DS-GA-1007 Project Report Movie Recommender System

Jin Han
(jh5695)

Xin Guan
(xg702)

***Abstract*—Our team aims to construct a user-interface to recommend movies based on genres and descriptions of movies. In our project, users can input a movie name or a genre to find movies that are similar to the ones they like or find high rating movies with a specific genre with our systems on their terminal or Jupyter Notebook. We download the raw data from TMDb [1]—-These files contain metadata for about 45,000 movies released on or before July 2017 including cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDb vote counts and vote averages. And we believe our rich data can provide a satisfied recommendation to users.**

I. Introduction

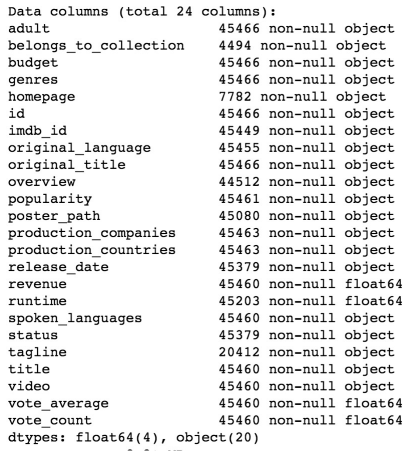
Recommender Systems have become an important part of our lives recent years. It can provide users with new items such as products, books, music, and movies they may like. For example, Amazon uses ”item-to-item collaborative filtering” to monitor their users' interactions: When a user has bought a MacBook online, then they may recommend a USB Adapter or a MacBook case to the user.

It may be quite a common question for a film lover like us that:”What movies should I watch tonight?” It could be answered by movie recommender system, which is designed to predict the preferences of a user, and make suggestions based on what they like. In our project, we developed two kinds of simple movie recommender systems with movie metadata from TMDB. The first one is based on movies genres and it will recommend top 20 highest rating movies related one specified genre. The second recommender system is content-based. Users need to input some keywords of a movie so that the system to calculate the most similar 10 movies to the one they like. As more and more movie fans there are, we believe that our recommendation system will be useful for every movie lover.

II. Methodology

A. Data Cleaning

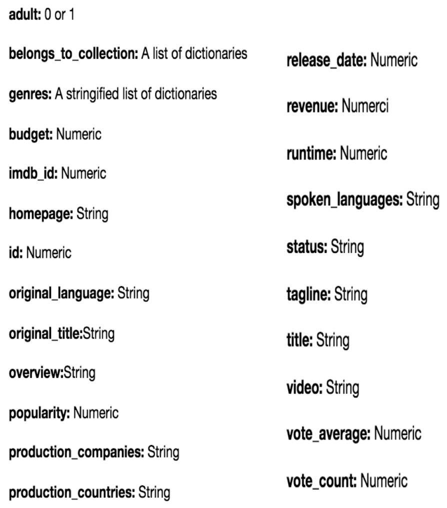
Our whole database has 45466 rows and 24 columns, which means we have 45466 movies and 24 features about each movie. Figure 1 shows us the feature name and the null value distribution using **df.info()**. We can see in the columns 'homepage' and 'tagline', the number of null values are highest. Then we use **pandas** and **numpy** to deal with the NaN data of columns that have only a few null values and select useful features for further work.



```
Data columns (total 24 columns):
adult                    45466 non-null object
belongs_to_collection    4494 non-null object
budget                   45466 non-null object
genres                   45466 non-null object
homepage                 7782 non-null object
id                       45466 non-null object
imdb_id                  45449 non-null object
original_language        45455 non-null object
original_title           45466 non-null object
overview                 44512 non-null object
popularity               45461 non-null object
poster_path              45080 non-null object
production_companies     45463 non-null object
production_countries     45463 non-null object
release_date             45379 non-null object
revenue                  45460 non-null float64
runtime                  45203 non-null float64
spoken_languages         45460 non-null object
status                   45379 non-null object
tagline                  20412 non-null object
title                    45460 non-null object
video                    45460 non-null object
vote_average             45460 non-null float64
vote_count               45460 non-null float64
dtypes: float64(4), object(20)
```

Fig. 1. Dataset NaN Values Information

B. Feature Engineering



adult: 0 or 1

belongs_to_collection: A list of dictionaries

genres: A stringified list of dictionaries

budget: Numeric

imdb_id: Numeric

homepage: String

id: Numeric

original_language: String

original_title: String

overview: String

popularity: Numeric

production_companies: String

production_countries: String

release_date: Numeric

revenue: Numerci

runtime: Numeric

spoken_languages: String

status: String

tagline: String

title: String

video: String

vote_average: Numeric

vote_count: Numeric

Fig. 2. Feature Description

At first, we need to understand what features in our data we have and what types they are so that we can select useful features to filter movies and deal with the data with correct methods. Figure 2 reveals a name and type lists of all features we have.

1. In our metadata, one of the useful features 'title' is a stringified list of dictionaries. We use the package **ast.literal_eval** to transfer it into a list of dictionaries type which could be accessed easily.

