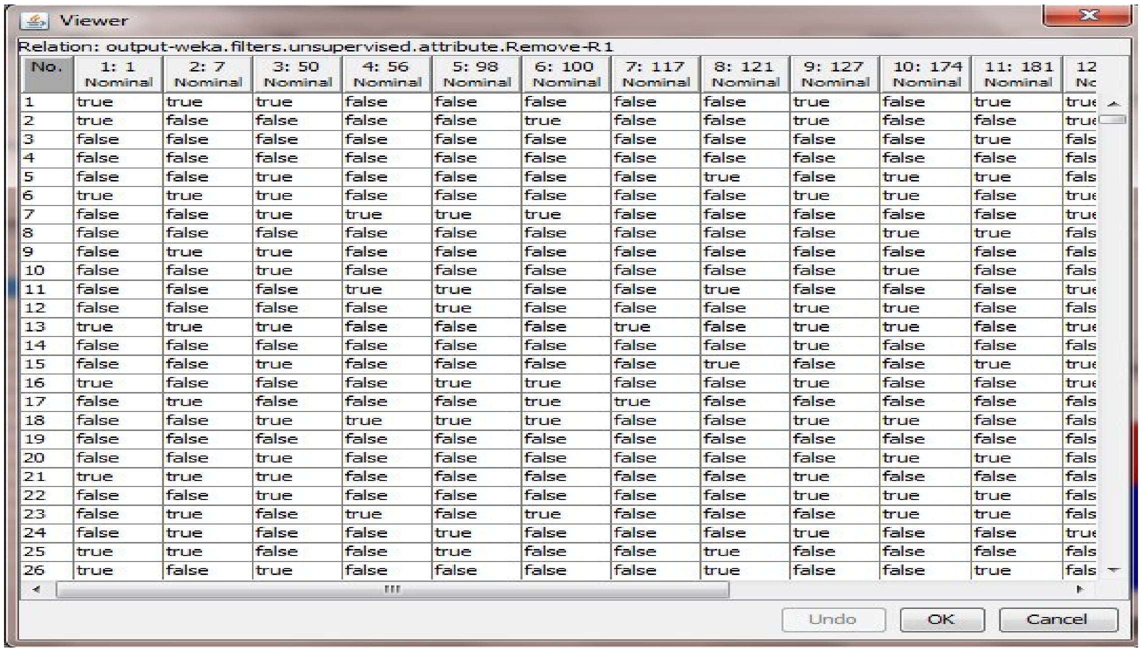
4.2.1.1 The Apriori Algorithm on the Entire Rating Matrix

To apply the Apriori algorithm on WEKA, we need to update the above binary rating matrix (Table 4.1) to a Boolean rating matrix, as shown in Table 4.2. $False$ means the movie has not rated yet, and $True$ means the movie has rated by the user.

Table 4.2: 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. Figure 4.2 shows how the ratings matrix is entered into WEKA:



