GETFLIX: A Movie Recommendation System

Asmit Singh
asmit18025@iiitd.ac.in

1.Introduction

Movies have become an unremovable part of our lives.With companies like NETFLIX and Amazon Prime, watching latest movies has become easier.
There is a huge possibility for analyzing the user and item patterns and learning from them to provide better movie and entertainment recommendations for the user and create a personalized experience. This is a report on my recommendation system.This will contain details about the processes and the methods used in its development.

2.Dataset

IMDB has a very wide corpus of movies that are in millions. I preferred to use the smallest available data set (9000 movies) available at Movie Lense with 600 users who rated the movies. This data set seemed good for this purpose at it had a usable amount of movie data that was required and the user rating data was also sufficient for my dummy users purposes.

I downloaded the required ZIP folder and it contained the following files:
1. Movies.csv : movieId , Title and Genres
2. Links.csv : movieId , imdbId ,tmdbId
3. Ratings.csv :movieId ,userId, ratings
4. tags.csv

I did not find tags.csv to be helpful therefore i decided to leave it

Data Cleaning

I decided to work on a smaller data set of 9000 movies and decided to choose a varied corpus of 500 movies from the above dataset.
Using pandas I joined the movies data and links data

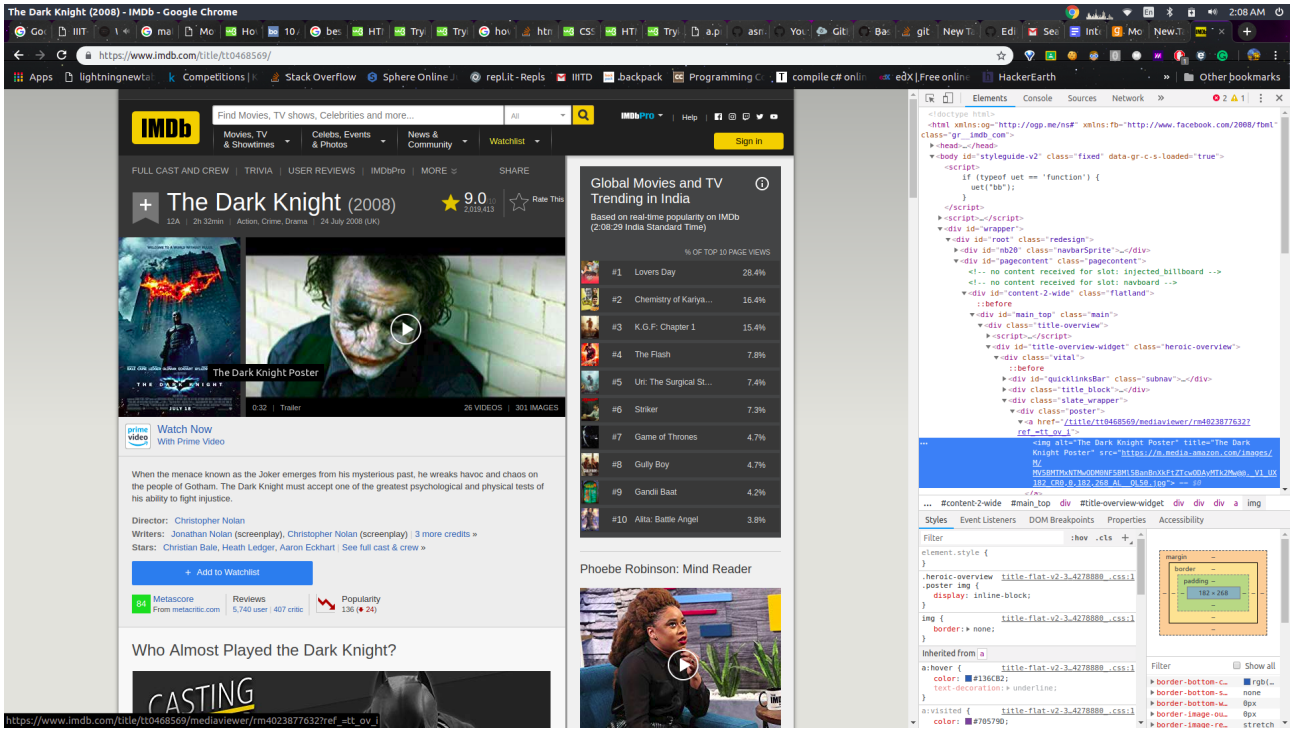
Using pandas I retrieved a random selection of 500 movies which had a variety in their genres.