2. We generate a new feature named 'Weighted_rating' to evaluate all movies in our list in a unified standard. The formula we use to calculate the Top Rated 250 Movies from the IMDb site is as follows:

Weighted Rating: $WR = \frac{v}{v+m} * R + \frac{m}{v+m} * C$

where :

R = average rating of the movie (mean) = (Rating)

v = number of votes for the movie = (vote_counts)

m = minimum votes required to be listed in the Top 250 (currently 25000)

C = the mean vote across the whole report (currently 7.0)

In our case, we redefine the parameter m by calculating the 95th percentile of 'vote_counts' data.

3. Since we plan to construct content-based recommender, we need to use columns 'overview' and 'tagline' to generate a new column called 'description' by using **pandas**.

C. Recommender System Construction

Method 1. *Genres and Rating Based Movie Recommender* This is simple recommender which could be constructed by sorting all the data by column 'weighter_rating' and then filtered by the specific genre from the sorted data frame.

In this part, we employ basic packages such as **pandas** and **numpy**. We also do some data visulazation with **matplotlib** and **seaborn** here. The following figures give us an overview of top 10 movies with specific genres.

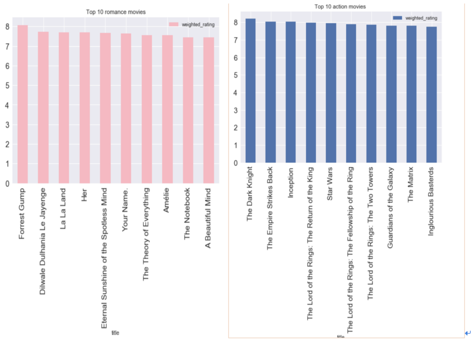


Fig. 3. top 10 movies of different genres

Method 2. *Description and Rating Based Movie Recommender*

Clearly, our first recommender suffers some severe limitations: It just gives same recommendation regardless of users taste. So we construct a content-based recommendation system based on movies overview and taglines. Due to the limitation of computer resource, we select a subset from our database. The subset we use here contains 9099 movies. In this system, users can input some keywords of a movie they like, then they would get a recommendation that is similar to that movie. In order to define 'similarity', we introduced Cosine Similarity [2]:

$$sim(A_i, A_k) = \cos(A_i, A_k) = \frac{A_i A_k}{(||A_i||)_2(||A_k||)_2} \tag{1}$$

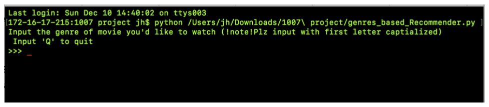
In our case, we employ **TF-IDF Vectorizer** to analyze the text of movies descriptions and calculating the Cosine Similarity Score to give us similarity between movies using **sklearn.metrics.pairwise.cosine_similairty**.

Also, we use Regular Expression to complete fuzzy matching between users input and our dataset employing **re** package in python.

III. Results

A. Genres and Rating Based Movie Recommender

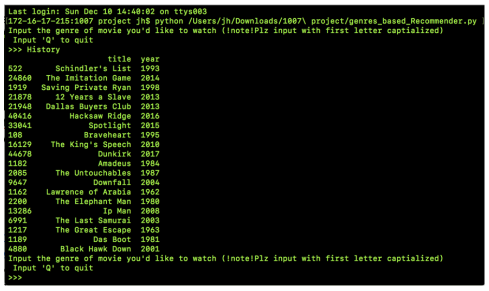
To make the system simple and user-friendly, the system is packaged in one folder. When users download our 'Movie recommendation.zip' folders and run 'genres_based_Recommender.py' in their terminal, they would be asked to input a specific genre to get a recommendation list (The genres information is provided by the file 'The Genres of Movies You Can Search.pdf'). Figure 4 shows the User's Input Interface in the terminal.



```
Last login: Sun Dec 10 14:40:02 on ttys003
[172-16-17-215:1007 project jh$ python /Users/jh/Downloads/1007\ project/genres_based_Recommender.py ]
Input the genre of movie you'd like to watch (!note!Plz input with first letter captialized)
 Input 'Q' to quit
>>> _
```

Fig. 4. User Input Interface

Users can get a list of movies of a specific genre after typing into a genre they are interested in. For example, if one wants to get some History genre movies, he could get a list of Top 20 Rated movies related to the genre he inputs in the terminal. Figure 5 reveals the list of top 20 history genre movies.



