Movie Recommender Systems

Mohit Sagar (ms3036) Pulak Raj (pr446) Kaustubh Nathuram Jadhav(knj25)

1. Introduction

We took this project as an opportunity to explore different aspects of recommender systems. To widen our reach and learn as much as possible, we decided to each implement our own algorithms.

2. NLP based recommender system

We used a pre-trained sentence-transformer **SBERT** to build two types of recommender systems: user-user and item-item. The way SBERT works is that it takes in sentences as input and creates embeddings for those sentences. We can calculate how close these sentences are by calculating their cosine similarities.

2.1.User-User

Implementation

For this task, we used the **MovieLens dataset** and mostly dealt with the “ratings.csv” file. We tried to embed all the users and their movie ratings by using a pre-trained transformer. The idea we came up with was to create a string for each user containing his movie ratings. For each user we created a list of movies and their ratings and sorted them from high to low. To break ties we used lexicographical order. We then created the final string from this list in the format: “movie_id : movie_rating”, for each movie.
For e.g. if user 1 has seen movies A, B, C, D and E with ratings 3, 3, 5, 4.5 and 2 respectively, we will represent the user as:

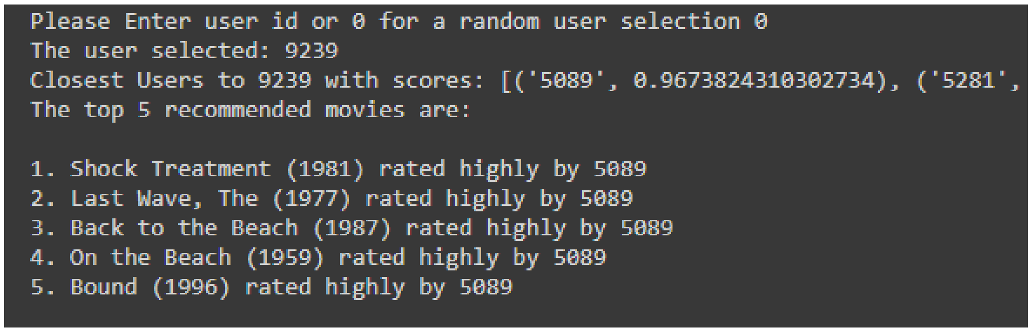
1 = “C:5, D:4.5, A:3, B:3, E:2”

We sorted the movie ratings to exploit the positional encoding done by the transformer. This way two users who have rated the same movies higher will have a higher similarity score than the users who might have rated the same movies but high and low. Another issue we had to keep in mind was the fact that there is a limit to how long the input sentences can be. So, we decided to choose 15 rated movies from each of the following ranges: 0 - 1, 1 - 2, 2 - 3, 3 - 4 and 4 - 5. The reason we decided to get movies from each range is because we want to consider a users’ dislikes as well to find similar users who disliked the same movies.

We then choose a user and compare his sentence with that of all the other users and get cosine similarities. We get the top N users (neighbors) who are similar to our original user and recommend the movies that are top rated by these neighbors. The idea is that since these neighbors are so similar to the user, any movie they rated high would be a good recommendation for the user as well.