Viewer

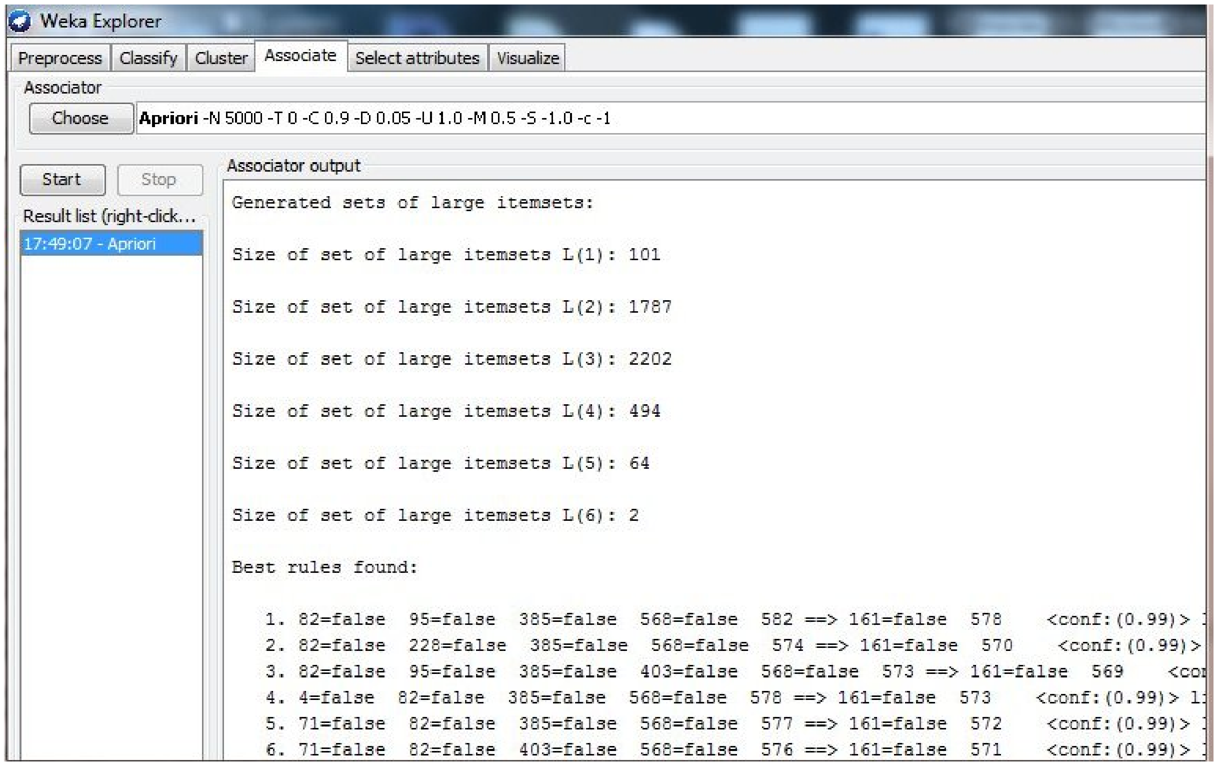
Relation: output-weka.filters.unsupervised.attribute.Remove-R1

No.	1: 1 Nominal	2: 7 Nominal	3: 50 Nominal	4: 56 Nominal	5: 98 Nominal	6: 100 Nominal	7: 117 Nominal	8: 121 Nominal	9: 127 Nominal	10: 174 Nominal	11: 181 Nominal	12 Nc
1	true	true	true	false	false	false	false	false	true	false	true	tru
2	true	false	false	false	false	true	false	false	true	false	false	tru
3	false	false	false	false	false	false	false	false	false	false	true	fals
4	false	false	false	false	false	false	false	false	false	false	false	fals
5	false	false	true	false	false	false	false	true	false	true	true	fals
6	true	true	true	false	false	false	false	false	true	true	false	tru
7	false	false	true	true	true	true	false	false	false	false	false	tru
8	false	false	false	false	false	false	false	false	false	true	true	fals
9	false	true	true	false	false	false	false	false	false	false	false	fals
10	false	false	true	false	false	false	false	false	false	true	false	fals
11	false	false	false	true	true	false	false	true	false	false	false	tru
12	false	false	false	false	true	false	false	false	true	true	false	fals
13	true	true	true	false	false	false	true	false	true	true	false	tru
14	false	false	false	false	false	false	false	false	true	false	false	fals
15	false	false	true	false	false	false	false	true	false	false	true	tru
16	true	false	false	false	true	true	false	false	true	false	false	tru
17	false	true	false	false	false	true	true	false	false	false	false	fals
18	false	false	true	true	true	true	false	false	true	true	false	fals
19	false	false	false	false	false	false	false	false	false	false	false	fals
20	false	false	true	false	false	false	false	false	false	true	true	fals
21	true	true	true	false	false	false	false	false	true	false	false	fals
22	false	false	true	false	false	false	false	false	true	true	true	fals
23	false	true	false	true	false	true	false	false	false	true	true	fals
24	false	true	false	false	true	false	false	false	true	false	false	tru
25	true	true	false	false	true	false	false	true	false	false	false	fals
26	true	false	true	false	false	false	false	true	false	false	true	fals

Figure 4.2: The rating matrix in WEKA

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 [9]. 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 as shown in the Figure 4.3:



```
Apriori -N 5000 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.5 -S -1.0 -c -1
```

