MOVIE RECOMMENDATION SYSTEM

Type -2 Project

Guided by:
Dr. Dongsong Zhang

SECTION 2

Submitted by:
Gokul Devunuri
Muthukumar Thevar
Prachi Dharurkar
Vamsi Krishna Ginjipalli

Table of Contents

1. Problem Statement
2. Abstract
3. Introduction
4. Methodology
5. System Design
    5.1 Process Flow Design
    5.2 Dataset
    5.3 Hardware and Software requirements
6. System Implementation
    6.1 R Programming
    6.2 R Package
    6.3 Pre-processing
    6.4 System Inputs
    6.5 Decision Model
    6.6 System Outputs
7. Evaluation
    7.1 Six Models
    7.2 RMSE, MSE and MAE Calculation
8. System Limitation
9. Conclusion
10. References

1. Problem Statement

Given a database of movies and users, our system has to predict a list of movies for a particular user based on the information of the other users in the database by applying the User Based Collaborative filtering approach.

2. Abstract

Collaborative filtering is key technique of recommendation system. We have implemented a User based Collaborative filtering algorithm based on user similarity combination, which combines the user similarity based on user-rating movies. In this paper, we have created a six models of User based Collaborative Filtering by changing the normalization techniques and similarity function (Cosine, Pearson and Jaccard). We have compared the prediction time of the individual models. We carried out an experiment on Movielens datasets to evaluate the algorithm and used RMSE, MSE and MAE as the performance index by changing the nearest neighbours of our User based system model.

3. Introduction

Recommendation System

Obtaining recommendations from trusted sources is a critical component of the natural process of human decision making. We live in the era of Information Systems and internet. Too much of information is out there to make a single choice. Right from looking for a good movie to watch to look for good property options, there is too much information available. Recommender systems help people to cope with such huge amount of information by suggesting items that they will like. These systems are a subclass of Information filtering system that try to predict the 'rating' or 'preference' that a user would give to an item [4]. Suggestions for books or products on Amazon, or movies on Netflix, are real world examples of recommended systems. Today recommender systems are an accepted technology used by many industry leaders. Recommender Systems have evolved to fulfil the need of buyers and sellers by automating the generation of recommendations based on data analysis. It has become common for enterprises to collect large volumes of transactional data that allows analysis of how a customer interacts with the product offerings. Such recommendations can help the customer to find products she/he wants to buy faster, promote cross-selling by suggesting additional products.

Collaborative Filtering:

Collaborative Filtering is one of the most commonly used methods in personalized recommendation systems. It is based on the assumption that people will get best recommendation from person who has similar tastes to themselves. Recommender systems need to maintain user profiles which certain information about the user preferences to achieve personalization.

CF is further classified as three main filtering techniques:

1. Memory-based Collaborative Filtering
2. Model-based Collaborative Filtering
3. Hybrid Collaborative Filtering

Memory-based Collaborative Filtering

The memory based collaborative filtering technique is used to make recommendations. This CF uses the entire or sample user-item rating dataset to compute similarity between users or items and then generates a prediction. The user-item rating data set contains rating given by users to particular item. The systems apply different techniques to find a set of similar users, known as neighbours. Next step is to produce prediction for active user and generate a top-N most frequent items as recommendation by aggregating similar uses. Some important algorithms are: User-based collaborative filtering, Item-based collaborative filtering.

However, the memory-based CF suffers from two basic problems: sparsity and scalability. Sparsity refers to the problem that most users rate only a few number of items which lead to difficulty in calculating correlation. The nearest neighbour algorithms requires computation that grows with both the number of items and the number of users. With millions of users, items and their ratings, a typical recommender system running existing algorithms will suffer serious scalability problems [5].

Model based Collaborative Filtering

Model-based CF techniques use the pure rating data to estimate or learn a model to make predictions. The model can be a data mining or machine learning algorithm [7]. The design and development of models can allow the system to learn to recognize complex patterns. Then make intelligent predictions for the collaborative filtering tasks for real-world data. Model-based CF algorithms have been investigated to solve the drawbacks of memory-based CF. Some important model-based CF techniques are Bayesian belief nets (BNs) CF models, clustering CF models, and latent semantic indexing CF models.

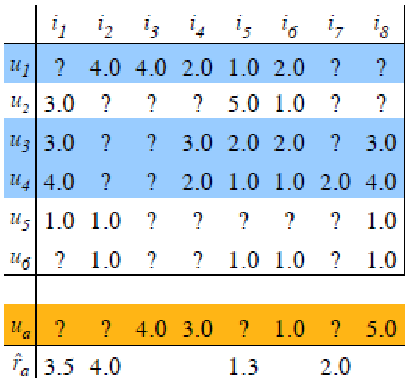
Hybrid Collaborative Filtering:

Hybrid collaborative filtering technique is used to overcome problems of memory-based and model-based collaborative filtering. Hybrid CFAs combine CF with other recommendation techniques to make predictions. Content-based CF makes recommendations by analysing the content of textual information. A content-based recommender then uses classification algorithms to make recommendations. Most hybrid methods applied user profiles and descriptions or textual information of items to find users who have similar interests, then used collaborative filtering to make predictions [6]. Some important algorithm is: Personality diagnosis collaborative filtering.

