**Project Report**

**Moeen Ahmad**

**Haider Ali**

**Muhammad Ramish**

**Sheraz**

**Movie Recommender System**

**Abstract:**

In this project, we attempt to understand the different kinds of recommendation systems and compare their performance on the **MovieLens** dataset. .We attempt to build a scalable model to perform this analysis. We start by preparing and comparing the various models on a smaller dataset of 50,000 ratings. Then, we try to scale the algorithm so that it is able to handle large dataset. We find that for the smaller dataset, using user-based and item based collaborative filtering results in the lowest MeanSquaredError on our dataset.

**Introduction and Motivation:**

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user, and make suggests based on these preferences. On the Internet, where the number of choices is overwhelming, there is need to filter, prioritize and efficiently deliver relevant information in order to alleviate the problem of information overload, which has created a potential problem to many Internet users. Recommender systems solve this problem by searching through large volume of dynamically generated information to provide users with personalized content and services.

These have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books and videos, to friends and stories on social media platforms, to products one-commerce websites, to people on professional to search results returned on Google.

Often, these systems are able to collect information about a user’s choices, and can use this information to improve their suggestions in the future. For example, Facebook can monitor your interaction with various stories on your feed in order to learn what types of stories appeal to you.

**Literature Background of the art:**

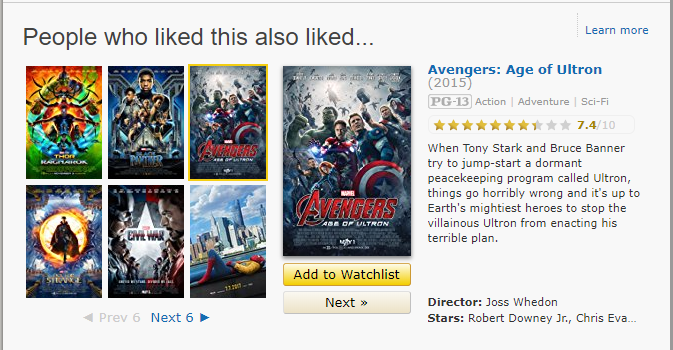
Three main approaches are widely used for movie recommender systems.

* Content-based filtering
* Collaborative filtering
* Item based collaborative filtering
* User based collaborative filtering
* Deep Learning /Neural Network

In IEEE paper in 2003, item-based collaborative filtering was widely deployed across **Amazon**.com. The homepage prominently featured recommendations based on your past purchases and items browsed in the store. Search result pages recommended items related to your search. Using e-mails, browse pages, product detail pages, and more, many pages on Amazon.com had at least some recommended content, starting to approach a store for every customer.

Others have reported using the algorithm, too. In 2010, **YouTube** reported using it for recommending videos. Many open source and third-party vendors included the algorithm, and it showed up widely in online retail, travel, news, advertising, and more. Similarly, **Netflix** used recommender systems so extensively that their Chief Product Officer, Neil Hunt, indicated that more than 80 percent of movies watched on Netflix came through recommendations, and placed the value of Netflix recommendations at more than US$1 billion per year. **IMDB** are use hybrid collaborative filter which is the combination of content based filter and collaborative filter. And in collaborative filter imdb use user-user collaborative filter and item-item collaborative filter to suggest the movies to you according to your taste.

So IMDB ,Netflix and Youtube are also use these methods to recommend the movies to user by actors and genre. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience.



**DATASET:**

We used the MovieLens movie ratings dataset for our experiments. The experiments are conducted on two versions of the dataset. The first version consists of 100004 ratings by 671 users across 9125 movies. The ratings allowed at intervals of 0.5 on a 5-point scale, starting from 0.5 and going to 5. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided. The dataset has additional information about the movies in the form of genre and tags; however we use only the ratings given by the users to the movies and ignore the other information for the collaborative filtering techniques. The second and bigger version of the dataset consists of 26,000,000 (20 million) ratings by 270,000 users across 45,000 movies. Apart from this, the structure of the two datasets also is identical. The movie ids for a particular movie are the same in both datasets, but the user id for the same user is different for the two datasets.

**Proposed Algorithms:**

The proposed algorithms are as below.