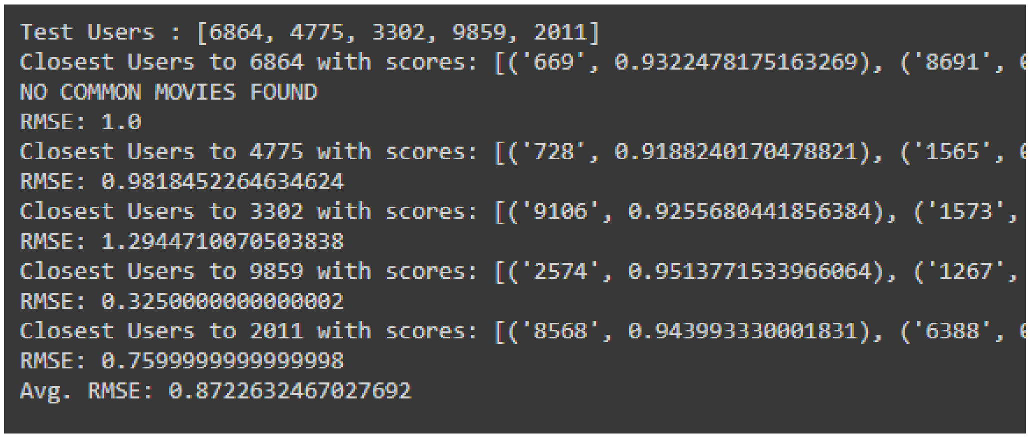
Results

For evaluation purposes, we decided to use RMSE. We implemented an additional feature to predict the ratings for recommended movies. For training and testing, we divided the users movie ratings into 70/30 splits respectively. We used the 70% to find similar users or neighbors for our user and used the rest 30 percent to test and calculate the RMSE. After finding the top N similar neighbors we simply look if those neighbors have seen any of the movies from the testing set. We then rated those movies using the weighted average of ratings from the neighbors and user bias.



```
Please Enter user id or 0 for a random user selection 0
The user selected: 9239
Closest Users to 9239 with scores: [('5089', 0.9673824310302734), ('5281',
The top 5 recommended movies are:

1. Shock Treatment (1981) rated highly by 5089
2. Last Wave, The (1977) rated highly by 5089
3. Back to the Beach (1987) rated highly by 5089
4. On the Beach (1959) rated highly by 5089
5. Bound (1996) rated highly by 5089
```

Fig 1 : Recommendations



```
Test Users : [6864, 4775, 3302, 9859, 2011]
Closest Users to 6864 with scores: [('669', 0.9322478175163269), ('8691',
NO COMMON MOVIES FOUND
RMSE: 1.0
Closest Users to 4775 with scores: [('728', 0.9188240170478821), ('1565',
RMSE: 0.9818452264634624
Closest Users to 3302 with scores: [('9106', 0.9255680441856384), ('1573',
RMSE: 1.2944710070503838
Closest Users to 9859 with scores: [('2574', 0.9513771533966064), ('1267',
RMSE: 0.3250000000000002
Closest Users to 2011 with scores: [('8568', 0.943993330001831), ('6388',
RMSE: 0.7599999999999998
Avg. RMSE: 0.8722632467027692
```

Fig 2 : Evaluations

Conclusion

Even though we used an unconventional approach, we managed to get a decent enough RMSE. A big problem with the evaluation criteria is that there are hardly common rated movies between the user and his neighbors. Intuitively, our system is quite reliable because even with a not so perfect evaluation metric we are able achieve a decent RMSE meaning that the neighbors we are finding are indeed very similar to our user. Therefore, any movies that are rated highly by the neighbors, will be rated decent if not high by our user.