4. Methodology:

User Based Collaborative Filtering:

User based collaborative filtering, also known as k-NN collaborative filtering, was the first of the automated collaborative filtering methods, and it works on the assumption that users with similar preferences will rate items similarly. So, missing ratings for any given user can be predicted by first finding the neighbourhood of the users with most similarity and then aggregating the ratings of these users to form prediction.



	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	?	4.0	4.0	2.0	1.0	2.0	?	?
u_2	3.0	?	?	?	5.0	1.0	?	?
u_3	3.0	?	?	3.0	2.0	2.0	?	3.0
u_4	4.0	?	?	2.0	1.0	1.0	2.0	4.0
u_5	1.0	1.0	?	?	?	?	?	1.0
u_6	?	1.0	?	?	1.0	1.0	?	1.0
u_a	?	?	4.0	3.0	?	1.0	?	5.0
$\hat{r}_a$	3.5	4.0			1.3		2.0	

Fig 1: User Rating Matrix[2]

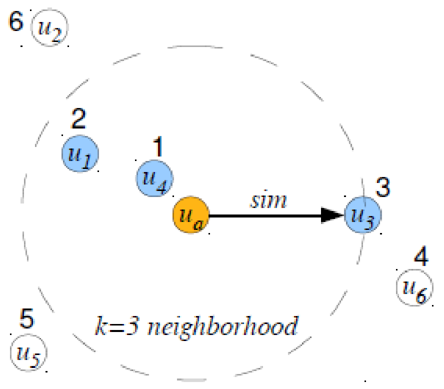


Fig 2: Neighbourhood Formation[2]

Fig 1 is the rating matrix with 6 users and 8 items with ratings ranging from 1 to 5, we need to find recommendations for the active user u_a shown at the bottom of the ratings matrix. In order to find the neighbourhood we first calculate the user similarity between active users and all other users, based on their given ratings, with help of Pearson correlation coefficient.

And then find the users with highest similarity which is 3 in our model, once the users in neighbourhood are determined then we need to aggregate the ratings for the items which are not rated by active user, in our example for the items i_1, i_2, i_5 and i_7 aggregated ratings are calculated and they are recommended as predicted ratings for the active user.

The algorithm can be summarized in the following steps [1]

Step 1: all users are weighted with respect to similarity with the active user.

Step 2: select 'n' active users that have the highest similarity.

Step 3: Compute a Prediction, $P_{a,u}$ from a weighted combination. Similarity between two users is computed using the Pearson correlation coefficient.

$$P_{a,u} = \frac{\sum_{i=1}^{m} (r_{a,i} - \bar{r}_a) \times (r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i=1}^{m} (r_{a,i} - \bar{r}_a)^3 \times \sum_{i=1}^{m} (r_{u,i} - \bar{r}_u)^3}}$$

Where $r_{a,i}$ is the rating given to item i by user a; and r_a is the mean rating given by user a.

In step 3, predictions are computed as the weighted average of deviations from the neighbour's mean:

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n} (r_{u,i} - \bar{r}_u) \times P_{a,u}}{\sum_{u=1}^{n} P_{a,u}}$$

Where P $_{a,i}$ is the prediction for the active user a for item i. P $_{a,u}$ is the similarity between users a and u. n is the number of users in the neighbourhood.

5. System Design

5.1 Process Flow Diagram:

In our Model, the Dataset we collected is loaded into R, once data is available in the system then we used User based Collaborative filtering Approach with Z score Normalization and Pearson Model, based on the low Process time and Root mean Square Error when compared with other models, and then we trained the system with all the input data, once the system is trained we predicted movie ratings for different users by calculating the neighbourhood of similar users and then aggregating the ratings of those neighbour users to form a prediction.

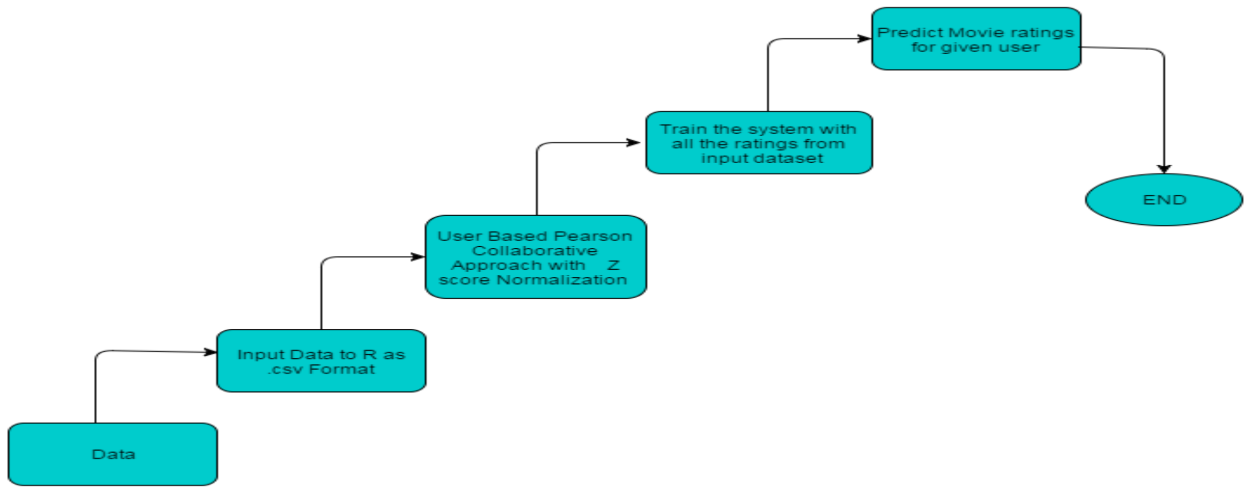

Data
Input Data to R as .csv Format
User Based Pearson Collaborative Approach with Z score Normalization
Train the system with all the ratings from input dataset
Predict Movie ratings for given user
END