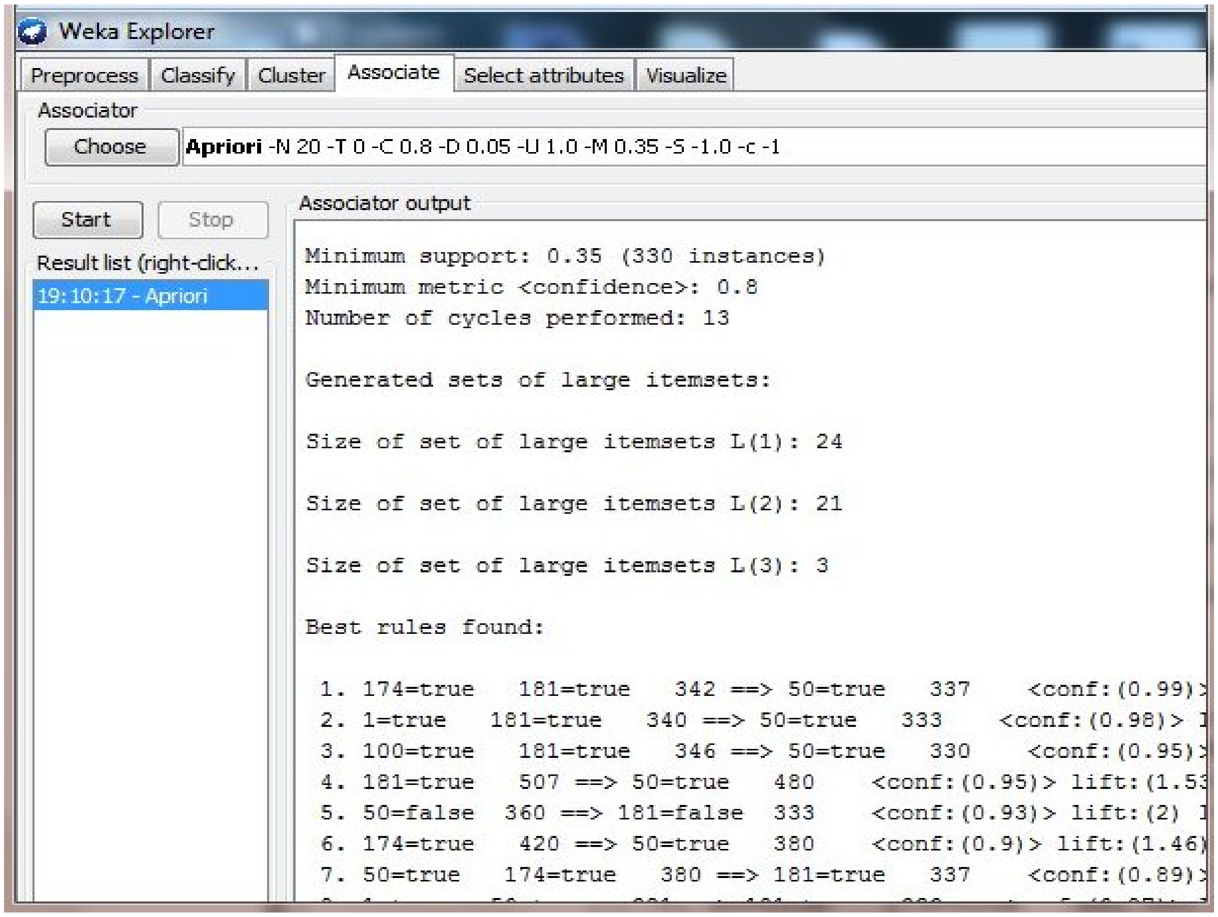
```
Generated sets of large itemsets:

Size of set of large itemsets L(1): 101

Size of set of large itemsets L(2): 1787

Size of set of large itemsets L(3): 2202

Size of set of large itemsets L(4): 494

Size of set of large itemsets L(5): 64

Size of set of large itemsets L(6): 2

Best rules found:

   1. 82=false  95=false  385=false  568=false  582 ==> 161=false  578    <conf:(0.99)>
   2. 82=false  228=false  385=false  568=false  574 ==> 161=false  570    <conf:(0.99)>
   3. 82=false  95=false  385=false  403=false  568=false  573 ==> 161=false  569    <co
   4. 4=false  82=false  385=false  568=false  578 ==> 161=false  573    <conf:(0.99)> l
   5. 71=false  82=false  385=false  568=false  577 ==> 161=false  572    <conf:(0.99)>
   6. 71=false  82=false  403=false  568=false  576 ==> 161=false  571    <conf:(0.99)>
```

Figure 4.3: Samples of generated association rules with 117 movies

From the Figure 4.3, we can interpret the rule number one as: If the *user* has not rated the *movie no. 82*, *movie no. 95*, *movie no. 385*, and *movie no. 568*, the *user* will not rate the *movie no. 161*. That means the user is less likely to watch it. This is irrelevant information, and it is not sufficient enough to use to provide accurate recommendations.

