Recommender Systems using Collaborative Filtering

D Yogendra Rao

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

Recommender Systems using Collaborative Filtering

Dissertation submitted in

11th May 2015 to the

department of

Computer Science and Engineering

of

National Institute of Technology Rourkela

in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

by

D Yogendra Rao

(Roll 111CS0152) *under the supervision*

of

Prof. Banshidhar Majhi

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela – 769 008, India

Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, India.
www.nitrkl.ac.in

Prof. Banshidhar Majhi
Professor

May 11 , 2015

Certificate

This is to certify that the work in the thesis entitled Recommender Systems using Collaborative Filtering by D Yogendra Rao, bearing roll number 111CS0152, is a record of his work carried out under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology* in *Computer Science and Engineering*.

Banshidhar Majhi

Abstract

With the vast amount of data that the world has nowadays, institutions are looking for more and more accurate ways of using this data. Companies like Amazon use their huge amounts of data to give recommendations for users. Based on similarities among items, systems can give predictions for a new items rating. Recommender systems use the user, item, and ratings information to predict how other users will like a particular item.

Recommender systems are now pervasive and seek to make profit out of customers or successfully meet their needs. However, to reach this goal, systems need to parse a lot of data and collect information, sometimes from different resources, and predict how the user will like the product or item. The computation power needed is considerable. Also, companies try to avoid flooding customer mailboxes with hundreds of products each morning, thus they are looking for one email or text that will make the customer look and act.

The motivation for this project comes from the eagerness to get a deep understanding of recommender systems. One of the goals set for this project was to apply machine learning dynamically and to verify the results. Thus, a large dataset is used to test the algorithm and to compare each algorithm in terms of error rate. In this project, a website has been developed that uses different techniques for recommendations namely User-based Collaborative Filtering, Item-Based Collaborative Filtering and Model Based Collaborative Filtering. Every technique has its way of predicting the user rating for a new item based on existing users data. To evaluate each method, I have used Movie Lens, an external data set of users, items, and ratings, and calculated the error rate using Mean Absolute Error Rate (MAE) and Root Mean Squared Error (RMSE)

Table of Contents

List of Figures

List of Tables

Introduction

The way in which people now a days search for information, products and even other people is changing with the advent of Recommender Systems. Recommender Systems study patterns of behavior to predict what someone may prefer from among a collection of items that he/she has never experienced. The technology behind the Recommender systems has evolved over the past 15 years into a rich collection of tools that now enables the researcher or users or practitioner to develop effective Recommender Systems. Recommender systems are now pervasive in consumers lives[1]. They help users to find items/products that they would consider or would like to buy on the basis of huge amounts of collected data. Websites like Facebook, Netflix, Last.fm and other social networking and commercial websites are using these systems.

Chapter 1

Introduction

1.1 Introduction to Recommender Systems

The way in which people now a days search for information, products and even other people is changing with the advent of Recommender Systems. Recommender Systems study patterns of behavior to predict what someone may prefer from among a collection of items that he/she has never experienced. The technology behind the Recommender systems has evolved over the past 15 years into a rich collection of tools that now enables the researcher or users or practitioner to develop effective Recommender Systems. Recommender systems are now pervasive in consumers lives[1]. They help users to find items/products that they would consider or would like to buy on the basis of huge amounts of collected data. Websites like Facebook, Netflix, Last.fm and other social networking and commercial websites are using these systems.

Working on a huge data set to predict a users similarity with other users or his/her ratings is the core objective of any recommender system. There are various approaches to implementing a recommender system such as Content-based Filtering, Collaborative Filtering and Hybrid filtering. The goal of this project is to apply a collaborative filtering algorithm in a website that can collect various users information from the user, such as Name, Email id, his location, gender, Movies Information and the rating for movies by that user.

There are many algorithms that could be applied on data to predict a user preference. Some of the algorithms of predicting a user preference are User-based

Collaborative Filtering, Item-based Collaborative Filtering, and Model-based Collaborative Filtering methods are ways of predicting a user preference[1].The number of users, items, or clusters in each one respectively will determine the function performance[1]. However, the most well-known and common one is User-based Collaborative Filtering[1]. In this algorithm, we predict an items rate for a user by collecting information about this user and similar users[1].

1.2 Important Concepts, Notations

We need several concepts of Collaborative filtering techniques to describe our problem domain and also to analyze the system requirements. Other recommender methods also share these concepts.

The users who have rated the various items in our dataset serves as the information domain for a collaborative filtering system. A rating or preference that is expressed by a user for an item/movie is called a rating and is often represented in (User, Item, Rating) triple format. The format of these rating can change according to the system in question. One can find systems using an integer-valued rating such as 0-5 stars or real-valued rating scales, while some systems use binary (like/dislike) or ternary scales. Unary ratings, such as has purchased, are particularly common in e-commerce deployments as they express well the users purchasing history absent ratings data.

A sparse matrix is formed using this set of all rating triples referred to as the rating matrix. The (User, Item) pairs where the user has not rated the item are unknown values in this matrix.

We commonly focus on two tasks when we describing the use and evaluation of recommender systems, including collaborative filtering systems, first being the prediction task: given a user and an item/movie, what is the users likely preference for the item? The prediction task for a recommender system can be equivalent to a matrix missing values problems if the ratings matrix is viewed as a sampling of values from a useritem p rating matrix.

The second task is the recommend task. In this task, given a user, our recommender system has to produce the best ranked list of n items for the new test case user. An n-item recommendation list does not guarantee that it will contain the

Sample Data		
	Snow Crash Girl	Girl with Dragon Tattoo
Amy	5	5
Bill	2	5
Jim	1	4

Table 1.1. Sample Data

n items with the highest predicted rating, as predicted rating may not be the only criteria used to produce the recommendation list.

In this project, we are using a consistent mathematical notation for referencing various elements for our recommender system model. Following are the notations that are being used in the report:

1. U - set of users
2. I - set of items
3. Iu - is the set of items that has been rated by user u
4. Ui - is the set of users who have rated the movie i.
5. R - The rating matrix
6. r(u,i) - being the rating user u provided for item i
7. ru - all the movies rated by user u
8. ri - Vector of all ratings provided for item i
9. pi(u,i) - elements of the user-item preference matrix pi.
10. p(u,I) - Recommenders prediction of pi(u,i)

Chapter 2

Collaborative Filtering

2.1 Introduction

Collaborative filtering (CF) is a popular recommendation algorithm that bases its predictions and recommendations on the ratings or behavior of other users in the system. The fundamental assumption behind this method is that other users opinions can be selected and aggregated in such a way as to provide a reasonable prediction of the active users preference. Intuitively, they assume that, if users agree about the quality or relevance of some items, then they will likely agree about other items if a group of users likes the same things as Mary, then Mary is likely to like the things they like which she hasnt yet seen.

There are other methods for performing recommendation, such as finding items similar to the items liked by a user using textual similarity in metadata (content-based filtering or CBF). The focus of this survey is on collaborative filtering methods, although content-based filtering will enter our discussion at times when it is relevant to overcoming a particular recommender system difficulty[4].