* **Collaborative Filtering:**

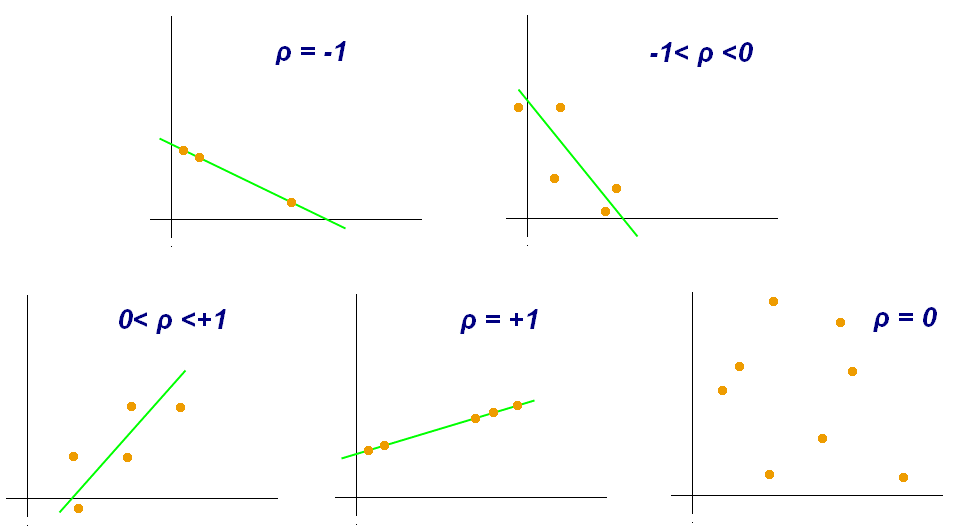
Collaborative Filtering techniques make recommendations for a user based on ratings and preferences data of many users. The main underlying idea is that if two users have both liked certain common items, then the items that one user has liked that the other user has not yet tried can be recommended to him.

We have used the **Nearest Neighbours Algorithm**

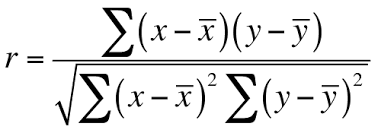
**Nearest Neighbours Collaborative Filtering**: This approach relies on the idea that users who have similar rating behaviours so far, share the same tastes and will likely exhibit similar rating behaviours going forward. The algorithm first computes the similarity between users by using the row vector in the ratings matrix corresponding to a user as a representation for that user. The similarity is computed by using **cosine similarity** and **Pearson R**. In order to predict the rating for a particular user for a given movie j, we find the correlation with other movies and based on the threshold value we recommend the movie to the user.

* **Pearson correlation coefficient:**

Pearson's correlation coefficient is the covariance of the two variables divided by the product of their standard deviations. is a measure of the linear correlation between two variables *X* and *Y*. It has a value between +1 and −1, where 1 is total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation. It is widely used in the sciences. It was developed by **Karl Pearson** from a related idea introduced by **Francis Galton** in the 1880s .



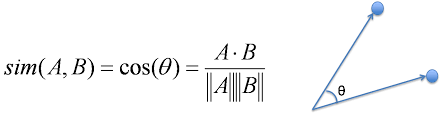
The **item based collaborative filtering** uses Pearson R to calculate the correlation between every two movies and that allows us to group similar movies together. The formula below calculates the correlation represented by r between two items(movies).



* **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. The cosine of 0° is 1, and it is less than 1 for any other angle in the interval [0,2π). It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude.

The **user based collaborative filtering** uses Cosine Similarity to calculate the similarity between every two users and that allows us to recommend movies between them. If the similarity measure is close to **1** than their is a strong relation between two users behaviour toward the movie. The formula below calculates the cosine similarity represented by sim between two users.

****

Where A represents the rating given by A user and B represents the rating given by B user.

* **Content Based Recommendations:**

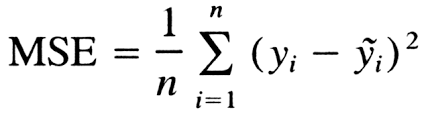
Content Based Recommendation algorithm takes into account the likes and dislikes of the user and generates a User Profile. For generating a user profile, we take into account the item profiles (vector describing an item) and their corresponding user rating. The user profile is the weighted sum of the item profiles with weights being the ratings user rated. Once the user profile is generated, we calculate the similarity of the user profile with all the items in the dataset, which is calculated using cosine similarity between the user profile and item profile. Advantages of Content Based approach is that data of other users is not required and the recommender engine can recommend new items which are not rated currently, but the recommender algorithm doesn’t recommend the items outside the category of items the user has rated.

* **Person name search:**

In addition, we also developed a separate search for names of people associated with the movie. These could be the actors, the directors, or the writers. Both searches are independent, in the fact that using the person search, you will not get results for queries appearing in the movie plots, and visa-versa. We display 10 appropriate results returned by the query.

**Evaluation** :

We use the mean squared error metric to evaluate the predictions made by our system. The mean squared error is computed as



where N is the number of ratings in the test partition, yi is the predicted rating for user i and movie j and y~i is the actual rating.