Fig 3: Process Flow Diagram

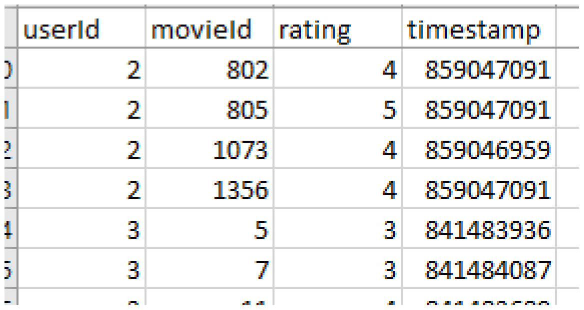
5.2 Dataset

GroupLens Research has collected and made available rating datasets from the MovieLens website (http://movielens.org). We have used the MovieLens dataset which is last updated on 1/2016. The dataset we have used contains 100,000 ratings applied to 10,000 movies by 700 users. Dataset can be download from the following link (http://files.grouplens.org/datasets/movielens/ml-latest-small.zip).

Dataset Description

Movie ratings of individual users is stored on the ratings.csv file which contains 668 users and 10325 movies ratings of individual users. Each movie is rated by at least 20 users.

Sample Data of ratings.csv file



userId	movieId	rating	timestamp
2	802	4	859047091
2	805	5	859047091
2	1073	4	859046959
2	1356	4	859047091
3	5	3	841483936
3	7	3	841484087

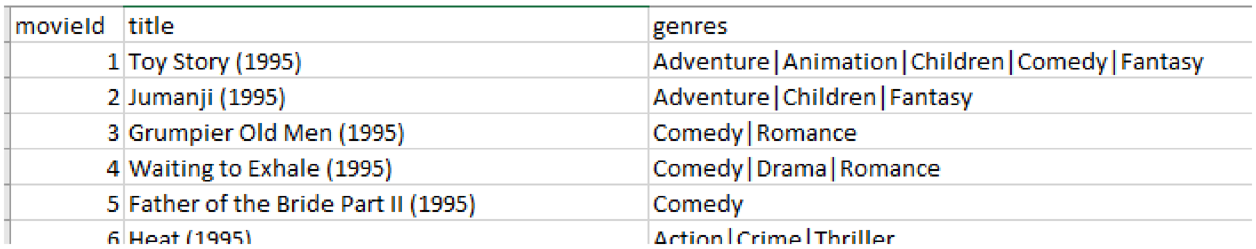
userId: - User ID of the Individual Users.

movieId: - Movie ID of the Individual Movies.

rating: - ratings given by the particular user for the respective movie (0.5 to 5, incremented by 0.5).

timestamp: - timestamp on which the user is rated the movie.

Movie names of the Respective movieId is stored on the movies.csv



movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller

movieId: - unique movie Id of the individual movies.

title: - movie name of the respective movieId.

genres :- genres of the respective movie.

5.3 Hardware and Software Requirements

Hardware

System will function properly on machine with following requirements:

1. RAM: 2 GB
2. Processor: Dual core (or higher)

Software

1. Our decision model is based on R Studio Desktop 0.99.896, it will work on system which has R installed on it, and our version of R is 3.3.0
2. OS: Windows XP, 7 or any higher version.

6. System Implementation

We have implemented of our Movie Recommendation System in R programming language.

6.1 R Programming

R is a programming language for statistical computing and graphics by the R foundation for Statistical Computing. It is widely used by data miner for developing statistical software and data analysis.

6.2 R Packages

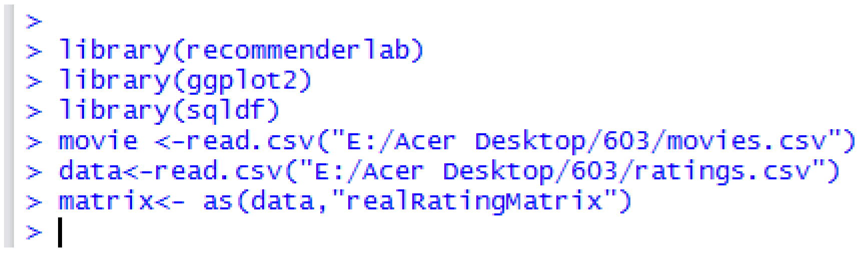
recommenderlab package for R provides us a research infrastructure for developing and testing Recommender Algorithms. More details can be found from the link (https://cran.r-project.org/web/packages/recommenderlab/index.html).

sqldf package helps us to manipulate R data frames using SQL. More details can be found from the link (https://cran.r-project.org/web/packages/sqldf/index.html).

