# Movie Recommendation System

Name: Parva Shah, Vraj Shah, Rajiv Patel
Institution: Rutgers University
email:parva.shah3@rutgers.edu,vraj.shah@rutgers.edu,Rp1074@scarletmail.rutgers.edu

## ABSTRACT

There are thousands of movies available to everyone, but a user can only watch few in his lifetime. The purpose of this project is to build a recommender system that would try to predict the taste of the user and recommend the best movies. We have studied different filtering techniques, compared them based on their methods and computed errors. We implemented content-based filtering and item-based collaborative filtering and chose the one, which was more focused. The content-based filtering produced more accurate results, recommending 10 movies to the user, similar to his interests.

## KEYWORDS

Content-based filtering
Item-based Collaborative filtering
KNN algorithm
Root Mean Square Error
Weighted Mean
Recommendation
Cosine Similarity
Pearson Correlation Coefficient
Prediction

## 1 INTRODUCTION

In recent times companies like Netflix, YouTube, Amazon Prime, Hulu etc are always looking forward to increase their profit. Their profit can be increased only if they engage more customers by providing what they want. That's why Item Recommendation Systems come into picture. In above mentioned examples items are nothing but the movies, videos or T.V. series. We built recommendation model which recommends top 10 movies to user (which user has not seen till now) based on ratings he/she had given in past.
This project has two parts. The first part is about recommending movies to the user. The second part is evaluating the model by calculating RMSE (Root Mean Squared Error). For the experiment purpose we used two of the very famous algorithms namely 1) Content Based Filtering, 2) Item-Item Collaborative Filtering. We got RMSE of 0.91 and 1.04 respectively.

*Conference'17, July 2017, Washington, DC, USA*
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 2 RELATED WORK

In building recommender system our first task was to pre-process given data (MovieLens). Data pre-processing consists of two subtasks.

(1) Data Cleaning
(2) Splitting data into training and testing data

**Data Cleaning:** Given dataset did not have any missing values, so we did not have to deal with any missing values.
**Splitting Data:** For each user, we randomly selected 80% of his/her ratings as training ratings and remaining 20% ratings as testing ratings. At last we clubbed training ratings of all users for training data and same for testing data. This can be depicted as following.

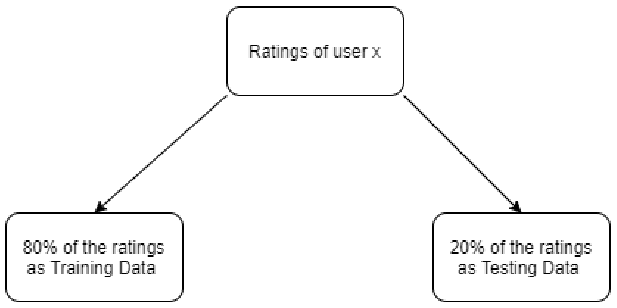

Ratings of user x
80% of the ratings as Training Data
20% of the ratings as Testing Data

**Figure 1: Splitting Data**

We have used following files from the given dataset.

(1) Rating.csv: This file contains all the ratings given by all the users. It has following columns [userId, movieId, rating, timestamp]
(2) Movies.csv: This file contains all the movies and their metadata. It has following columns [movieId, title, genre]

We developed this model using standard libraries available in python like Numpy, Pandas, flask etc.
We used the article from the below link to divide the data and find pearson correlation coefficient.

https://towardsdatascience.com/how-to-build-a-simple-recommender-system-in-python-375093c3fb7d

After completion of our core model, instead of displaying output on console we developed UI where you can search using User ID and recommended movies to that user will be displayed on that page. To develop this we have used template which is available on internet. (https://colorlib.com/preview/theme/hostza/).
To develop this we built client-server architecture using flask library. Below are the screenshots of our system.

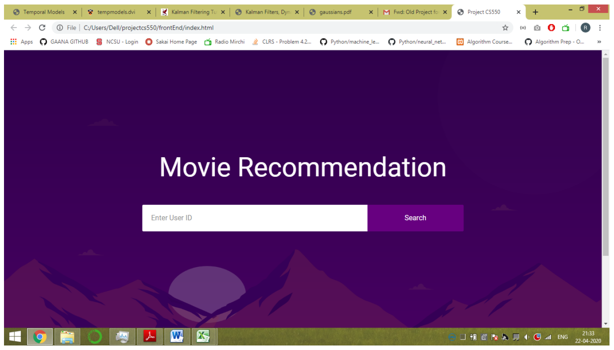
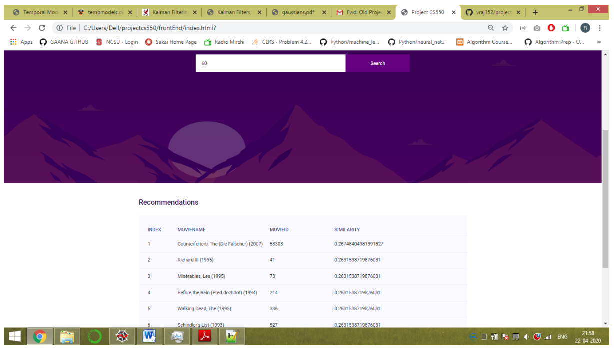


**Figure 2: Movie Recommendation**

When you enter userID and click on search, it will call server and server will send response which will be rendered here.



