

Movie Recommendation System

PYTHON

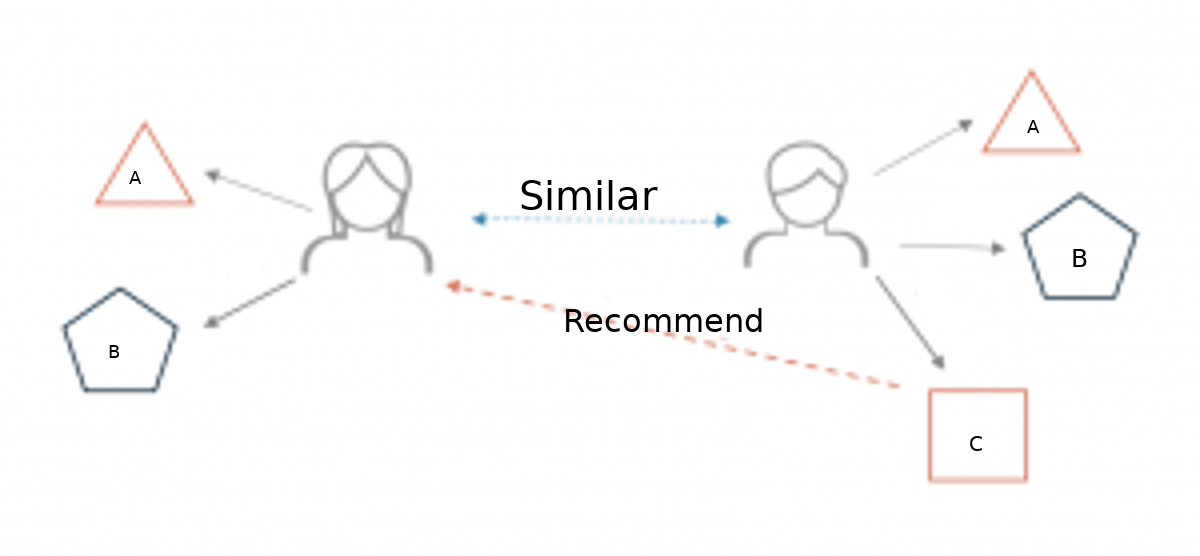
By Sweta Agarwal | Submitted To- Mr. Akshay Rajputs |20-06-2021

**Project Report**

**ABSTRACT**

A recommender system is a simple algorithm whose aim is to provide the most relevant information to a user by discovering patterns in a dataset. The algorithm rates the items and shows the user the items that they would rate highly. An example of recommendation in action is when you visit Amazon and you notice that some items are being recommended to you or when Netflix recommends certain movies to you. They are also used by Music streaming applications such as Spotify and Deezer to recommend music that you might like.

Below is a very simple illustration of how recommender systems work in the context of an e-commerce site.



Two users buy the same items A and B from an e-commerce store. When this happens the similarity index of these two users is computed. Depending on the score the system can recommend item C to the other user because it detects that those two users are similar in terms of the items they purchase.

**Different types of recommendation engines**

The most common types of recommendation systems are **content-based** and **collaborative filtering** recommender systems. In collaborative filtering, the behavior of a group of users is used to make recommendations to other users. The recommendation is based on the preference of other users. A simple example would be recommending a movie to a user based on the fact that their friend liked the movie.There are two types of collaborative models **Memory-based**methods and **Model-based** methods.The advantage of memory-based techniques is that they are simple to implement and the resulting recommendations are often easy to explain. They are divided into two:

* **User-based collaborative filtering**: In this model, products are recommended to a user based on the fact that the products have been liked by users similar to the user. For example, if Derrick and Dennis like the same movies and a new movie come out that Derick like,then we can recommend that movie to Dennis because Derrick and Dennis seem to like the same movies.
* **Item-based collaborative filtering**: These systems identify similar items based on users’ previous ratings. For example, if users A,B, and C gave a 5-star rating to books X and Y then when a user D buys book Y they also get a recommendation to purchase book X because the system identifies book X and Y as similar based on the ratings of users A,B, and C.

Model-based methods are based on Matrix Factorization and are better at dealing with sparsity. They are developed using data mining, machine learning algorithms to predict users’ rating of unrated items. In this approach techniques such as dimensionality reduction are used to improve accuracy. Examples of such model-based methods include Decision trees, Rule-based Model, Bayesian Model, and latent factor models.