6.3 Pre-processing

All the ratings of the users for a particular movie are stored in the ratings.csv file. To apply the User based Collaborative filter algorithm these ratings as to be converted into users-movies rating matrix.

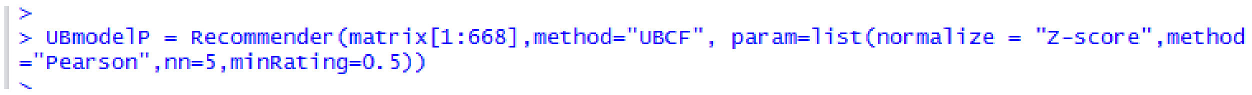
6.4 System Inputs


```
>
> library(recommenderlab)
> library(ggplot2)
> library(sqldf)
> movie <-read.csv("E:/Acer Desktop/603/movies.csv")
> data<-read.csv("E:/Acer Desktop/603/ratings.csv")
> matrix<- as(data,"realRatingMatrix")
> 
```

The above commands help us to create the users-movie rating matrix in R.

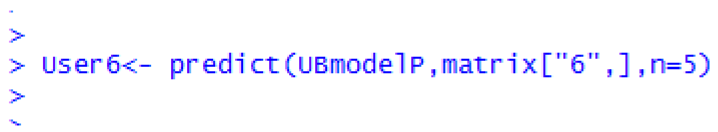
6.5 Decision Model

User Based Collaborative Filter Model in R


```
>
> UBmodelP = Recommender(matrix[1:668],method="UBCF", param=list(normalize = "Z-score",method
="Pearson",nn=5,minRating=0.5))
```

Class Recommender implements the data structure to store recommendation models. The above is the creation method where

"UBCF" specify the User based Collaborative Filtering model, normalize = "Z-score" specify to create the model with Z-score Normalization, "Pearson" specify to create the model with Pearson similarity function, nn = 5 specify to use the 5 nearest neighbour to the model and minRating = 0.5 specify select only those rating which at least 0.5

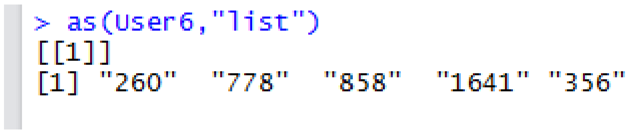

```
>
> User6<- predict(UBmodelP,matrix["6",],n=5)
>
```

Where UBmodelP is a recommender model.

matrix ["6",] is realRatingmatrix for the User 6.

n is the number of recommendations in the top-N list.

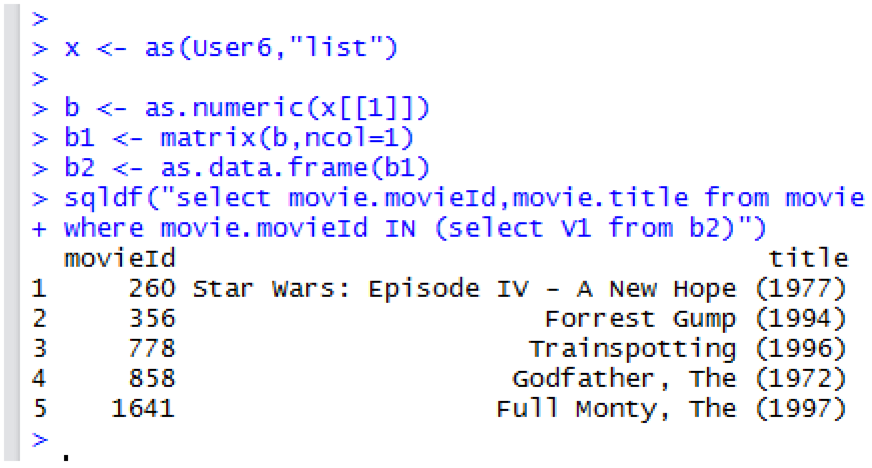
6.6 System Outputs



```
> as(User6,"list")
[[1]]
[1] "260"  "778"  "858"  "1641" "356"
```

Now displaying the User6 variable which contains the top 5 movieId for the User6.

To get the movie name from the movies.csv, we have taken the help of the sqldf package.



```
>
> x <- as(User6,"list")
>
> b <- as.numeric(x[[1]])
> b1 <- matrix(b,ncol=1)
> b2 <- as.data.frame(b1)
> sqldf("select movie.movieId,movie.title from movie
+ where movie.movieId IN (select V1 from b2)")
  movieId                                      title
1     260 Star Wars: Episode IV - A New Hope (1977)
2     356                       Forrest Gump (1994)
3     778                      Trainspotting (1996)
4     858                     Godfather, The (1972)
5    1641                    Full Monty, The (1997)
> 
```

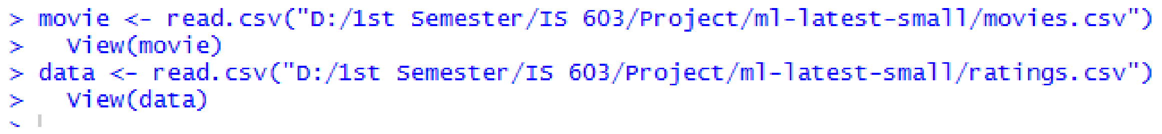
7. Evaluation:

As discussed before, we have various parameters for evaluating a collaborative filtering model. So, we have compared all the 6 possible models by changing the normalization and the similarity function. We have 2 different types of normalization i.e. Z-Score and center, and 3 types of similarity functions i.e. Cosine, Pearson and Jaccard.