Recommendations

| INDEX | MOVIENAME | MOVIEID | SIMILARITY |
|---|---|---|---|
| 1 | Counterfeiters, The (Die Fälscher) (2007) | 58303 | 0.26748404981391827 |
| 2 | Richard III (1995) | 41 | 0.2631538719876031 |
| 3 | Misérables, Les (1995) | 73 | 0.2631538719876031 |
| 4 | Before the Rain (Pred dozhdot) (1994) | 214 | 0.2631538719876031 |
| 5 | Walking Dead, The (1995) | 336 | 0.2631538719876031 |
| 6 | Schindler's List (1993) | 527 | 0.2631538719876031 |

**Figure 3: Movie Recommendation**

## 3 PROBLEM FORMALIZATION

For implementing content-based and item-based approaches, we have used various algorithms and mathematical equations for computing the similarity and predicting the ratings of movies.

**Similarity:**

**Cosine Similarity:** Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. If the two vectors are completely alike, the angle between them is 0 (cos 0 = 1).

$$\cos(\theta) = \frac{\overrightarrow{a} \,.\, \overrightarrow{b}}{\|\overrightarrow{a}\| \|\overrightarrow{b}\|}$$

$$= \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\sqrt{\sum_1^n b_i^2}}$$

where $\overrightarrow{a} \,.\, \overrightarrow{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + ... + a_n b_n$ is the dot product of two vectors.

We used this to compute similarities between user profile and item profile in content-based filtering.

**Pearson Correlation Coefficient:** This statistic is used to measure linear correlation between two variables. This number will lie between -1 and 1. 1 indicates a positive linear correlation while -1 indicates a negative correlation. 0 indicates no linear correlation. Therefore, movies with a zero correlation are not similar at all.

$$S(i, j) = corr_{i,j}$$

$$= \frac{\sum_{u \in U}(R_{u,i} - \bar{R_i})(R_{u,j} - \bar{R_j})}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R_i})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R_j})^2}}$$

$R_{u,i}$ = Rating of user u on item i
$\bar{R_i}$ = Average rating of the $i^{th}$ item
U = Set of all the users
The above equation is used to compute similarity between item i and item j in item-based filtering.

**KNN Algorithm:** This is the algorithm used in both content-based and item-based filtering technique to predict the rating of a movie by a particular user. It simply means selecting 'N' number of things similar to what you are interested in.

*Step 1:* Find similarity scores between the movie and all the movies user has rated
*Step 2:* Sort and choose top 20 (depends on problem) nearest neighbors of the movie
*Step 3:* Take the weighted average and predict the ratings

**Root Mean Squared Error:** RMSE compares a predicted value and an observed or known value. The smaller an RMSE value, the closer predicted and observed values are. First, the movies that are present in the testing data are predicted using above algorithms. That value is then compared to the actual user ratings to find the error.

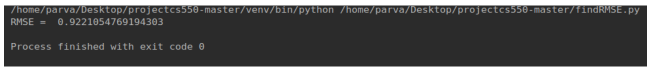
$$RMSE = \sqrt{\frac{\sum_{i=1}^n (P_i - O_i)^2}{n}}$$

P = Predicted Value
O = Observed Value
n = Total number of data

## 4 EXPERIMENTS

We have implemented two famous recommendation system methods – content-based filtering and item based collaborative filtering. We compared the RMSE error for both the systems and have reached a conclusion.
The biggest advantage of using content-based filtering is that you can create user profile for each individual user which takes into account their likes and dislikes. This makes the process of recommending easier and more focused. In the database, we had access to the genre of each movie that helped us to determine the kind of

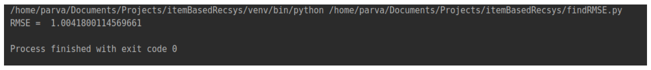
movies that the user preferred to watch. We assigned weights to each category of movie, designed to represent a particular user. This helped in achieving pretty close predictions with RMSE of 0.92 on the testing data.



```
/home/parva/Desktop/projectcs550-master/venv/bin/python /home/parva/Desktop/projectcs550-master/findRMSE.py
RMSE =  0.9221054769194303

Process finished with exit code 0
```

We also decided to implement item-item based collaborative filtering. This was done to compare to the prior algorithm and use whichever produced lowest errors. The RMSE we got is 1.004 as shown in the below figure.



```
/home/parva/Documents/Projects/itemBasedRecsys/venv/bin/python /home/parva/Documents/Projects/itemBasedRecsys/findRMSE.py
RMSE =  1.0041800114569661

Process finished with exit code 0
```

There are several factors contributing to the high error. First and biggest of all is that there are many movies which are rarely watched and hence have been rated by only a small number of users. In fact, there are some movies which have been watched by only one or two users. This misleads the similarities between movies. In order to avoid such a bias, these movies need to be removed from our consideration of recommending to the user. Hence, there are several movies which will never be recommended to the user.
We have tried many such threshold values. However, we got better results when threshold value is equal to 50. This means that all the movies whose frequency is less than 50 will be ignored while measuring similarity. We can further experiment by changing this threshold value.