The ratings data also needed to be updated as there were many users who had rated movies which were not in my new selection of 500 movies.
Using Pandas I removed instances from the ratings data having movies that were not there in my 500 movies.This ensured creation of an entirely unique user rating dataset which varied in the number and types of movies that were rated

My database of choice was MongoDB.The above data was stored in mongodb(mlab). In two collections the Movies data and the ratings data.

Data scraping

Since the movie poster were not already provided in the dataset.I decided to scrape the required tthumbnails from the dedicated movie pages using the imdbid
in the links data.
Using the beautifulsoup library i was able to scrape the posters links and store them in my mongo movie data collection.



Now I had the required data to use .

3.Finding Recommendations

User-User collaborative filtering:

The idea behind user-user collaborative filtering is finding users that have similar taste in movies like you.This can be done by comparing each person and finding a similarity score.
The similarity score can be calculated through various methods the ones i cam eacross were Euclidean distance and Pearson correlation score

In euclidean distance approach we tend to find the users which are at minimum distance in terms of ratings but this doesnot take care of cases where some users are more harsh with there rating system ie. For user 1 3 is a score for an average movie but it is possible that user 2 is more harsh and for him 2 is an average score.
For this a more complicated approach is required. We try to fit a correlation line between users and get a correlation coefficient to compare how similar two users are.

$$sim(u,v) = \frac{\sum(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum(r_{ui} - \bar{r}_u)^2}\sqrt{\sum(r_{vi} - \bar{r}_v)^2}}$$

Thus I found Pearson’s coefficient better than Euclidian distance for this problem and decided to go with it.

Now, that we know how similar two users are we need to recommend items.One possible way is to find the top matched user and returning its top rate movies but this is not reliable because there may be movies that were rated by everyone as bad but the top matched user gave it good ratings.For this we need a weighted score that can rank the users.

We can weight the movie scores by multiplying the similarity scores of the users with the ratings they gave to the movies, this will give higher weightage to users that are similar to me.Dividing this by sum of all similarities that reviewed the movie will ensure that the score is normalised and is not biased on the frequency of movies in the ratings list.

Now we have the recommendations.

Item-Item collaborative filtering:

Here instead of finding similar users we find similar items in terms of the ratings given by the user .Same similarity measure (Pearson's coefficient) is used. Once we have the data for similar movies we need to find the movies similar to the movies the user has rated.Among these similar movies we can find the most similar movies by multiplying the rating given by the user and the similarity score of the rated and recommended movies.Now we can choose the top similar movies.

User-User vs Item-Item:

User-User algo is better for smaller data sets which have a good amount of ratings given by the user and it requires a large and varied amount of user ratings for good recommendations.
Item-Item is better for bigger datasets and also it requires less computation because once computed the similar items data can be stored and reused.
<u>In my case user-user recomendations do not always return recommendations due to a small amount of user ratings and inconsistency in ratings data.</u>

Why I chose k=4:

I decided to get atleast 4 movies rated by the user since the user ratings that I had ,had an average of 5-6 movies rated per user. Also we need to take care about the number of movies recommended by the algos. I observed that item-item recommendations always safely returned 4 movies for a movie dataset of size approx. 500.

4.User Interface

I decided to use flask as my framework for creating a web app and used HTML/CSS.