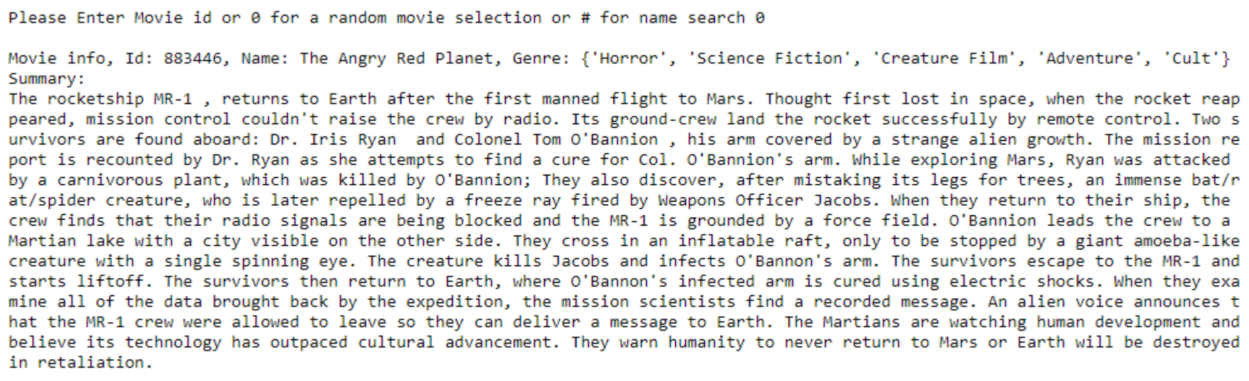
2.2.Item-Item

Implementation

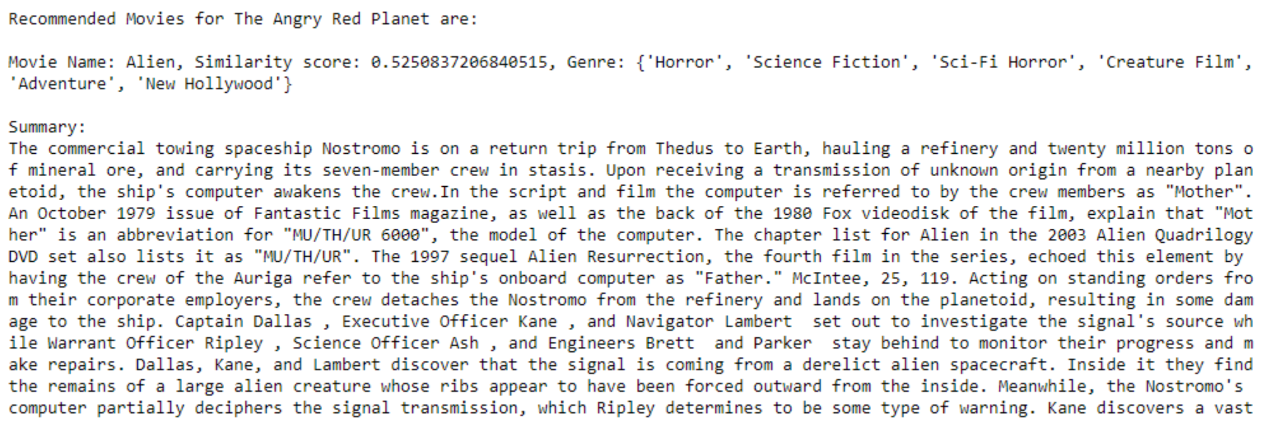
Unlike our last approach, we decided to use the transformers at its full capacity for implementing our item item recommendation engine. We use the plot of a movie and compare it with the plots of all other movies to find similar movies. We again used the sentence transformer to get embeddings for our plots and find similarity using cosine similarity. The MovieLens dataset does not contain movie plots so we had to use a different dataset. We used the **CMU Movie Summary Corpus** dataset which contains over 40,000 movie plots. To minimize the movie comparisons, we decided to only compare the movies with similar genres else comparing 40,000 movie plots line by line will take forever to run even when using the parallel computation power of a gpu.