**Content-based systems** use metadata such as genre, producer, actor, musician to recommend items say movies or music. Such a recommendation would be for instance recommending Infinity War that featured Vin Diesel because someone watched and liked The Fate of the Furious. Similarly, you can get music recommendations from certain artists because you liked their music. Content-based systems are based on the idea that if you liked a certain item you are most likely to like something that is similar to it.

**Datasets to use for building recommender systems**

In this tutorial, we are going to use the tmdb.csv Data Set. This dataset was put together by the Grouplens research group at the University of Minnesota. It contains 1, 10, and 20 million ratings. Movielens also has a website where you can sign up, contribute reviews and get movie recommendations.

**Data Preprocessing**

We are going to use the movielens to build a simple item similarity-based recommender system. The first thing we need to do is to import pandas and numpy.

**importpandasaspd**  
**importnumpyasnp**  
**importwarnings**  
warnings.filterwarnings('ignore')

Next, we load in the data set using pandas read\_csv() utility. The dataset is tab separated so we pass in \t to the sep parameter. We then pass in the column names using the names parameter.

df = pd.read\_csv('u.data', sep='**\t**', names=['user\_id','item\_id','rating','titmestamp'])

Now let’s check the head of the data to see the data we are dealing with.

df.head()

It would be nice if we can see the titles of the movie instead of just dealing with the IDs. Let’s load in the movie titles and merge it with this dataset.

movie\_titles = pd.read\_csv('Movie\_Titles')

movie\_titles.head()

Since the item\_id columns are the same we can merge these datasets on this column.

df = pd.merge(df, movie\_titles, on='item\_id')

df.head()

Let’s look at what each column represents:

* user\_id - the ID of the user who rated the movie.
* item\_id - the ID of the movie.
* rating - The rating the user gave the movie, between 1 and 5.
* timestamp - The time the movie was rated.
* title - The title of the movie.

ratings = pd.DataFrame(df.groupby('title')['rating'].mean())

ratings.head()

Next we would like to see the number of ratings for each movie. We do this by creating a number\_of\_ratings column. This is important so that we can see the relationship between the average rating of a movie and the number of ratings the movie got. It is very possible that a 5-star movie was rated by just one person. It is therefore statistically incorrect to classify that movie has a 5-star movie. We will, therefore, need to set a threshold for the minimum number of ratings as we build the recommender system. In order to create this new column, we use pandas groupby utility. We group by the title column and then use the countfunction to calculate the number of ratings each movie got. Afterward we view the new data frame by using the head() function.

ratings['number\_of\_ratings'] = df.groupby('title')['rating'].count()  
ratings.head()

movie\_matrix = df.pivot\_table(index='user\_id', columns='title', values='rating')  
movie\_matrix.head()

Next let’s look at the most rated movies and choose two of them to work with in this simple recommender system. We use pandas sort\_values utility and set ascending to false in order to arrange the movies from the most rated. We then use the head() function to view the top 10.

ratings.sort\_values('number\_of\_ratings', ascending=**False**).head(10)

Let’s assume that a user has watched Air Force One (1997) and Contact (1997). We would like to recommend movies to this user based on this watching history. The goal is to look for movies that are similar to Contact (1997) and Air Force One (1997 which we shall recommend to this user. We can achieve this by computing the correlation between these two movies’ ratings and the ratings of the rest of the movies in the dataset. The first step is to create a dataframe with the ratings of these movies from our movie\_matrix.

AFO\_user\_rating = movie\_matrix['Air Force One (1997)']  
contact\_user\_rating = movie\_matrix['Contact (1997)']

SO, that was some important Knowledge of the data that I have processed for the recommendation engine. So lets do not dive more deep for the code.

## PROJECT MOTIVATION AND METHODOLOGY

This system can be improved by building a Memory-Based Collaborative Filtering based system. In this case, we’d divide the data into a training set and a test set. We’d then use techniques such as cosine similarity to compute the similarity between the movies. An alternative is to build a Model-based Collaborative Filtering system. This is based on matrix factorization. Matrix factorization is good at dealing with scalability and sparsity than the former. You can then evaluate your model using techniques such as Root Mean Squared Error (RMSE).