The majority of collaborative filtering algorithms in service today, including all algorithms detailed in this section, operate by first generating predictions of the users preference and then produce their recommendations by ranking candidate items by predicted preferences. Often this prediction is onthe same scale as the ratings provided by users, but occasionally the prediction is on a different scale and is meaningful only for candidate ranking. This strategy is analagous to the common information retrieval method of producing relevance scores for each document in a corpus with respect to a particular query and presenting the top-scored items. Indeed, the recommend task can be viewed as an information retrieval problem in which the domain of items (the corpus) is queried with the users preference profile.

Therefore, this section is primarily concerned with how various algorithms predict user preference. In later sections we will discuss recommendation strategies that diverge from this structure, but in actual implementation they frequently start with a preference-ranked list of items and adjust the final recommendation list based on additional criteria[4].

The traditional collaborative filtering algorithms include User-based, Item-based, and Model-based methods[1]. To explain how these methods works we are going to use the following notations. Let U be a set of N users and I a set of M items. Vui denotes the rating of user u U on item i I, and S I stands for the set of items that user u has rated[4]

2.2 Calculating User Similarity

The selecting of similarity function plays a critical design decision while implementing a user user CF.

2.2.1 Pearson Coefficient Correlation

This method computes the statistical correlation (also known as Pearsons Coefficient) between the common ratings of two user's to determine their similarity.

$$\bar{v} = \frac{\sum_{i \in S_u} v_{ui}}{|S_u|}$$

Figure 2.1. Calculation of Mean

Firstly, we can calculate the mean rate of a user using the formula given above.

$$w(a,u) = \frac{\sum_{i \in S_a \cap S_u} (v_{ai} - \bar{v}_a)(v_{ui} - \bar{v}_u)}{\sqrt{\sum_{i \in S_a \cap S_u} (v_{ai} - \bar{v}_a)^2 \sum_{i \in S_a \cap S_u} (v_{ui} - \bar{v}_u)^2}}$$

Figure 2.2. Pearson Coefficient

A better approximation to the above equation is given below :

$$w(a,u) = \frac{\sum v_{ai}v_{ui} - \frac{\sum v_{ai} \sum v_{ui}}{n}}{\sqrt{\sum v_{ai}^2 - \frac{\sum v_{ai}^2}{n}}\sqrt{\sum v_{ui}^2 - \frac{\sum v_{ui}^2}{n}}} \tag{2.1}$$

The approximation is better as the value can be found in one pass thus reducing the computational time.

2.2.2 Cosine similarity

Cosine similarity ignores 0-0 matches. It is defined as :

$$cos(x,y) = \frac{x.y}{\|x\| \times \|y\|} \tag{2.2}$$

where indicates the dot product and ——x—— indicates the length of the vector x. The length of a vector is

$$\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2} \tag{2.3}$$

2.3 User Based Collaborative Filtering

It is a memory-based algorithm that tries to mimics the daily word-of-mouth experience by analyzing the rating data from many users. We assume that the users with similar preferences are most likely to rate the items similarly. Thus we predict the missing ratings for a user by first finding a the nearest neighbor or similar users and then aggregating the ratings of these neighbor users to form the prediction.

The similarity is found using the Pearson Coefficient :

$$w(a,u) = \frac{\sum_{i \in S_a \cap S_u}(v_{ai} - \bar{v}_a)(v_{ui} - \bar{v}_u)}{\sqrt{\sum_{i \in S_a \cap S_u}(v_{ai} - \bar{v}_a)^2 \sum_{i \in S_a \cap S_u}(v_{ui} - \bar{v}_u)^2}} \tag{2.4}$$

and the predicted value is found using the equation :

$$p_{ai} = \bar{v}_a + \frac{\sum_{u \in U | i \in S_u} w(a,u) \times (v_{ui} - \bar{v}_u))}{\sum_{u \in U | i \in S_u} |w(a,u)|} \tag{2.5}$$

2.4 Item-Based Collaborative Filtering

When we were analyzing the recommender system that Amazon used, it was observed that the traditional collaborative filtering algorithm was not being used. Amazon recommender system doesn't use the User-based and Cluster models due to many reasons. One of the reasons is the expensive computation time O(MN), where M is the number of similar users and N is the number of common items among those users so the company decided not to use these methods[5]. To solve this computation problem, amazon came up with using clusters to reduce the number of items and users. However, we can find a reduction in the quality of recommendations[5]. In other words, the similarity may not be accurate, as this algorithm will compare the user to a small set of users. Also, partitioning items to item-space will limit the recommendations to specific types of products. Additionally, if the cluster does not include the popular or unpopular items, they will never be recommended to users[1].

However, in the website that has been developed as part of this project, I am applying Item-based Collaborative Filtering to display similar items:

$$w(a,u) = \frac{\sum_{i \in S_a \cap S_u} (v_{ai} - \bar{v}_a)(v_{ui} - \bar{v}_u)}{\sqrt{\sum_{i \in S_a \cap S_u} (v_{ai} - \bar{v}_a)^2 \sum_{i \in S_a \cap S_u} (v_{ui} - \bar{v}_u)^2}} \tag{2.6}$$

2.5 Model Based Collaborative Filtering

This method of Collaborative Filtering uses an unsupervised learning algorithm to divide the data set and it then classifies the users to clusters on the basis of a similarity metric. There are chances that a large number of clusters get made in the process, so the recommender system uses different ways to create small clusters of users. Initially there is only one user in the cluster, after which users are added to the clusters on the basis of the similarity metric repeatedly. The user can be classified to more than one cluster based on the similarity metric as the vectors that are created by the system may match with the other user vector[5].

Chapter 3

Implementation : Building of the Recommender Engine

The chapter deals with the building of the recommender engine with the implementation of Collaborative Filtering. Python was used to build the engine.