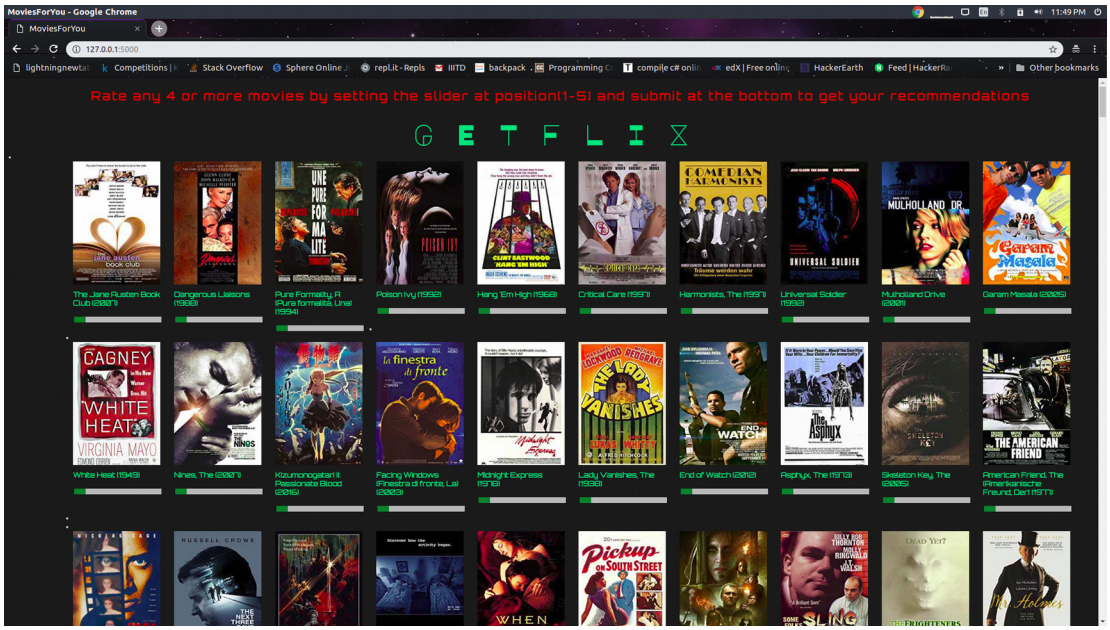


Illustration 1: Getting user ratings

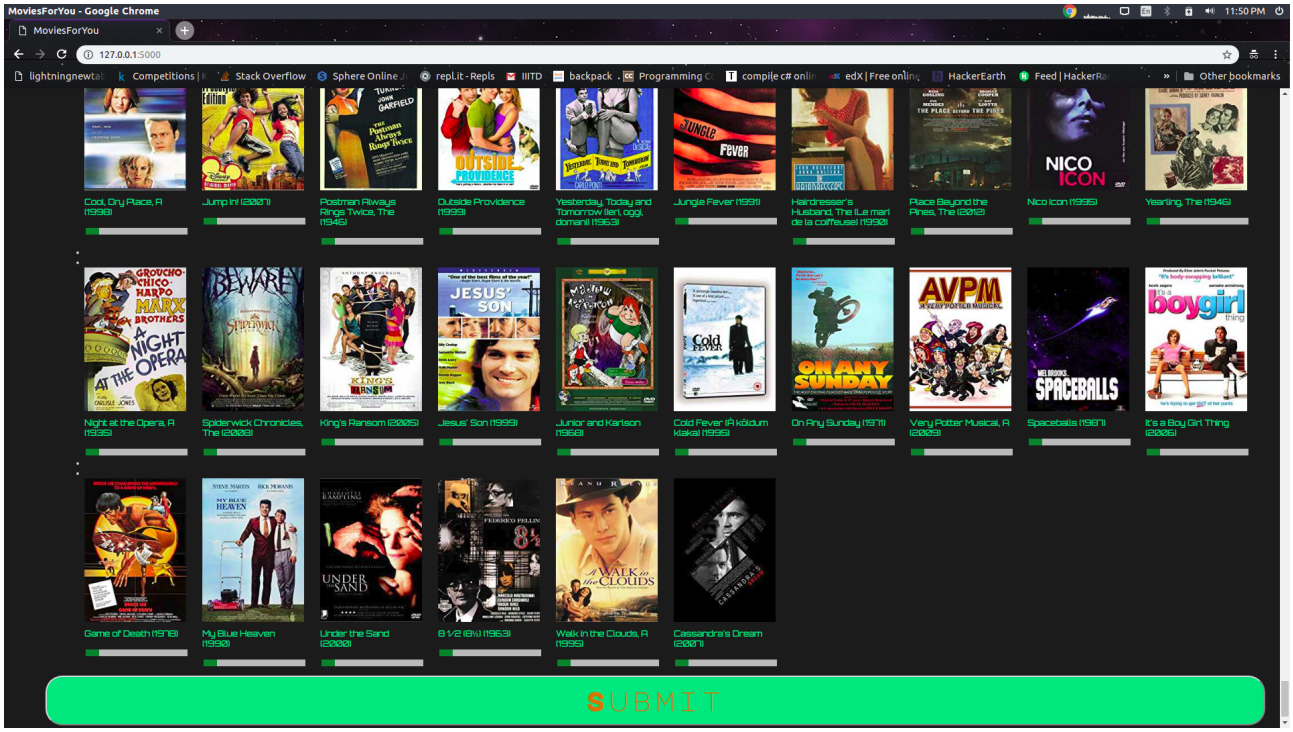