7.1 Six Models:

Normalization	Similarity Function
Z Score	Pearson Cosine Jaccard
Centre	

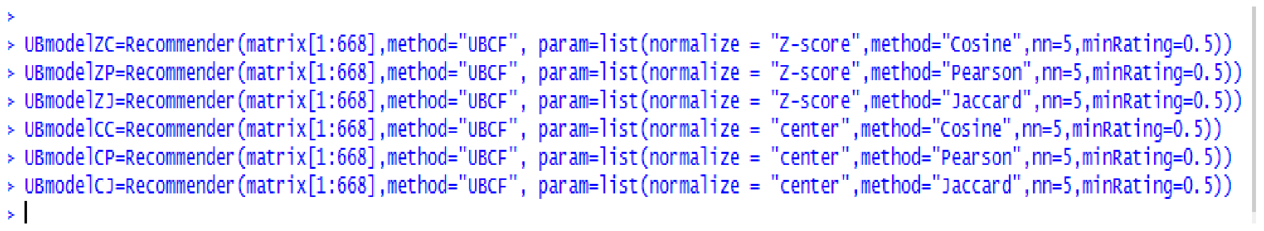
1. User Based CF Z-Score Cosine
2. User Based CF Z-Score Pearson
3. User Based CF Z-Score Jaccard
4. User Based CF Center Cosine
5. User Based CF Center Pearson
6. User Based CF Center Jaccard



```
> movie <- read.csv("D:/1st Semester/IS 603/Project/ml-latest-small/movies.csv")
>   View(movie)
> data <- read.csv("D:/1st Semester/IS 603/Project/ml-latest-small/ratings.csv")
>   View(data)
```

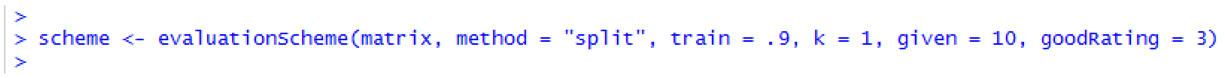
We have loaded the movies.csv and ratings.csv in R.

Below is the command to create the six models in R



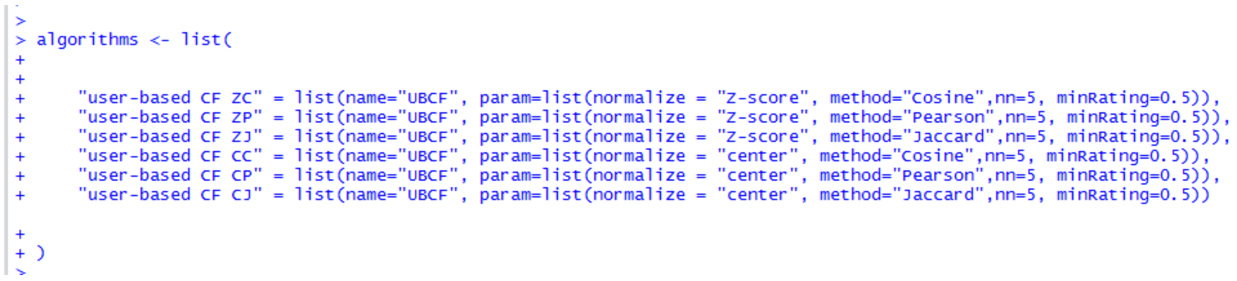
```
>
> UBmodelZC=Recommender(matrix[1:668],method="UBCF", param=list(normalize = "Z-score",method="Cosine",nn=5,minRating=0.5))
> UBmodelZP=Recommender(matrix[1:668],method="UBCF", param=list(normalize = "Z-score",method="Pearson",nn=5,minRating=0.5))
> UBmodelZJ=Recommender(matrix[1:668],method="UBCF", param=list(normalize = "Z-score",method="Jaccard",nn=5,minRating=0.5))
> UBmodelCC=Recommender(matrix[1:668],method="UBCF", param=list(normalize = "center",method="Cosine",nn=5,minRating=0.5))
> UBmodelCP=Recommender(matrix[1:668],method="UBCF", param=list(normalize = "center",method="Pearson",nn=5,minRating=0.5))
> UBmodelCJ=Recommender(matrix[1:668],method="UBCF", param=list(normalize = "center",method="Jaccard",nn=5,minRating=0.5))
> |
```

Then we create the 6 models, and evaluate all the above 6 models by splitting the dataset into 90% training data and 10% test data by using the method = “split” and train = “.9”. For the test set 10 items will be given to the recommender algorithm and the other items will be held out for computing the error as given = 10. We do the cross validation for 1 time as k = 1, and a good rating is considered if the rating of the movie is 3 or more.



```
>
> scheme <- evaluationScheme(matrix, method = "split", train = .9, k = 1, given = 10, goodRating = 3)
>
```