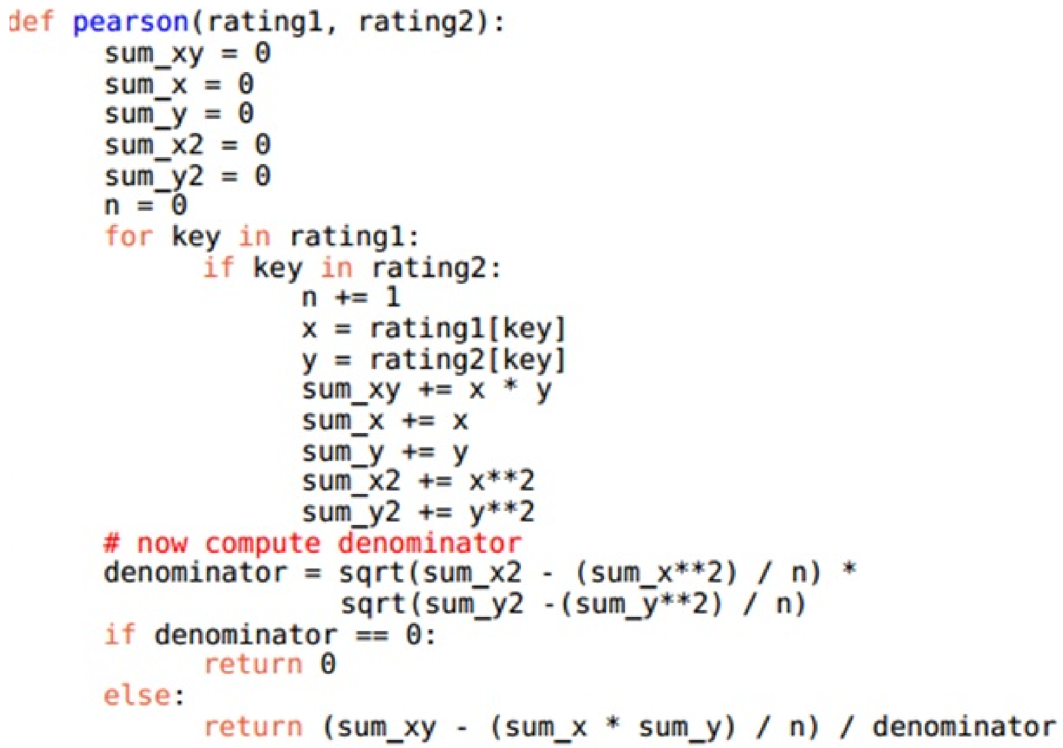
3.1 Calculation of Pearson Coefficient

First step of building the engine was to find the way to calculate the Pearson Coefficient.

We calculate the value of Pearson Coefficient using the approximate formula that we have mentioned in Chapter 2. The Pearson Correlation Coefficient is a measure of correlation between two variables. It ranges between -1 and 1 inclusive. 1 indicates perfect agreement. -1 indicates perfect disagreement.

The approximation formula was :

$$w(a,u) = \frac{\sum v_{ai}v_{ui} - \frac{\sum v_{ai} \sum v_{ui}}{n}}{\sqrt{\sum v_{ai}^2 - \frac{\sum v_{ai}{}^2}{n}}\sqrt{\sum v_{ui}^2 - \frac{\sum v_{ui}{}^2}{n}}} \tag{3.1}$$



```
def pearson(rating1, rating2):
        sum_xy = 0
        sum_x = 0
        sum_y = 0
        sum_x2 = 0
        sum_y2 = 0
        n = 0
        for key in rating1:
                if key in rating2:
                        n += 1
                        x = rating1[key]
                        y = rating2[key]
                        sum_xy += x * y
                        sum_x += x
                        sum_y += y
                        sum_x2 += x**2
                        sum_y2 += y**2
        # now compute denominator
        denominator = sqrt(sum_x2 - (sum_x**2) / n) *
                      sqrt(sum_y2 -(sum_y**2) / n)
        if denominator == 0:
                return 0
        else:
                return (sum_xy - (sum_x * sum_y) / n) / denominator
```

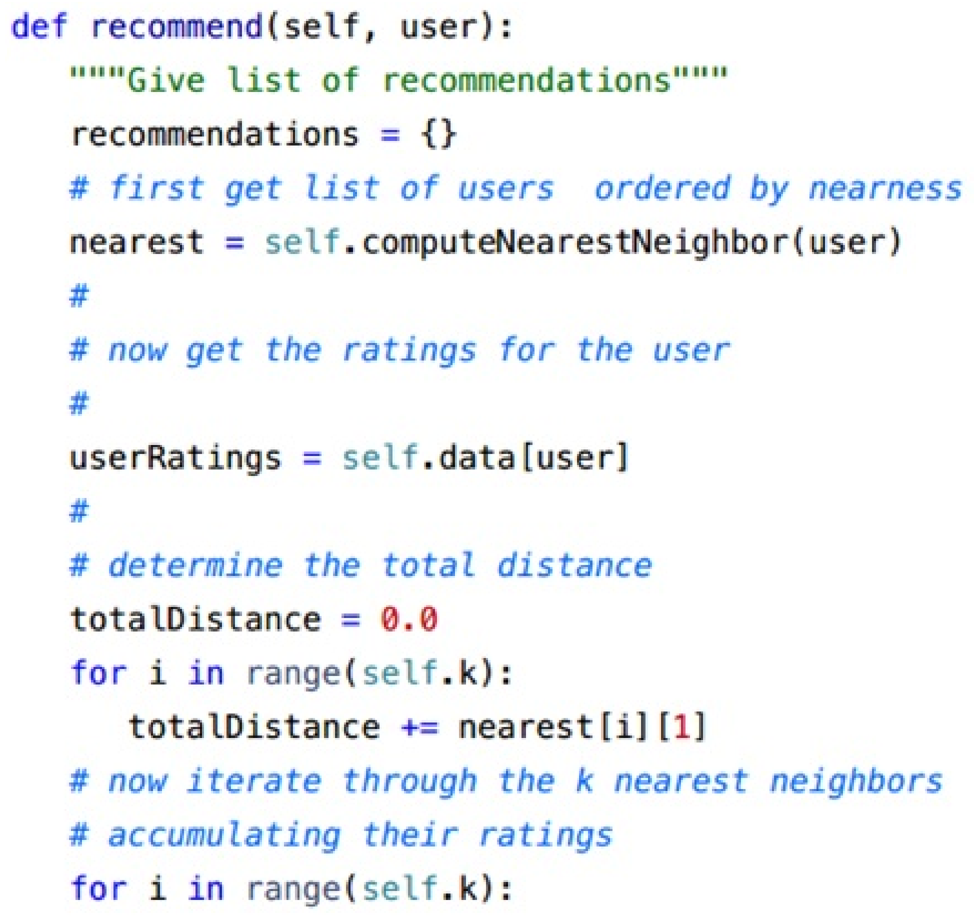
Figure 3.1. Implementation of Pearson Coefficient Calculation

3.2 Recommending Users

The system initially starts with calculation of distance between 2 users that are sent to the function as parameters. The movies and their rating by these users are stored as associative array. Associative arrays are like dictionaries which has a set of ¡key-value¿ values. So the 2 arrays are sent to the function. Once the distance is calculated using Pearson Coefficient, the nearest neighbors are found. The nearest neighbors are the one who most likely rate the movies in the same manner as that of the current user. The list is sent as an associative array. The value is collected by JSON which later parses the value so that the same value can be used by PHP to display the values in the website.