## 5 THE PROPOSED MODEL

o recommend top 10 movies based on user's watch history, we have implemented widely used algorithm called "Content Based Filtering". Idea behind this approach is, system understands user's preferences from the history of particular user and recommend similar items to the user. This can be explained using below figure.

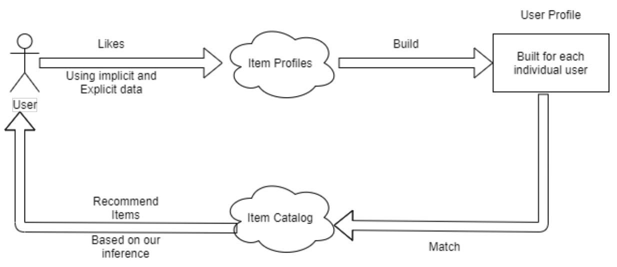
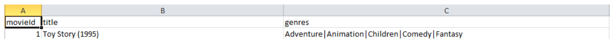


**Figure 4**

**HOW TO BUILD ITEM PROFILE?**

As we can see in above diagram we first build item profiles for each and every item available in system. Item profile is nothing but the feature(s) of individual item. So, in our case features of the item i.e. movie would be genres of the movie. For example consider movie having ID 1.

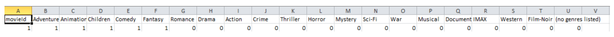
**Eg:** This movie has 5 genres namely adventure, animation, children, comedy and fantasy. So this would be our features of this particular movie.



| A | B | C |
|---|---|---|
| movieId | title | genres |
| 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy |

**Figure 5**

Item profile would look like this. We have taken 1 in genres which are present in this movie else 0. Below image shows item profile for above mentioned movie.



| A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|
| movieId | Adventure | Animation | Children | Comedy | Fantasy | Romance | Drama | Action | Crime | Thriller |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

| L | M | N | O | P | Q | R | S | T | U | V |
|---|---|---|---|---|---|---|---|---|---|---|
| Horror | Mystery | Sci-Fi | War | Musical | Document | IMAX | Western | Film-Noir | (no genres listed) | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**Figure 6**

**HOW TO BUILD USER PROFILE?**

User profiles are nothing but the weights of each feature for individual user. Let's say user has rated items with profiles (i1,i2,...,in). They are vectors in high dimensions. (As we have explained earlier. It has exactly same number of dimensions as number of features. In our case dimensions would be 20, because we have 20 unique genres.)

**Idea 1:** To build user profile we take weighted average of rated item profile. But by doing so we are neglecting the actual rating given by the user to specific movie. So, we have come up with following strategy.

**Idea 2:** We are normalizing the rating by subtracting user's mean rating. Consider following example.

Example - Suppose user x has watched 5 movies. Among them 2 movies featuring actor A and 3 movies featuring actor B. User has rated actor A's movies 3 and 5; and actor B's movies 1,2 and 4. Now we will subtract each rating by user's mean rating i.e. 3. So, actor A's normalized rating would be (0) and (+2) and actor B's normalized rating would be (-2),(-1) and (1). Profile weight for actor A: 0+2 / 2 = 1 Profile weight for actor B: -2-1+1 / 3 = -2/3 Thus, we are capturing the intuition that user has mild preference to actor A with compare to actor B.

**MAKING PREDICTION:**

Consider the item profile i and user profile x. Now these both are vectors in high dimensions. In order to make prediction we need to find similarity between these two vectors. For that we have several similarity measures but we have used cosine similarity for this approach. The more the angle smaller the cos, hence lower the similarity between two vectors. And as angle decreases between two vectors cos's value become larger, hence similarity between two vectors is higher.

**FINAL STEP:**

So, we calculate cosine similarity between each item in item catalogue and we pick items with highest cosine similarity and recommend those to user.

## 6 CONCLUSIONS AND FUTURE WORK

For item-based filtering, we can try to gather more data on certain movies whose frequency is very low. By having more data, we can make the predictions more accurate and reduce the error. Also, we can always play around the threshold value to obtain better results. For content-based filtering, we can increase the number of features which are taken into consideration. In our project, we have only included the genre of the film. We can consider more factors like directors, actors and even the user ratings that the particular user has written (if any). The user ratings will help us to understand about user's choice and what factors affect his inclination towards the movie. This will make the user profile more solid and make the predictions more accurate.
We can obtain the user ratings by web scrapping the IMDB. Based on the movie title we can search for user and critique reviews and make our database even larger. We can introduce Machine Learning and Natural Language Processing to analyze the reviews.

## REFERENCES

https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada
http://www.cs.carleton.edu/cscomps/0607/recommend /recommender /itembased.html
https://medium.com/fnplus/content-based-recommendations-ffb221931485
https://towardsdatascience.com/how-to-build-a-simple-recommender-system-in-python-375093c3fb7d