Results

We couldn’t think of any good evaluation metric for this approach. Since, we are comparing the movie plots using a similarity metric and the fact that our dataset does not have movie ratings, we couldn’t come up with an evaluation criteria that can accurately test our model. Following are the sample output:



```
Please Enter Movie id or 0 for a random movie selection or # for name search 0

Movie info, Id: 883446, Name: The Angry Red Planet, Genre: {'Horror', 'Science Fiction', 'Creature Film', 'Adventure', 'Cult'}
Summary:
The rocketship MR-1 , returns to Earth after the first manned flight to Mars. Thought first lost in space, when the rocket reap
peared, mission control couldn't raise the crew by radio. Its ground-crew land the rocket successfully by remote control. Two s
urvivors are found aboard: Dr. Iris Ryan  and Colonel Tom O'Bannion , his arm covered by a strange alien growth. The mission re
port is recounted by Dr. Ryan as she attempts to find a cure for Col. O'Bannion's arm. While exploring Mars, Ryan was attacked
by a carnivorous plant, which was killed by O'Bannion; They also discover, after mistaking its legs for trees, an immense bat/r
at/spider creature, who is later repelled by a freeze ray fired by Weapons Officer Jacobs. When they return to their ship, the
crew finds that their radio signals are being blocked and the MR-1 is grounded by a force field. O'Bannion leads the crew to a
Martian lake with a city visible on the other side. They cross in an inflatable raft, only to be stopped by a giant amoeba-like
creature with a single spinning eye. The creature kills Jacobs and infects O'Bannon's arm. The survivors escape to the MR-1 and
starts liftoff. The survivors then return to Earth, where O'Bannon's infected arm is cured using electric shocks. When they exa
mine all of the data brought back by the expedition, the mission scientists find a recorded message. An alien voice announces t
hat the MR-1 crew were allowed to leave so they can deliver a message to Earth. The Martians are watching human development and
believe its technology has outpaced cultural advancement. They warn humanity to never return to Mars or Earth will be destroyed
in retaliation.
```



```
Recommended Movies for The Angry Red Planet are:

Movie Name: Alien, Similarity score: 0.5250837206840515, Genre: {'Horror', 'Science Fiction', 'Sci-Fi Horror', 'Creature Film',
'Adventure', 'New Hollywood'}

Summary:
The commercial towing spaceship Nostromo is on a return trip from Thedus to Earth, hauling a refinery and twenty million tons o
f mineral ore, and carrying its seven-member crew in stasis. Upon receiving a transmission of unknown origin from a nearby plan
etoid, the ship's computer awakens the crew.In the script and film the computer is referred to by the crew members as "Mother".
An October 1979 issue of Fantastic Films magazine, as well as the back of the 1980 Fox videodisk of the film, explain that "Mot
her" is an abbreviation for "MU/TH/UR 6000", the model of the computer. The chapter list for Alien in the 2003 Alien Quadrilogy
DVD set also lists it as "MU/TH/UR". The 1997 sequel Alien Resurrection, the fourth film in the series, echoed this element by
having the crew of the Auriga refer to the ship's onboard computer as "Father." McIntee, 25, 119. Acting on standing orders fro
m their corporate employers, the crew detaches the Nostromo from the refinery and lands on the planetoid, resulting in some dam
age to the ship. Captain Dallas , Executive Officer Kane , and Navigator Lambert  set out to investigate the signal's source wh
ile Warrant Officer Ripley , Science Officer Ash , and Engineers Brett  and Parker  stay behind to monitor their progress and m
ake repairs. Dallas, Kane, and Lambert discover that the signal is coming from a derelict alien spacecraft. Inside it they find
the remains of a large alien creature whose ribs appear to have been forced outward from the inside. Meanwhile, the Nostromo's
computer partially deciphers the signal transmission, which Ripley determines to be some type of warning. Kane discovers a vast
```

Conclusion

The results we get from this approach are promising. We can evaluate this approach ourselves by going over the plots of recommended movies and finding how similar they look to us. We believe that this approach when augmented with other item item based algorithms can help us achieve even better recommendations. Moreover, the applications of this approach are not limited to only movie recommendations. It would be a great fit for making e-commerce recommendations as well. We can get similar products by comparing their descriptions and recommend the top rated one.

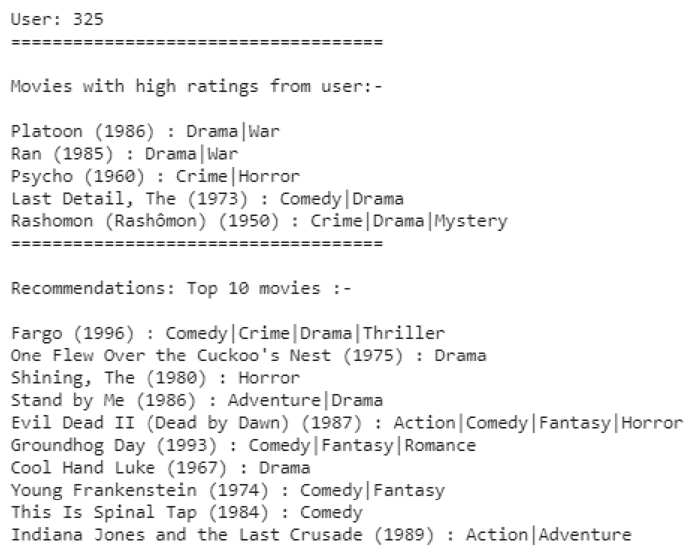
3. User - Item Based Collaborative Filtering Using Keras