3.3 Dataset

The dataset used to check the efficiency of our application was the MovieLens dataset. It is a 3 column movie file: (user, movie, rating). The MovieLens Dataset



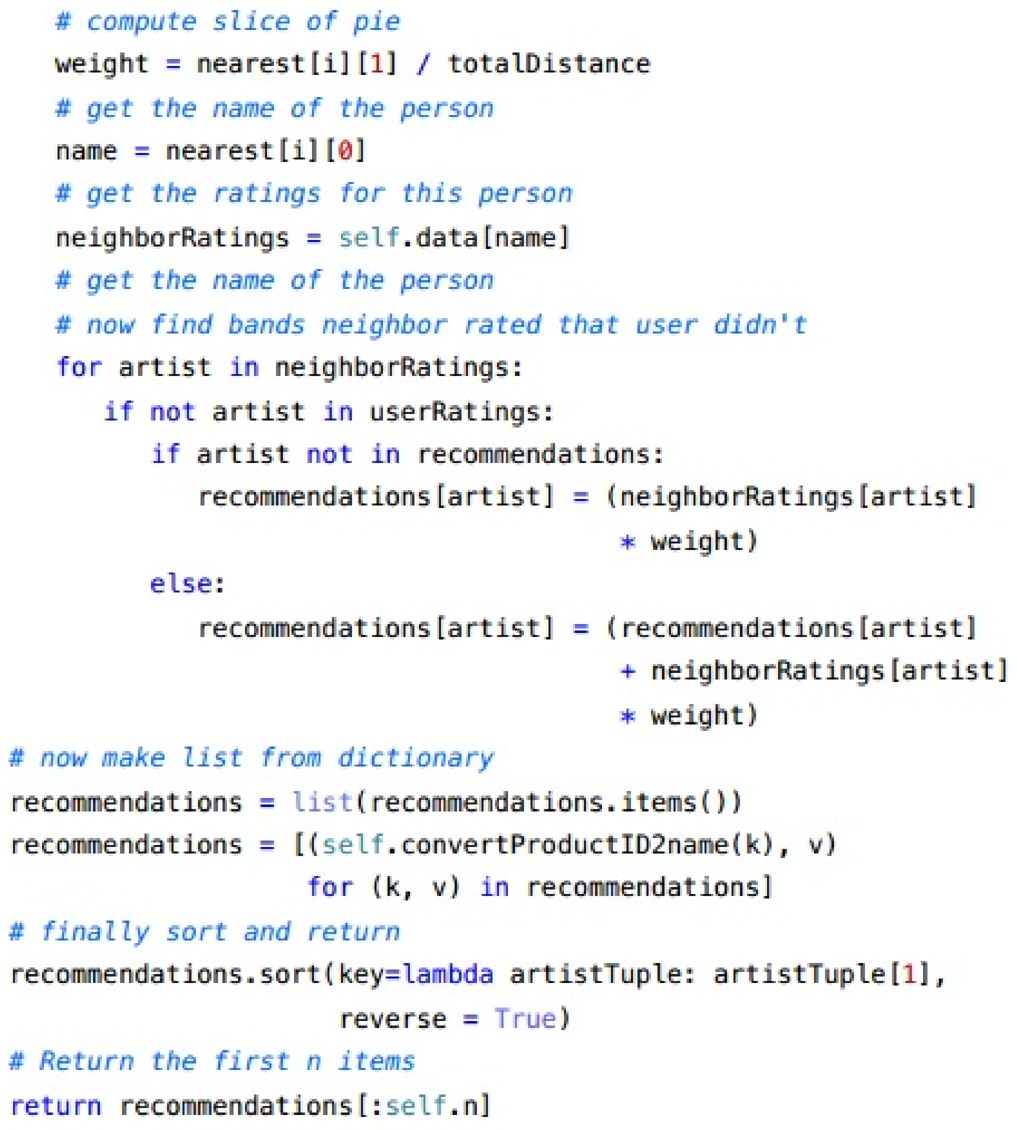
```
def recommend(self, user):
    """Give list of recommendations"""
    recommendations = {}
    # first get list of users  ordered by nearness
    nearest = self.computeNearestNeighbor(user)
    #
    # now get the ratings for the user
    #
    userRatings = self.data[user]
    #
    # determine the total distance
    totalDistance = 0.0
    for i in range(self.k):
       totalDistance += nearest[i][1]
    # now iterate through the k nearest neighbors
    # accumulating their ratings
    for i in range(self.k):
```

Figure 3.2. Recommending Users

contains movie ratings of 1000 users on 1700 movies with 10,000 Movie Ratings (10K)

Link : http://www.grouplens.org/node/73

The other available datasets were the 100K dataset and the 1000K dataset by MovieLens.



```
    # compute slice of pie
    weight = nearest[i][1] / totalDistance
    # get the name of the person
    name = nearest[i][0]
    # get the ratings for this person
    neighborRatings = self.data[name]
    # get the name of the person
    # now find bands neighbor rated that user didn't
    for artist in neighborRatings:
       if not artist in userRatings:
          if artist not in recommendations:
             recommendations[artist] = (neighborRatings[artist]
                                        * weight)
          else:
             recommendations[artist] = (recommendations[artist]
                                        + neighborRatings[artist]
                                        * weight)
# now make list from dictionary
recommendations = list(recommendations.items())
recommendations = [(self.convertProductID2name(k), v)
                   for (k, v) in recommendations]
# finally sort and return
recommendations.sort(key=lambda artistTuple: artistTuple[1],
                     reverse = True)
# Return the first n items
return recommendations[:self.n]
```

Figure 3.3. Finding the nearest neighbor and recommending users

Chapter 4

System Requirement Specification

The Movie Recommender System (MRS) is an online rating website where in users can login to the system and get personalized recommendations. Users can view the movies in a categorized manner and than rate them.

4.1 Document Purpose

Presenting a detailed description of the Movie Recommender System is the purpose of this chapter. The chapter will provide the features of the applications and its purpose in real-life. It will also include the various interfaces of the application, the database setup and the various constraints in which the application can run. This chapter can be referred both by the user and the developer of the application and will be proposed to NIT Rourkela Society for its approval.

4.2 Product Scope

This application will serve as a Web based Movie Recommender System where people can browse, watch and give personal reviews on the various categories of products. This application is designed to engage the online Users/Viewers to provide as many reviews as possible, in an attempt to gather maximum customer/viewer opinion. In return, the system would suggest the customer various similar products, best suited to them, taking into consideration the products they have viewed/purchased and the products other people of similar taste have purchased. More specifically, this system is designed to suggest a customer the product he/ she is most likely to like.

4.3 Intended Audience and Document Overview

The chapter gives a detailed insight, how the recommender system is to be built and, how the application is to be used after its completion. The chapter is thus a reference and guidelines by the developer. Thus the document can be viewed as a reference and suggestive guideline by the developers to work upon.

4.4 Motivation

Interface styling is inspired from the Last.fm website, a site used to recommend music. Other references include the MOOC in Coursera namely, Introduction to Recommender Systems.

4.5 Product Functions

A web based application has been developed that contains a registration page, User Login page, Movies page, Rating page, Recommendations page, a profile page and other added features. Users can register themselves to the application and then browse the movies in the list in the order they wish and can also rate them. The collaborative filtering algorithm comes to play as soon as the user logs in to the application. It look for movies in the database that are predicted to be rated highly by the logged in user. The user can then browse and rate those items.