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. In order to provide accurate recommendations, a sufficient number of ratings should exist in the system [5].

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. The Figure 4.4 shows the association rules that were generated by the Apriori algorithm in WEKA with parameters of 35 % minimum support, 80 % confidence:



Apriori -N 20 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0.35 -S -1.0 -c -1

```
Minimum support: 0.35 (330 instances)
Minimum metric <confidence>: 0.8
Number of cycles performed: 13

Generated sets of large itemsets:

Size of set of large itemsets L(1): 24

Size of set of large itemsets L(2): 21

Size of set of large itemsets L(3): 3

Best rules found:

 1. 174=true   181=true   342 ==> 50=true   337    <conf:(0.99)>
 2. 1=true   181=true   340 ==> 50=true   333    <conf:(0.98)> l
 3. 100=true   181=true   346 ==> 50=true   330    <conf:(0.95)>
 4. 181=true   507 ==> 50=true   480    <conf:(0.95)> lift:(1.53
 5. 50=false  360 ==> 181=false  333    <conf:(0.93)> lift:(2) l
 6. 174=true   420 ==> 50=true   380    <conf:(0.9)> lift:(1.46)
 7. 50=true   174=true   380 ==> 181=true   337    <conf:(0.89)>
```

Figure 4.4: Samples of generated association rules with 12 movies

4.2.1.2 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 4.3 shows the results of the computed- predicted ratings in decimal form:

Table 4.3: Accuracy measured in terms of MAE and RMSE

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

The Table 4.4 shows the results after applying the floor function to the computed-predicted ratings:

Table 4.4: Accuracy measured in terms of MAE and RMSE (after applying the floor function)

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

The Table 4.5 shows the results after applying the ceiling function to the

computed-predicted ratings:

Table 4.5: Accuracy measured in terms of MAE and RMSE (after applying the ceiling function)

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

Now, we can summarize the results on the following table and chart:

Table 4.6: Evaluation of experiment's results

$Evaluation$	MAE	$RMSE$
$In\ Decimal\ Numbers$	0.6865502804	0.8786227744
$After\ Applying\ Floor\ Function$	0.892116171	1.10688392
$After\ Applying\ Ceiling\ Function$	0.806277217	1.186032072

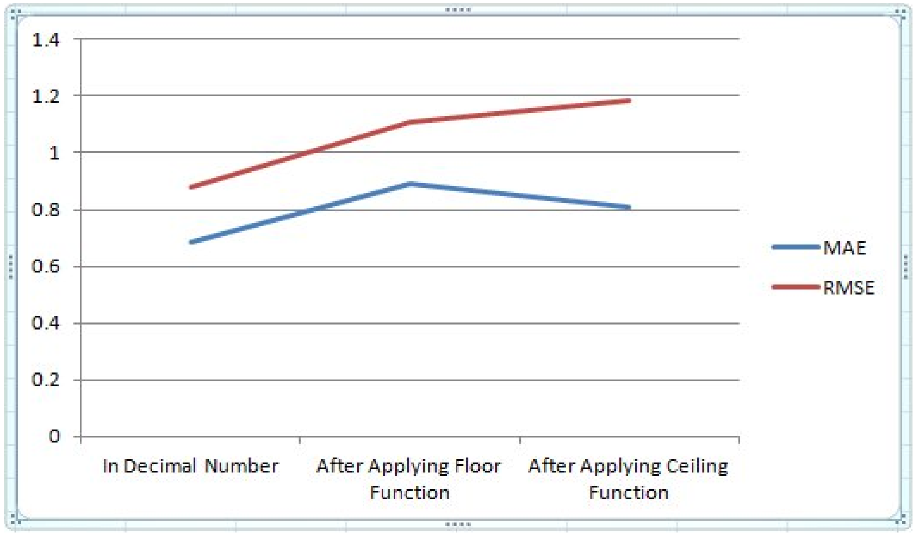


Figure 4.5: The results of the evaluation of the three cases

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

4.2.2 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 4.7 shows how we represent a movie based on its genre:

Table 4.7: The representation of a movie

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

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

4.2.2.1 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 show the results after applying the Jaccard coefficient with similarity of 50 % and up among the movies:

Table 4.8: Accuracy measured in terms of MAE and RMSE

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

Table 4.9: Accuracy measured in terms of MAE and RMSE (after applying the floor function)

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

Table 4.10: Accuracy measured in terms of MAE and RMSE (after applying the ceiling function)

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

Below, Table 4.11 provides the summary of the results of the evaluation of the *Non-Favorite Items* part:

Table 4.11: Summary of experiment's results of Non-Favorite items

$Evaluation$	MAE	$RMSE$
$In\ Decimal\ Numbers$	0.871955461	1.104979437
$After\ Applying\ Floor\ Function$	1.161398927	1.474234682
$After\ Applying\ Ceiling\ Function$	1.304981189	1.632056341

The results of the experiment on *Non-Favorite Items* from the table and chart above show the predicted ratings in decimal form, which provides more accurate predicted ratings than the other floor and ceiling functions.

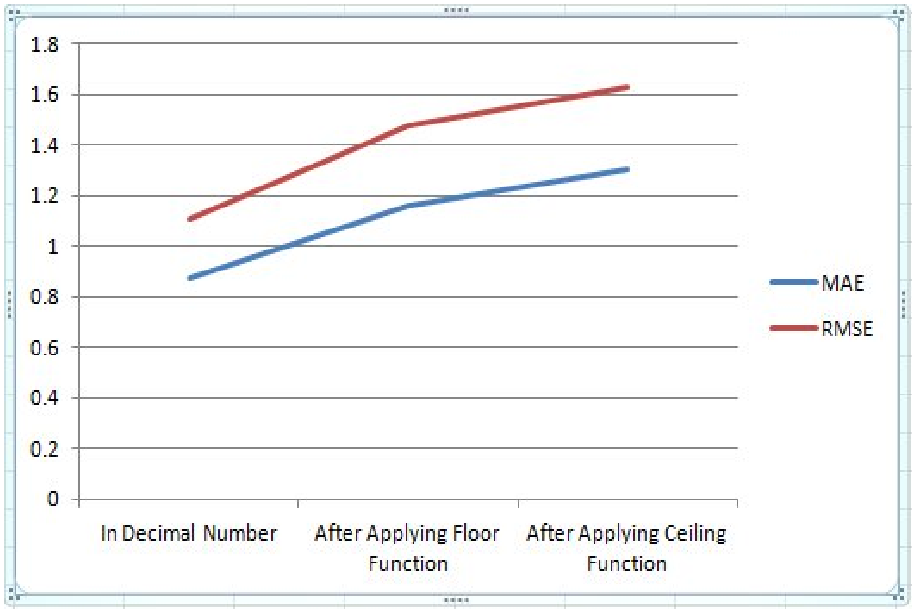


Figure 4.6: Experiment's results of Non-Favorite items

4.3 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.

Conclusion and Future Work

5.1 Conclusion

Our thesis 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.

5.2 Future Work

In the first part of our proposed framework, we have to run the Apriori algorithm on the rating matrix $(User \times Item)$ which in most recommendation systems is a sparse matrix. Running the Apriori algorithm on a sparse matrix can produce many irrelevant association rules. The total number of ratings is important in the recommendation systems to provide accurate recommendations to users. Thus, with a limited number of ratings, the rating matrix $(User \times Item)$ is considered a sparse matrix. Our future work is to find a certain technique to address the sparsity from the rating matrix. Reducing the sparsity gives us a less sparse rating matrix which can provide us with relevant association rules when we apply the Apriori algorithm on it.

BIBLIOGRAPHY

[1] 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.

[2] E. Aïmeur, G. Brassard, J.M. Fernandez, and F.S.M. Onana, Privacy-preserving demographic filtering, Proceedings of the 2006 ACM symposium on Applied computing - SAC 06, 2006, p.872.

[3] S.S., Anand, and B., Mobasher Intelligent Techniques for Web Personalization, IJCAI 2006, Springer, 2006, pp. 1-37.

[4] 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.

[5] 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.

[6] P.,Lops, M.,Gemmis, and G.,Semeraro (2011). Content-based recommender systems: State of the art and trends. Recommender Systems Handbook, 73-105.