* **Experimental setup**
* We are use python language thats why we need to install the compiler of python or online compiler e.g(AWS account, toturialpoint,etc).
* Import all libraries which are used in this project like panda, cosin\_similarity , surprise, numpy, metrics etc.
* Load the dataset in the project using read\_csv method.
* Divide the data in different genre and users to train the data
* Make the user profile for user in the test set.
* Apply the content based filter and collaborative filter together to create the **hybrid model** for best results because its perform better.
* In content based filter we built a movie description recommender. We use movies description and taglines. And check the mechine performance by using qualitative method.
* we have used the TF-IDF Vectorizer, calculating the Dot Product will directly give us the Cosine Similarity Score as qualitative method. Therefore, we will use sklearn's **linear\_kernel** instead of cosine\_similarities since it is much faster.
* Now in collaborative filter i used **svd()** function to minimise **RMSE** (Root Mean Square Error) and give great recommendations of surprise library.
* Gird is created in this model and for each combination the hybrid model is trained.
* Now the model is ready for test set of user profiles.
* **Expected Results and Analysis**
* **Small data set results(100k)**

Table 1 and Table2 represents the performance of User-User and Item- Item Collaborative filtering with the hyperparameter as the number of nearest neighbors(k) on MovieLens 100k(small) data. The best result corresponds to k=300, as we increase the k to 500 the performance reduces, as we are taking into account almost all the users (500/670) for predicting the rating. When we increase the number of neighbors to 500, even the users with very small similarity values will be included.

Table 1: User-User Collaborative Filtering for 100k

|  |  |
| --- | --- |
| **K Nearest User** | **MSE** |
| **5** | **1.0144** |
| **10** | **0.9980** |
| **20** | **0.9681** |
| **50** | **0.9074** |
| **100** | **0.8660** |
| **200** | **0.8426** |
| **300** | **0.8366** |
| **500** | **1.0701** |

Table 2: Item-Item Collaborative Filtering for 100k

|  |  |
| --- | --- |
| **K Nearest Movies** | **Accuracy** |
| **5** | **90%** |
| **10** | **96%** |
| **20** | **87%** |
| **30** | **87%** |
| **40** | **85%** |
| **50** | **83%** |

* **Large Dataset (10M):**

Table 3 and Table 4 represents the performance of User-User and Item- Item Collaborative filtering with the hyperparameter as the number of nearest neighbors(k) on MovieLens 10M(large) data. The best result corresponds to k=500.

Table 3: User-User Collaborative Filtering for 10M

|  |  |
| --- | --- |
| **K Nearest User** | **MSE** |
| **20** | **0.8462** |
| **50** | **0.8006** |
| **100** | **0.7664** |
| **200** | **0.7345** |
| **500** | **0.7044** |

Table 4: Item-Item Collaborative Filtering for 10M

|  |  |
| --- | --- |
| **K Nearest Movies** | **Accuracy** |
| **15** | **92%** |
| **20** | **87%** |
| **25** | **93%** |
| **40** | **88%** |
| **70** | **87%** |

* **Work breakdown**
* First of all we got the idea of this project from IMDB,NetFlix and Amazon.
* Did research about what imdb,Netflix are using to implement this task.
* Then select the hybrid filter for this project. It is a combination of two filters content based filter and collaborative filter.
* Collaborative filter also have a user based collaborative filter and item based collaborative filter.
* For implementation, selected python.
* Created environment for python programming i.e. used several libraries of python.
* Done training of data.
* Then tested on some sample data.
* **Possible Future Work :**

There are plenty of way to expand on the work done in this project. Firstly, the content based method can be expanded to include more criteria to help categorize the movies. The most obvious ideas is to add features to suggest movies with common actors, directors or writers. In addition, movies released within the same time period could also receive a boost in likelihood for recommendation. Similarly, the movies total gross could be used to identify a users taste in terms of whether he/she prefers large release blockbusters, or smaller indie films. However, the above ideas may lead to over fitting, given that a users taste can be highly varied, and we only have a guarantee that 20 movies (less than 0.2%) have been reviewed by the user. In addition, we could try to develop hybrid methods that try to combine the advantages of both content-based methods and collaborative filtering into one recommendation system.

* **References:**

[1] A Survey of Collaborative Filtering Techniques; Su et al; https://www.hindawi.com/journals/aai/2009/421425/

[3] Intro to Recommender Systems: Collaborative Filtering; [http://blog.ethanrosenthal.com/2015/11/02/intro-to-collaborative -filtering/](http://blog.ethanrosenthal.com/2015/11/02/intro-to-collaborative%20-filtering/)

<https://www.kaggle.com/rounakbanik/movie-recommender-systems#_=_>

<https://beta.vu.nl/nl/Images/werkstuk-postmus_tcm235-877824.pdf>