4.6 User Class and Characteristics

The recommender system basically works between the interaction between the users and the rating they give. The main aim of recommender system is just to process the ratings of the users and generate appropriate predictions for a particular user. Thus following is the class structure of our application:

1. **User**: This class models the reviewer. The information contained would be the credentials to identify the users and the content he generates in form of reviews.

2. **Review**: This class models a review written by the user . This contains information about the what,when and by whom of the review.

3. **Item**: An interface that can be implemented as a movie , a documentary or any other relevant piece of the media, the user is writing reviews about

4.7 Design and Implementation Constraints

The website provides a simple interface that collects ratings for certain movies which can be viewed in a category form. Then, using Collaborative filtering algorithms, the system then recommends other movies based on those ratings and allows the user to rate them. The Home page will display moving boxes with movies that have been added recently and are highly rated by the other users. If the user has logged into the system, the recommendations page will display all the recommended movies for that particular user. These movies are predicted to be highly rated using User-based Collaborative Filtering while the second part will show the movies that are predicted to be highly rated based on Model-based Collaborative filtering. Also, the website has search functionality where users can search for products.

If a particular movie is selected, the system will display a box with similar products based on Item-based Collaborative Filtering. If the user is logged in to the system, then the system will display a predicted rating for that movie using the Item-based Collaborative Filtering next to the item that has been selected.

4.8 User Interfaces

Various Interfaces of the Movie Recommender System are as follows:

1. Home

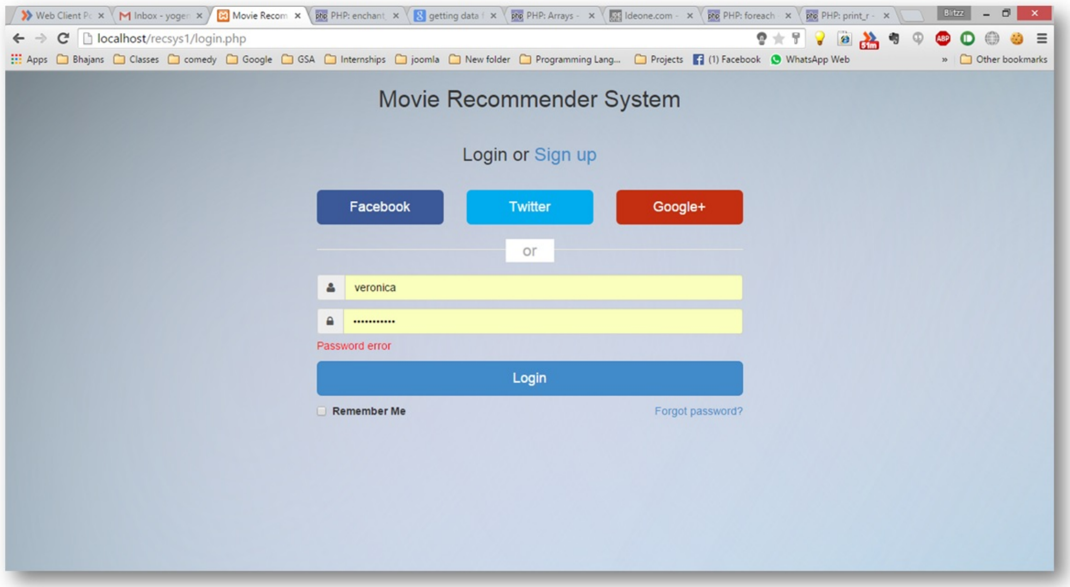
2. Login Form

3. Registration Form

4. View/Browse Movies Form

5. Get Recommendations

4.8.1 Login Page And Registration Form



veronica

Password error

Figure 4.1. Login Page

The first page of our website has login and registration links. The login redirects the user to the login page which has 2 input fields, namely Username and Passowrd. Social Login Plugins has been integrated into the website. Thus a user can register or login to the website using any of his Facebook, Google or Twitter accounts. This has been integrated seeing the comfort of the user and the ease with which he can log in to the site.

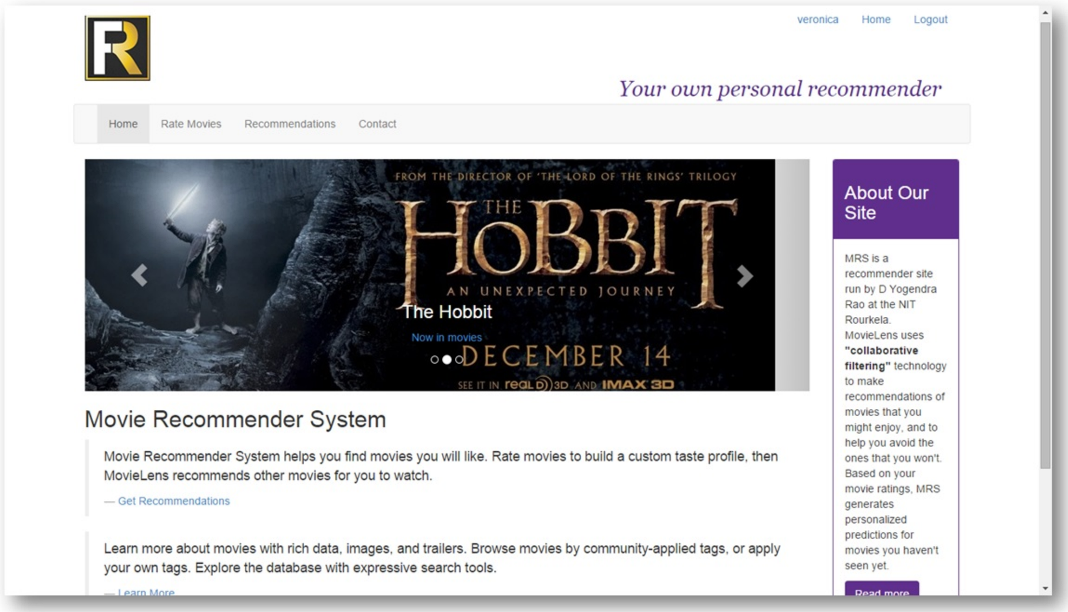
The registration page includes the following fields :

- username
- password

- passoword confirmation
- email
- country
- email
- region
- city
- zip code
- gender

These information that are provided by the user helps in buildings clusters. For more accurate and desired predictions we form clusters of users on the basis of country, city, region and zip code.

4.8.2 Home Page



Movie Recommender System

Movie Recommender System helps you find movies you will like. Rate movies to build a custom taste profile, then MovieLens recommends other movies for you to watch.

Learn more about movies with rich data, images, and trailers. Browse movies by community-applied tags, or apply your own tags. Explore the database with expressive search tools.