Goal: To be able to predict ratings for movies a user has not yet watched. The movies with the highest predicted ratings can then be recommended to the user. Our task is to predict the rating for a user/movie pair, with the idea that if we had a model that's good at this task then we could predict how a user would rate movies they haven't seen yet and recommend movies with the highest predicted rating

1. With the help of the Embedding matrix, user ID is mapped to a "user vector".
2. With the help of the Embedding matrix, movie ID is mapped to a "movie vector".

3. To predict rating, dot product is computed between user vector and movie vector.
4. Trained the embeddings via gradient descent using all known user-movie pairs.

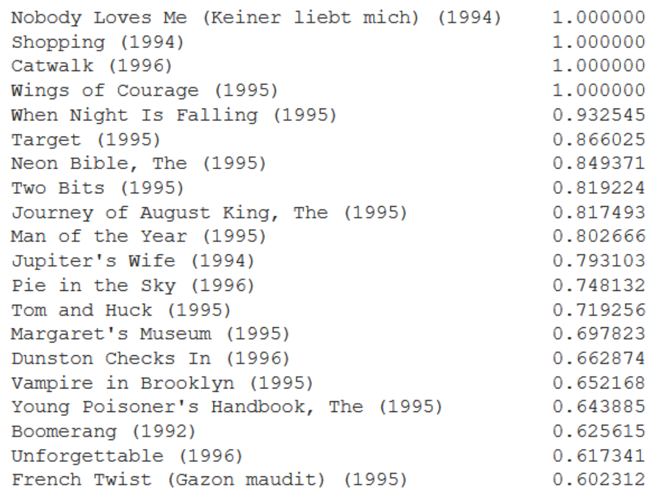
The model computes a match score between user and movie embeddings via a dot product and adds a per movie and per user bias. The match score is scaled to 0 and 1. Both users and movies are embedded in 50 dimensional vectors. Results can be seen in the below image.



```
User: 325
====================================

Movies with high ratings from user:-

Platoon (1986) : Drama|War
Ran (1985) : Drama|War
Psycho (1960) : Crime|Horror
Last Detail, The (1973) : Comedy|Drama
Rashomon (Rashômon) (1950) : Crime|Drama|Mystery
====================================

Recommendations: Top 10 movies :-

Fargo (1996) : Comedy|Crime|Drama|Thriller
One Flew Over the Cuckoo's Nest (1975) : Drama
Shining, The (1980) : Horror
Stand by Me (1986) : Adventure|Drama
Evil Dead II (Dead by Dawn) (1987) : Action|Comedy|Fantasy|Horror
Groundhog Day (1993) : Comedy|Fantasy|Romance
Cool Hand Luke (1967) : Drama
Young Frankenstein (1974) : Comedy|Fantasy
This Is Spinal Tap (1984) : Comedy
Indiana Jones and the Last Crusade (1989) : Action|Adventure
```