Now, we create an ‘algorithm’ function call which has all the 6 models described in it.



```
>
> algorithms <- list(
+
+
+     "user-based CF ZC" = list(name="UBCF", param=list(normalize = "Z-score", method="Cosine",nn=5, minRating=0.5)),
+     "user-based CF ZP" = list(name="UBCF", param=list(normalize = "Z-score", method="Pearson",nn=5, minRating=0.5)),
+     "user-based CF ZJ" = list(name="UBCF", param=list(normalize = "Z-score", method="Jaccard",nn=5, minRating=0.5)),
+     "user-based CF CC" = list(name="UBCF", param=list(normalize = "center", method="Cosine",nn=5, minRating=0.5)),
+     "user-based CF CP" = list(name="UBCF", param=list(normalize = "center", method="Pearson",nn=5, minRating=0.5)),
+     "user-based CF CJ" = list(name="UBCF", param=list(normalize = "center", method="Jaccard",nn=5, minRating=0.5))

+
+ )
>
```

Then we evaluate all the 6 models based on prediction time by changing the number of recommendation movies to predict.

results <- evaluate (scheme, algorithms, n=c (1, 3, 5, 10, 15, 20))

Model	Prediction Time
User Based Z-Score Cosine	4.04sec
User Based Z-Score Pearson	2.26sec
User Based Z-Score Jaccard	2.78sec
User Based Center Cosine	3.99sec
User Based Center Pearson	2.25sec
User Based Center Jaccard	2.77sec

True Positive Rate vs False Positive Rate

True Positive Rate or Recall rate measures the proportion of actual positives which are correctly identified.

False Positive Rate measures the proportion of actual positives which are incorrectly identified.