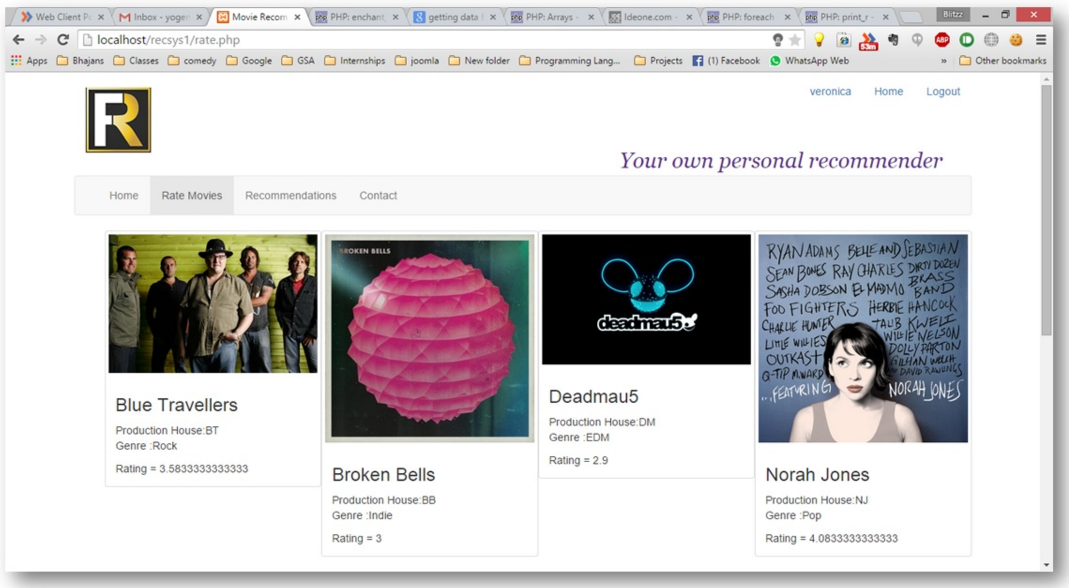
About Our Site

MRS is a recommender site run by D Yogendra Rao at the NIT Rourkela. MovieLens uses **"collaborative filtering"** technology to make recommendations of movies that you might enjoy, and to help you avoid the ones that you won't. Based on your movie ratings, MRS generates personalized predictions for movies you haven't seen yet.

Figure 4.2. Home Page

Once the user logs in to the system, this is the page he/she lands on. This page contains to the navigational links to other webpages where he/she can browse for movies and get recommendations. This page displays moving boxes with movies that have been added recently and are highly rated by the other users.

4.8.3 Movies Page



Blue Travellers
Production House:BT
Genre :Rock
Rating = 3.5833333333333

Broken Bells
Production House:BB
Genre :Indie
Rating = 3

Deadmau5
Production House:DM
Genre :EDM
Rating = 2.9

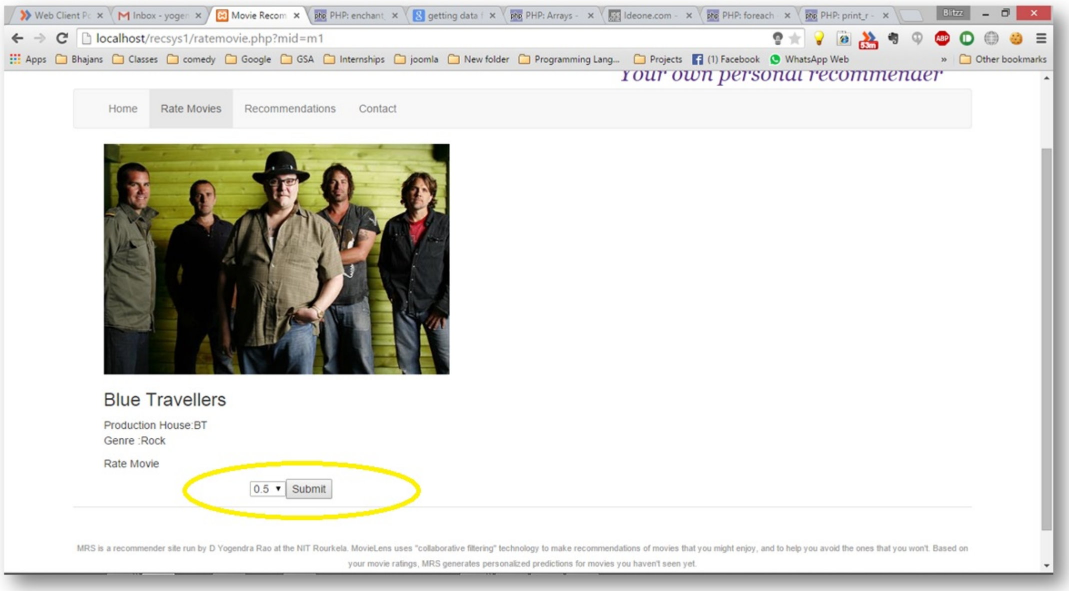
Norah Jones
Production House:NJ
Genre :Pop
Rating = 4.0833333333333

Figure 4.3. Browse/View Movies Page

Users can log in to the site and reach this page via the Home Page. This page displays all the movies in the database along with basic information of the movies like the Production House, Genre, Current Ratings. Users can click on the movie item to check for more details of the same movie.

4.8.4 Rate Page

In this page, user has a option to view the details of the movie. User can then rate the movie in a scale of 0-5 with 0.5 step. The value is then submitted to the database where in the rating of the movie is updated. The user can come to this page even if he has already rated the movie.



localhost/recsys1/ratemovie.php?mid=m1

Your own personal recommender

Home | Rate Movies | Recommendations | Contact

Blue Travellers

Production House:BT
Genre :Rock

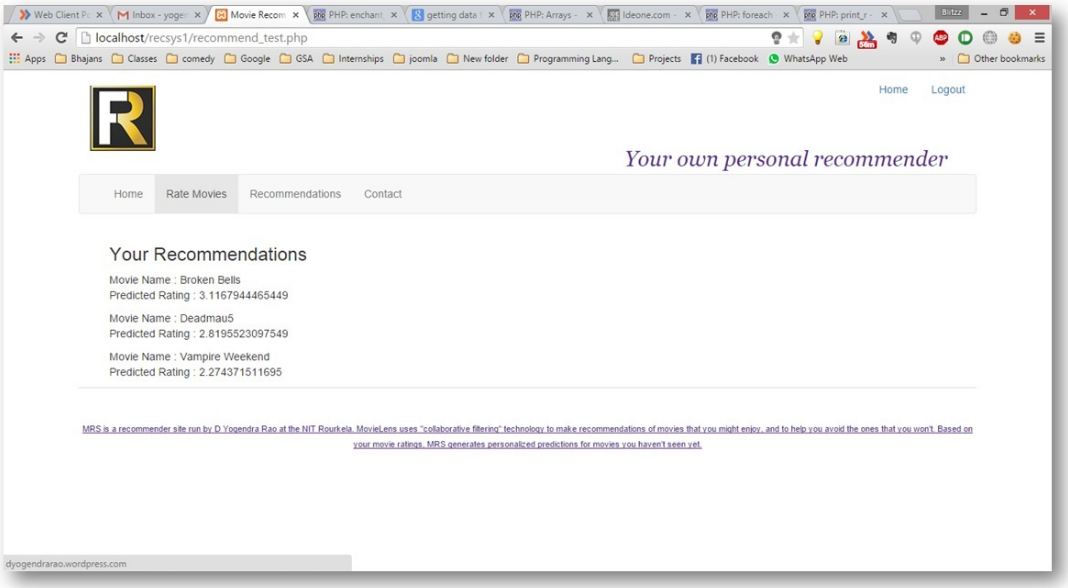
Rate Movie

0.5 Submit

MRS is a recommender site run by D Yogendra Rao at the NIT Rourkela. MovieLens uses "collaborative filtering" technology to make recommendations of movies that you might enjoy, and to help you avoid the ones that you won't. Based on your movie ratings, MRS generates personalized predictions for movies you haven't seen yet.