Illustration 2: submit button

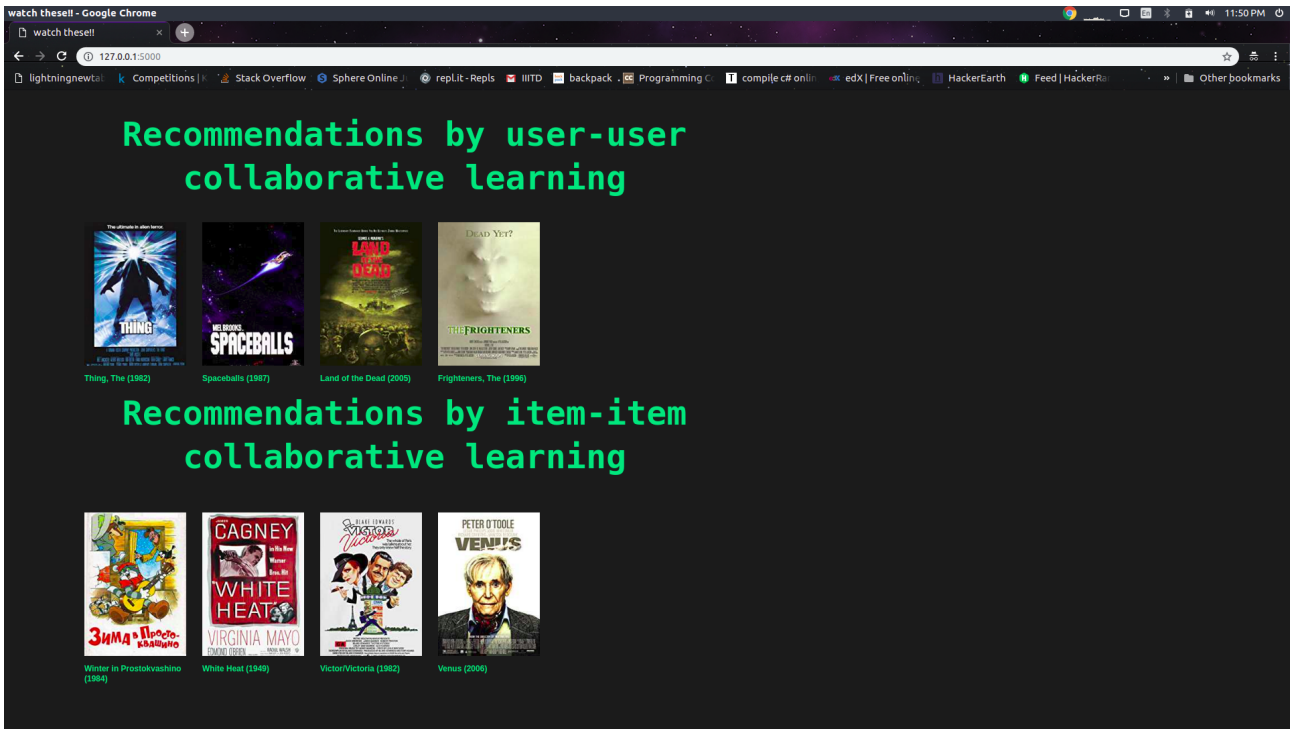


Illustration 3: recommendations