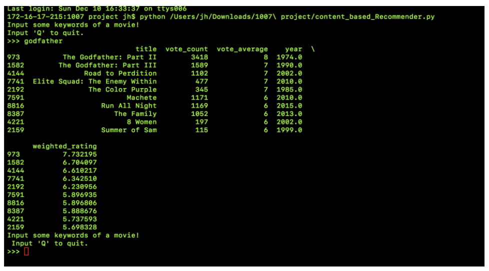
```
Last login: Sun Dec 10 14:40:02 on ttys003
172-16-17-215:1007 project jh$ python /Users/jh/Downloads/1007\ project/genres_based_Recommender.py
Input the genre of movie you'd like to watch (!note!Plz input with first letter captialized)
 Input 'Q' to quit
>>> History
                    title  year
522      Schindler's List  1993
24860  The Imitation Game  2014
1919  Saving Private Ryan  1998
21878    12 Years a Slave  2013
21948  Dallas Buyers Club  2013
40416       Hacksaw Ridge  2016
33041           Spotlight  2015
108            Braveheart  1995
16129   The King's Speech  2010
44678             Dunkirk  2017
1182              Amadeus  1984
2085     The Untouchables  1987
9647             Downfall  2004
1162   Lawrence of Arabia  1962
2200     The Elephant Man  1980
13286              Ip Man  2008
6991      The Last Samurai  2003
1217      The Great Escape  1963
1189             Das Boot  1981
4880       Black Hawk Down  2001
Input the genre of movie you'd like to watch (!note!Plz input with first letter captialized)
 Input 'Q' to quit
>>>
```

Fig. 5. Top 20 History Genre Movies

B. Description and Rating Based Movie Recommender

When users run the file named ’content_based_ Recommender.py’ in their terminal, they will be asked to input the keywords of a movie they like and get a list of top 10 high related movies which contains the necessary information such as the movies title, released years, and vote information.

Take the movie ’The Godfather’ for example, once the user input ’godfather’ in their terminal, the list of movies whose names are most related to godfather would be shown in the interface. The result of it is shown in Figure 6.



```
Last login: Sun Dec 10 16:33:37 on ttys006
172-16-17-215:1007 project jh$ python /Users/jh/Downloads/1007\ project/content_based_Recommender.py
Input some keywords of a movie!
Input 'Q' to quit.
>>> godfather
                           title  vote_count  vote_average    year  \
973         The Godfather: Part II        3418             8  1974.0
1582       The Godfather: Part III        1589             7  1990.0
4144             Road to Perdition        1102             7  2002.0
7741  Elite Squad: The Enemy Within         477             7  2010.0
2192               The Color Purple         345             7  1985.0
7591                        Machete        1171             6  2010.0
8816                   Run All Night        1169             6  2015.0
8387                      The Family        1052             6  2013.0
4221                        8 Women         197             6  2002.0
2159                  Summer of Sam         115             6  1999.0

      weighted_rating
973          7.732195
1582         6.704097
4144         6.610217
7741         6.342510
2192         6.230956
7591         5.896935
8816         5.896806
8387         5.888676
4221         5.737593
2159         5.698328
Input some keywords of a movie!
 Input 'Q' to quit.
>>> 
```

Fig. 6. Top 10 Movies Related to Godfather

IV. Discussion

Even though our system is able to identify a specific film and subsequently recommend other related films as its top recommendations, there are still some constraints of our system. The system doesn’t take into considerations very important features such as cast and crew, which would contribute a lot to the rating and the popularity of a movie. As we could see from the example of ’Godfather’, someone who liked ’The Godfather’ probably likes it more because of Marlon Brando, who is a really famous actor in the world, or the prototype of the movie.

Moreover, there is no personal adaptive function in our system. That is, ever time when users type in one specific word, the list will be all the same. However, the user may not like some movies in the list. This weakness may limit the use of our system.

In the future, we could mine more information related feature. We could also develop our own interactive movie system online to attract more users and enhance the comfort of the users using it.

V. Conclusion

In this report, we first execute necessary data cleaning procedures and extract related features of movies. We then construct two different recommendation systems based on different ideas and algorithms.

The first system is based on genres and rating, using overall TMDb Vote Count and Vote Averages to build Top Movies list. However, since it doesn’t take users’ preferences into consideration, we developed another movie recommendation system based on the description and rating which took movie overview and taglines as input to come up with predictions.

It was a pleasure for us to work on such an interesting topic which could be useful for many people. We hope that the project could be implemented by more people in the future.

VI. Future Work

There are several ways to improve our systems:

1. We can construct a more informal recommender system not only based on genres and overview but also based on directors, movie stars, plot, original language and released years. In this way, our system can provide much more specific and personal movies with users and fit user’s preference more accurately.

2. Until now we are only capable of suggesting movies which are similar to a certain movie. We can use the Collaborative Filtering [3] technique to catch users’ tastes and biases. The first type of collaborative filtering is that try to find like users and make recommendations based on ratings/scores of like users. Another way is to find similar items and recommend items similar to an item a user has shown interest in.

3. Each system has its own advantages and disadvantages. We can incorporate other algorithms and combine content-based methods and collaborative filtering to one recommender system to lead to a better suggestion for our users.

References

[1] https://www.themoviedb.org/documentation/api

[2] Brian dAlessandroNYU DS-GA 1001 Data Science Course Slides https://newclasses.nyu.edu/access/content/group/f696fbe6-3f31-4217-a401-b897106e1985/Lecture16.pdf

[3] Schafer J B, Frankowski D, Herlocker J, et al. *Collaborative filtering recommender systems*[M]//The adaptive web. Springer, Berlin, Heidelberg, 2007: 291-32