Figure 4.4. Rate Movies Page

4.8.5 Recommendations Page



localhost/recsys1/recommend_test.php

Home Logout

Your own personal recommender

Home | Rate Movies | Recommendations | Contact

Your Recommendations

Movie Name : Broken Bells
Predicted Rating : 3.1167944465449

Movie Name : Deadmau5
Predicted Rating : 2.8195523097549

Movie Name : Vampire Weekend
Predicted Rating : 2.274371511695

MRS is a recommender site run by D Yogendra Rao at the NIT Rourkela. MovieLens uses "collaborative filtering" technology to make recommendations of movies that you might enjoy, and to help you avoid the ones that you won't. Based on your movie ratings, MRS generates personalized predictions for movies you haven't seen yet.

dyogendrarao.wordpress.com

Figure 4.5. Get Recommendations Page

This is the page where user get the recommendations according to movies that he has rated. The predicted value is calculated using the formula that has been introduced in Chapter 2. The Recommender System displays a predicted rating using the Item-based Collaborative Filtering Algorithm next to the item that has been recommended.

4.8.6 Profile Page

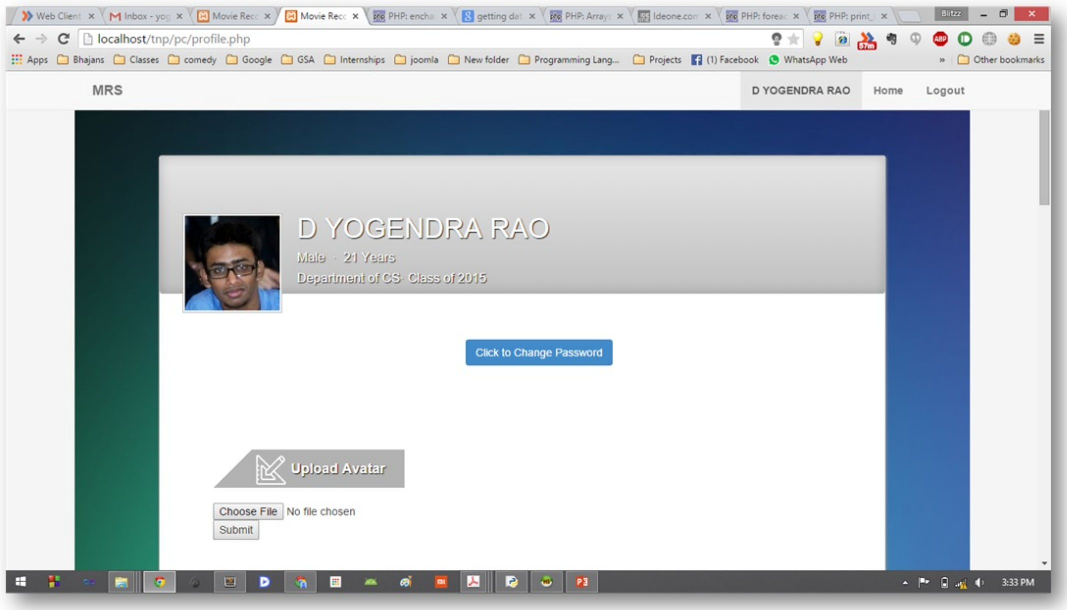


Figure 4.6. Get Recommendations Page

This is the profile page of a user. It has a feature to upload his/her avatar and his profile details like Name, Email, Country etc. User can directly check the number of movies he/she has rated.

4.9 Technologies Used

The following were the tools and technologies used to build the engine and the website :

1. Front-end : Bootstrap framework for CSS and JavaScript
2. Server side scripting : PHP the server side

3. Recommender Engine : Python
4. Web Services : JSON
5. Web Applications : XAMPP Server
6. Database : MySQL
7. Server Deployment : Xampp Server

4.10 Database Structure

The database was designed and implemented in MySQL. There are total of 6 tables in the database that the website is using.

Table User		
Field Name	Type	Keys
uid	varchar	Primary
username	varchar	
Password	varchar	
email	varchar	
Gender	enum('M','F')	
country	varchar	
region	varchar	
city	int	
zipcode	int	

Table 4.1. Table 1 : User

Table Country		
Field Name	Type	Keys
ccode	varchar	Primary
country	varchar	

Table 4.2. Table 2 : Country

Table Region		
Field Name	Type	Keys
region_id	int	Primary
ccode	varchar	
name	varchar	

Table 4.3. Table 3 : Region

Table City		
Field Name	Type	Keys
city_id	int	Primary
ccode	varchar	
reg_id	int	
city	varchar	

Table 4.4. Table 4 : City

Table Movies		
Field Name	Type	Keys
mid	int	Primary
mname	varchar	
prod_house	int	
genre	varchar	
mimg	varchar	

Table 4.5. Table 5 : Movies

Table Ratings		
Field Name	Type	Keys
uid	varchar	
mid	varchar	
rate	float	

Table 4.6. Table 6 : Rating

Chapter 5

Results and Evaluations

To evaluate the three implemented algorithms, I had used the Movie Lens data set and the algorithms were coded in Python using Eclipse IDE. I have implemented the three Collaborative Filtering algorithms namely User-Based, Item-Based and Model-Based and then calculated the error rate for each one of the algorithms using the Mean Absolute Error Rate (MAE) and Root Mean Squared Error using the following formulas[1]:

$$MAE = \frac{1}{|T|} \sum_{(u,i,r) \in T} |p_{ui} - r|$$

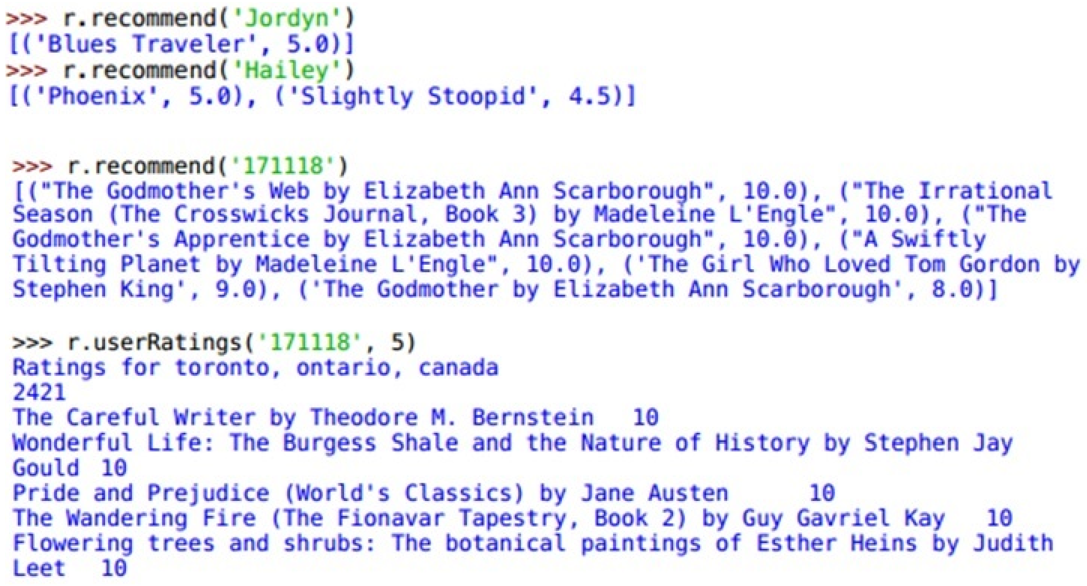
$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(u,i,r) \in T} (p_{ui} - r)^2}$$

Figure 5.1. Error Calculation

Where T is the total number of test cases, in our case 10K(10000), pui is the predicted rating given by the algorithm, and r is the actual rate that the user is rating.

User Based		Item Based		Content Based	
MAE	RMSE	MAE	RMSE	MAE	RMSE
0.816429099	1.022968896	0.834149128	1.04480249	0.85019701	1.10512221

Table 5.1. Results



```
>>> r.recommend('Jordyn')
[('Blues Traveler', 5.0)]
>>> r.recommend('Hailey')
[('Phoenix', 5.0), ('Slightly Stoopid', 4.5)]

>>> r.recommend('171118')
[("The Godmother's Web by Elizabeth Ann Scarborough", 10.0), ("The Irrational
Season (The Crosswicks Journal, Book 3) by Madeleine L'Engle", 10.0), ("The
Godmother's Apprentice by Elizabeth Ann Scarborough", 10.0), ("A Swiftly
Tilting Planet by Madeleine L'Engle", 10.0), ('The Girl Who Loved Tom Gordon by
Stephen King', 9.0), ('The Godmother by Elizabeth Ann Scarborough', 8.0)]

>>> r.userRatings('171118', 5)
Ratings for toronto, ontario, canada
2421
The Careful Writer by Theodore M. Bernstein   10
Wonderful Life: The Burgess Shale and the Nature of History by Stephen Jay
Gould  10
Pride and Prejudice (World's Classics) by Jane Austen      10
The Wandering Fire (The Fionavar Tapestry, Book 2) by Guy Gavriel Kay   10
Flowering trees and shrubs: The botanical paintings of Esther Heins by Judith
Leet   10
```

Figure 5.2. Results : Getting Recommendations along with Predicted Rating

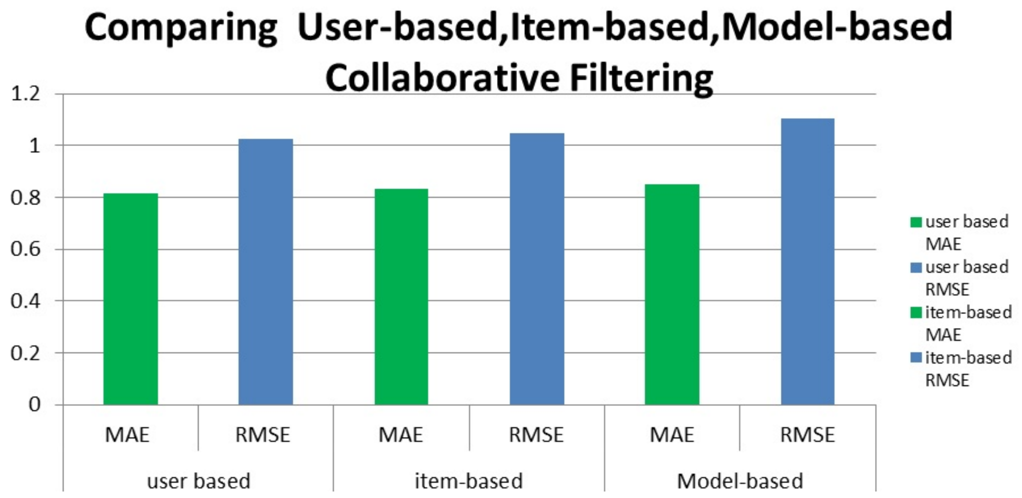


Figure 5.3. Comparison of the 3 Collaborative Filtering Algorithms

User-based collaborative filtering approach is the best as the accuracy was higher in its case. Additionally, the time complexity of Item-based CF and Model based CF are higher than the User-Based CF.

Chapter 6

Conclusion

Thus the recommender system was successfully implemented. We found that User-Based Collaborative Filtering was the best as the accuracy was higher in its case as compared to the rest of the methods. For working on large dataset, it was an approach in implementing the algorithm and making it a web-based Recommender System. This is similar to the algorithm that Netflix uses in its website to recommend movies to its customers. It was a challenge for me to implement a web-based recommender system on this scale of huge data.

Recommender systems have become ubiquitous. People use them to find books, music, news, smart phones, vacation trips, and romantic partners. Nearly every product, service, or type of information has recommenders to help people select from among the myriad alternatives the few they would most appreciate. Sustain- ing these commercial applications is a vibrant research community, with creative interaction ideas, powerful new algorithms, and careful experiments

Bibliography

[1] Allahaidan, Ala *Recommender System Using Collaborative Filtering Algorithm.* Technical Library: School of Computing and Information Systems; p155 2013

[2] Ekstrand, Michael D., John T. Riedl, and Joseph A. Konstan "Collaborative filtering recommender systems." *Foundations and Trends in Human-Computer Interaction* 4, no. 2 (2011): 81-173

[3] R. Burke "Hybrid Recommender Systems: Survey and Experiments, In: User Modeling and User-Adapted Interaction" *Kluwer Academic Publishers*, 12 (4) (2002), pp. 331370

[4] Laurent Candillier and Frank Meyer and Marc Boull "Comparing State-of-the-Art Collaborative Filtering Systems"

[5] Linden, Greg, Brent Smith, and Jeremy York. "Amazon. com recommendations: Item-to-item collaborative filtering." *Internet Computing, IEEE 7*, no. 1 (2003): 76-80.

[6] Joseph Konstan and Michael Ekstrand "Introduction to Recommender Systems" *Coursera, Online MOOC.*