4. Item- Item Based Collaborative Filtering Using Knn And Pivot Table

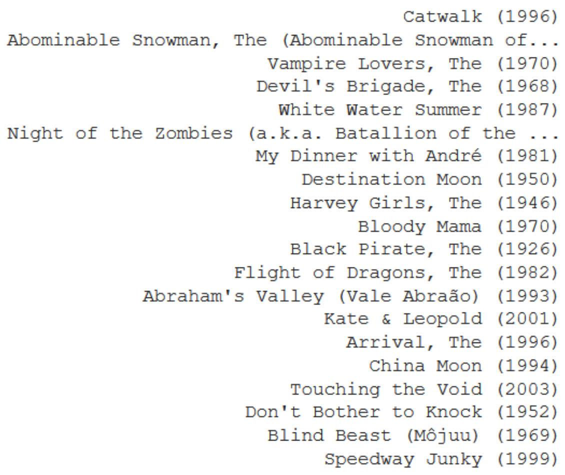
Item-item collaborative filtering is useful to figure out which movie is similar to which other movie. Given a movie, this program will spit out n movies that someone who watched the input movie would watch. From the movielens data, we merge the dataset containing movies and the dataset containing user ratings to label each rating with the movie name and reduce the number of rows to 1000000. The pivot table uses “userId” as the index, “title” as the columns and “rating” is the value. The predictor function takes in a movie title and then calls the built in “corrwith” function for pandas pivot tables. The function then spits back n movies with the highest correlation to the given movie. To demonstrate, we used a sample movie, Catwalk, a documentary and n=20 recommendations. The results are as follows:



```
Nobody Loves Me (Keiner liebt mich) (1994)    1.000000
Shopping (1994)                               1.000000
Catwalk (1996)                                1.000000
Wings of Courage (1995)                       1.000000
When Night Is Falling (1995)                  0.932545
Target (1995)                                 0.866025
Neon Bible, The (1995)                        0.849371
Two Bits (1995)                               0.819224
Journey of August King, The (1995)            0.817493
Man of the Year (1995)                        0.802666
Jupiter's Wife (1994)                         0.793103
Pie in the Sky (1996)                         0.748132
Tom and Huck (1995)                           0.719256
Margaret's Museum (1995)                      0.697823
Dunston Checks In (1996)                      0.662874
Vampire in Brooklyn (1995)                    0.652168
Young Poisoner's Handbook, The (1995)         0.643885
Boomerang (1992)                              0.625615
Unforgettable (1996)                          0.617341
French Twist (Gazon maudit) (1995)            0.602312
```

Since Catwalk is a documentary, a lot of the movies in the top 20, like Shopping (1994), are based on real life. However, some of these movies don't seem to have anything similar to Catwalk which is why we decided to increase the accuracy using KNN.

When using KNN, we decided to use ratings from users that have rated at least 80 movies to make sure every rating holds value and filtered the movie dataset using this same metric. The ratings again need to be reduced to 1000000 and the two data frames are merged and the same pivot table is created. This pivot table is turned into a sparse matrix. The function creates a classifier using 30 neighbors and uses this sparse matrix to fit the data and create a KNN model. Similar to the pivot table method, it takes in movie name and n and takes the n nearest neighbors and returns them:



```
                                   Catwalk (1996)
Abominable Snowman, The (Abominable Snowman of...
                       Vampire Lovers, The (1970)
                      Devil's Brigade, The (1968)
                        White Water Summer (1987)
Night of the Zombies (a.k.a. Batallion of the ...
                      My Dinner with André (1981)
                          Destination Moon (1950)
                         Harvey Girls, The (1946)
                               Bloody Mama (1970)
                         Black Pirate, The (1926)
                    Flight of Dragons, The (1982)
           Abraham's Valley (Vale Abraão) (1993)
                            Kate & Leopold (2001)
                              Arrival, The (1996)
                                China Moon (1994)
                         Touching the Void (2003)
                     Don't Bother to Knock (1952)
                       Blind Beast (Môjuu) (1969)
                            Speedway Junky (1999)
```

Looking over the returned movies, almost all of them were knowledge based or based on real life and the results were much more accurate than the pivot table method.

References:

1. *https://keras.io/getting_started/intro_to_keras_for_engineers/*
2. *https://towardsdatascience.com/supervised-machine-learning-model-validation-a-step-by-step-approach-771109ae025*
3. *https://towardsdatascience.com/machine-learning-model-regularization-in-practice-an-example-with-keras-and-tensorflow-2-0-52a96746123e*
4. *https://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/*
5. *https://towardsdatascience.com/neural-collaborative-filtering-96cef1009401*