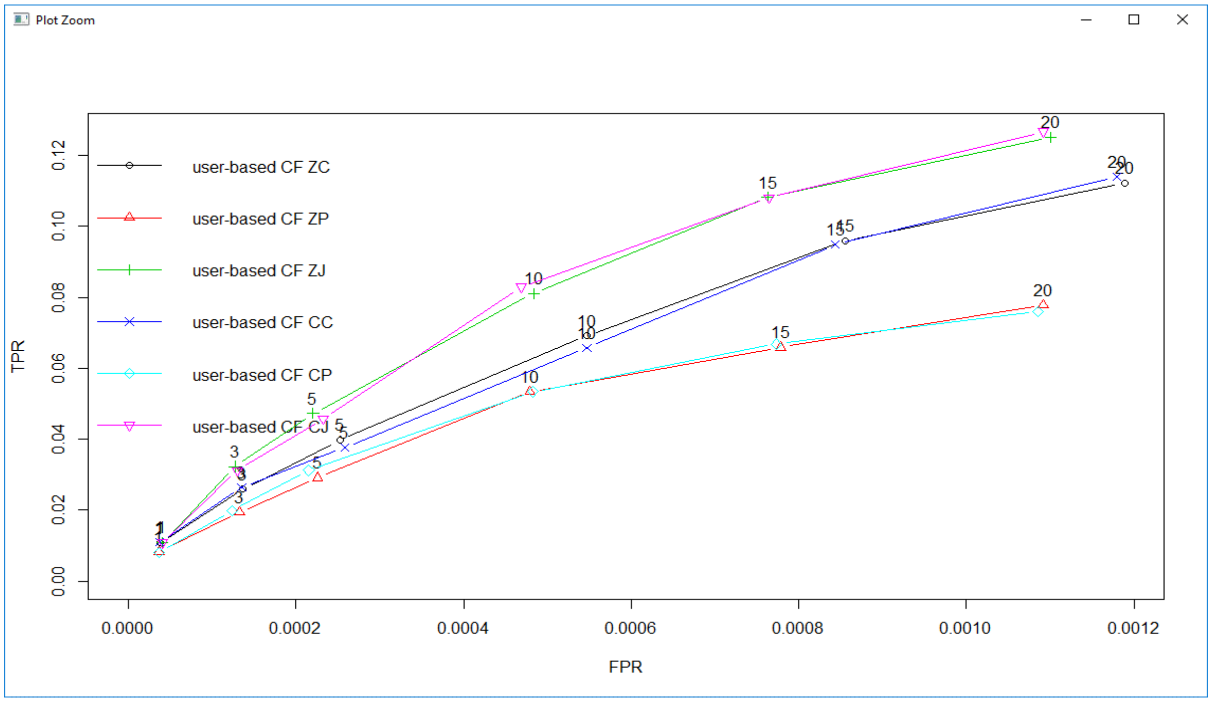


Fig 4: TPR vs FPR

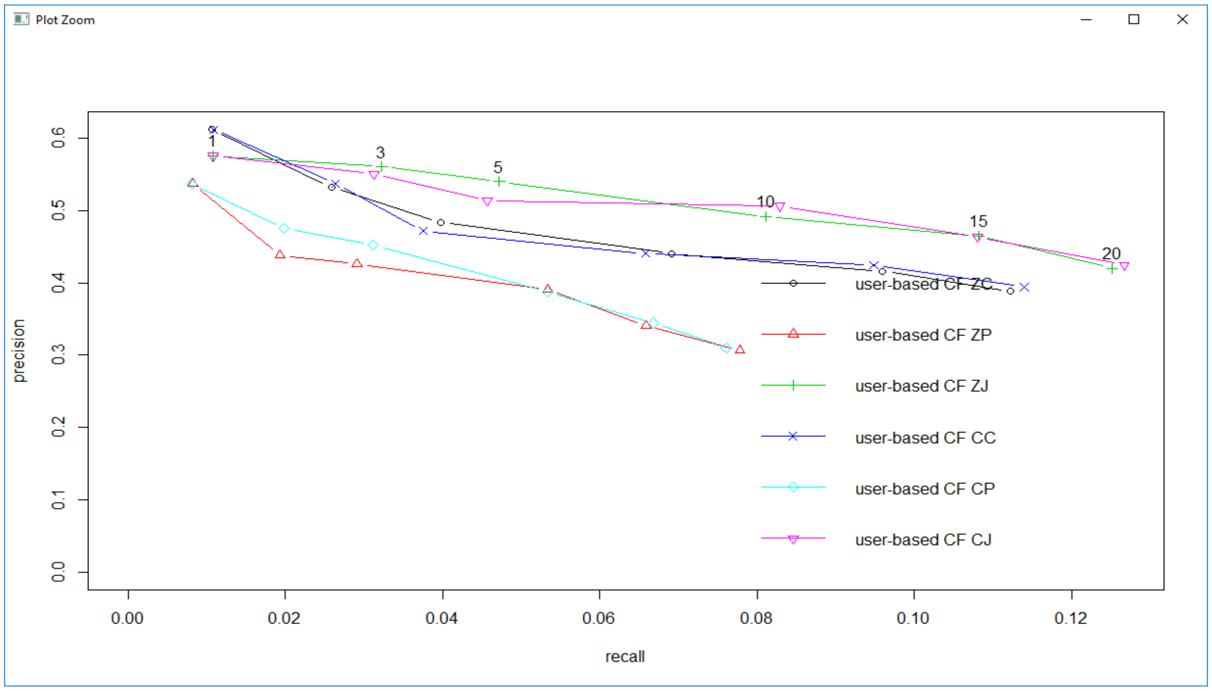


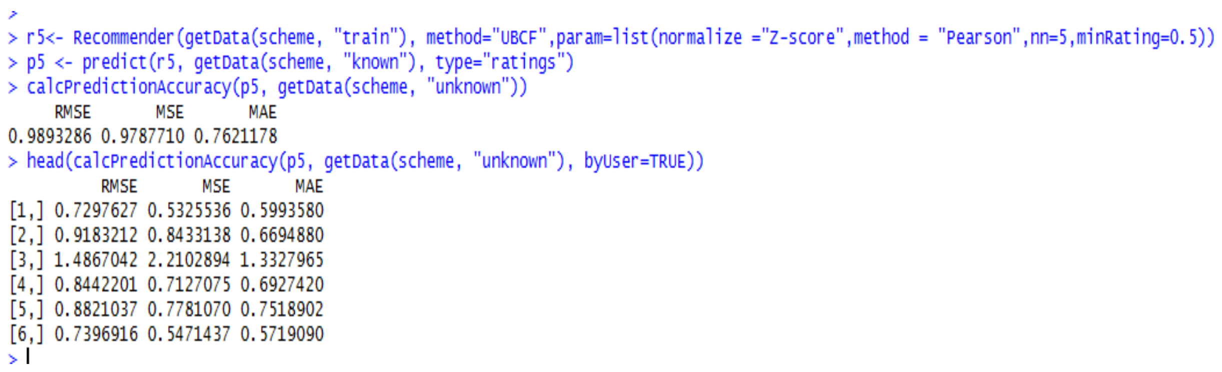
Fig 5: Precision vs Recall

We can infer that Pearson similarity function has the least prediction time, and normalization doesn't have much impact on the system.

7.2 RMSE, MSE, MAE Calculation:

We then evaluate the algorithm with RMSE, MSE and MAE as the performance index by changing the nearest neighbours of our User based system model with 90% training dataset and 10% testing dataset.

Below are commands we used to calculate the RMSE, MSE and MAE.



```
>
> r5<- Recommender(getData(scheme, "train"), method="UBCF",param=list(normalize ="Z-score",method = "Pearson",nn=5,minRating=0.5))
> p5 <- predict(r5, getData(scheme, "known"), type="ratings")
> calcPredictionAccuracy(p5, getData(scheme, "unknown"))
     RMSE       MSE       MAE 
0.9893286 0.9787710 0.7621178 
> head(calcPredictionAccuracy(p5, getData(scheme, "unknown"), byUser=TRUE))
          RMSE       MSE       MAE
[1,] 0.7297627 0.5325536 0.5993580
[2,] 0.9183212 0.8433138 0.6694880
[3,] 1.4867042 2.2102894 1.3327965
[4,] 0.8442201 0.7127075 0.6927420
[5,] 0.8821037 0.7781070 0.7518902
[6,] 0.7396916 0.5471437 0.5719090
> |
```

.