[7] 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.

[8] X.,Su, and T. M.,Khoshgoftaar (2009). A survey of collaborative filtering techniques. Advances in Artificial Intelligence, 2009, 4.

[9] J.,Han, and M.,Kamber (2006). Data mining: concepts and techniques (2nd ed.). Amsterdam: Elsevier .

[10] 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.

[11] Z.,Xizheng (2007, July). Building personalized recommendation system in E-commerce using association rule-based mining and classification. In Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on (Vol. 3, pp. 853-857). IEEE.

[12] MovieLens Data Sets. (2011, August 8). GroupLens Research. Retrieved November 18, 2012, from http://www.grouplens.org/node/73

[13] 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

[14] T.,Chellatamilan, and R.,SURESH (2011). An e-Learning Recommendation System using Association Rule Mining Technique. European Journal of Scientific Research Vol. 64 No, 2, 330-339.

[15] N.,Bendakir, and E.,Aïmeur (2006, July). Using association rules for course recommendation. In Proceedings of the AAAI Workshop on Educational Data Mining (pp. 31-40).

[16] About the Eclipse Foundation. (n.d.). Eclipse. Retrieved November 18, 2012, from http://www.eclipse.org/

[17] R.,Burke (2002). Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4), 331-370.

[18] B.,Krulwich (1997). Lifestyle finder: Intelligent user profiling using large-scale demographic data. AI magazine, 18(2), 37.

[19] G.,Shani, and A.,Gunawardana (2011). Evaluating recommendation systems. Recommender Systems Handbook, 257-297.

[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] 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.

[22] P.,Tan, M.,Steinbach, and V.,Kumar (2005). Introduction to data mining. Boston: Pearson Addison Wesley.

[23] J. L.,Herlocker, J. A.,Konstan, A.,Borchers, and J.,Riedl (1999, August). An algorithmic framework for performing collaborative filtering. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (pp. 230-237). ACM.

[24] M.,Hegland (2007). The apriori algorithma tutorial. Mathematics and Computation in Imaging Science and Information Processing, 11, 209-262.

[25] J.,Han, J.,Pei, and Y.,Yin (2000, May). Mining frequent patterns without candidate generation. In ACM SIGMOD Record (Vol. 29, No. 2, pp. 1-12). ACM.

[26] R.,Perego, S.,Orlando, and P.,Palmerini (2001). Enhancing the apriori algorithm for frequent set counting. Data Warehousing and Knowledge Discovery, 71-82.

[27] M.,Balabanovi, and Y.,Shoham (1997). Fab: content-based, collaborative recommendation. Communications of the ACM, 40(3), 66-72.

[28] W.,Pan, E. W.,Xiang, N. N.,Liu, and Q.,Yang (2010, March). Transfer learning in collaborative filtering for sparsity reduction. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (pp. 230-235).

[29] Y. Z.,Wei, L.,Moreau, and N. R.,Jennings (2005). A market-based approach to recommender systems. ACM Transactions on Information Systems (TOIS), 23(3), 227-266.

[30] 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.

[31] 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.

[32] 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.

[33] 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).

[34] 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.

[35] C.,Christakou, S.,Vrettos, and A.,Stafylopatis (2007). A hybrid movie recommender system based on neural networks. International Journal on Artificial Intelligence Tools, 16(05), 771-792.

[36] S.,Debnath, N.,Ganguly, and P.,Mitra (2008, April). Feature weighting in content based recommendation system using social network analysis. In Proceedings of the 17th international conference on World Wide Web (pp. 1041-1042). ACM.

[37] M. J. ,Pazzani (1999). A framework for collaborative, content-based and demographic filtering. Artificial Intelligence Review, 13(5-6), 393-408.

[38] I.,Zukerman, and D. W.,Albrecht (2001). Predictive statistical models for user modeling. User Modeling and User-Adapted Interaction, 11(1-2), 5-18.

[39] G.,Linden, B.,Smith, and J.,York (2003). Amazon. com recommendations: Item-to-item collaborative filtering. Internet Computing, IEEE, 7(1), 76-80.

[40] B.,Mobasher, X.,Jin, and Y.,Zhou (2004). Semantically enhanced collaborative filtering on the web. In Web Mining: From Web to Semantic Web (pp. 57-76). Springer Berlin Heidelberg.