By changing the nearest neighbour nn value from 5,10,15,20,25,30 and 35 we have calculated the individual error values. Below is the table and their respective error calculation.

nn	RMSE	MSE	MAE
5	0.9362157	0.8764998	0.7223302
10	0.9287557	0.8625872	0.7169214
15	0.9210859	0.8483991	0.7106088
20	0.9180216	0.8427637	0.7099800
25	0.9143406	0.8360188	0.7074983
30	0.9259031	0.8572966	0.7164412
35	0.9294946	0.8639603	0.7196776

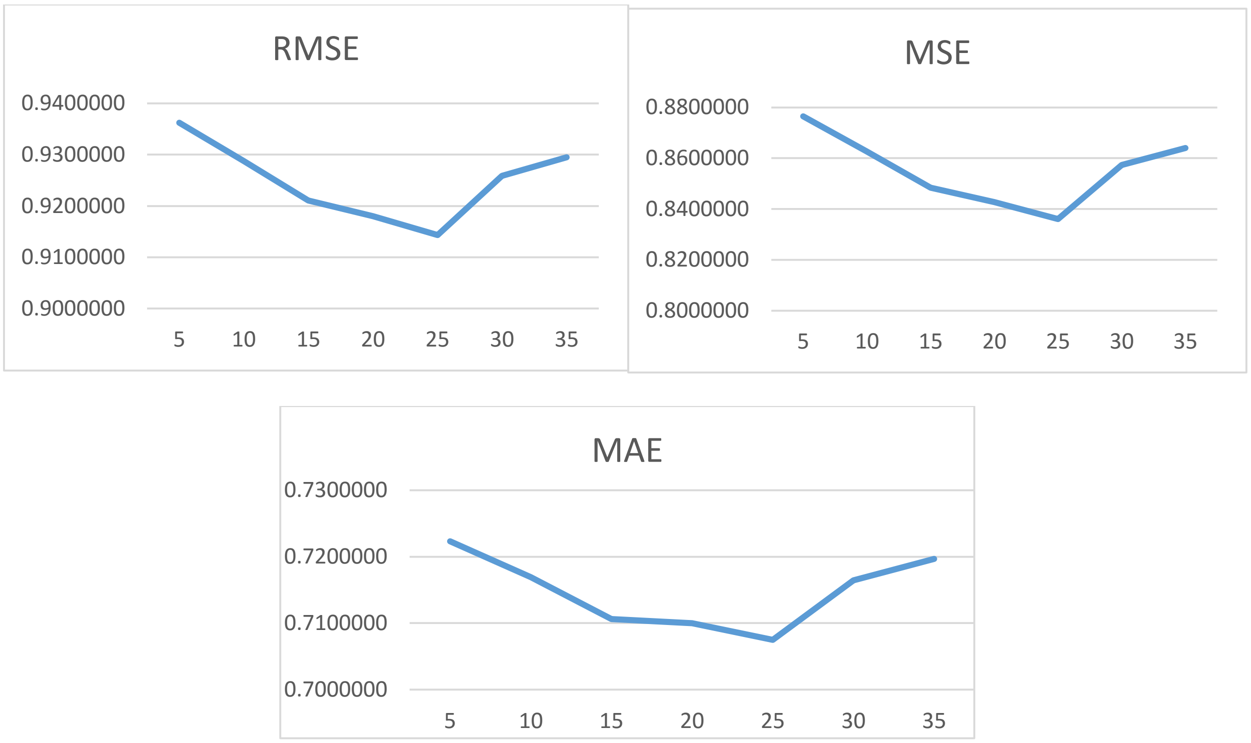


Fig 6 : RMSE , MSE , MAE Graph

We have calculated the RMSE, MSE and MAE by changing the nearest neighbour value and as we can observe from the above figures that the error tends to increase after 25, so 25 neighbours is the threshold value.

8. System Limitations

In general, the limitations for User based collaborative filtering are

1. Cold Start: The number of users in the system have to be high enough so as to find a possible existing user with respect to the active user.
2. Sparsity: If there are too many items to be recommended to the user, then even if there are many users, the user/item rating matrix will be too huge, and it will be hard to find the users that have rated the same items.
3. First Rater: We can only those movies which have been recommended by the users, we cannot recommend an item that has not been previously rated by any user.

9. Conclusion

We have successfully evaluated and implemented user based collaborative filtering algorithm. User based collaborative filtering with Pearson similarity function performed best out of all the six models. To this model, we have calculated the RMSE, MSE and MAE by changing the nearest neighbour value and found that 25 neighbours is the threshold value. Changing the Normalization techniques didn't have much impact on our system.

10. References

[1] Babu, M. S. P., & Kumar, B. R. S. (2011). An Implementation of the User-based Collaborative Filtering Algorithm. IJCSIT) International Journal of Computer Science and Information Technologies, 2(3), 1283-1286.

[2] Hahsler, M. (2011). recommenderlab: A Framework for Developing and Testing Recommendation Algorithms. Nov.

[3] Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (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.

[4] Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook (pp. 1-35). Springer US

[5] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295). ACM.

[6] Shih, Y. Y., & Liu, D. R. (2005, January). Hybrid recommendation approaches: collaborative filtering via valuable content information. In System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on (pp. 217b-217b). IEEE.

[7] Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009, 4