Movie Recommender Systems

GUO Fusheng(19413238)

SU Zhanyi(19430051)

LEI Jinghao(19439954)

1. **Abstract**

The application of recommendation system is very extensive and becomes more and more popular today. A good recommendation system can not only establish a close relationship between users and products, but also allow users to rely on recommendations. The recommendation system exists to solve the problem of information overload, which is caused by the existence of a large amount of information on the network reducing the efficiency of use of information. Therefore, the recommendation system allows users to obtain the part of information that is really useful for themselves from a large amount of information.

1. **Motivation**

The recommendation system is now widely used in many fields, and one of the most typical application scenarios is the recommendation system of movie portals. This project is to explore the application of different algorithms in the movie recommendation system and discuss their respective advantages and limitations. In this project, we focus on applying three algorithms, namely content-based recommendation system, item-based collaborative filtering, and user-based collaborative filtering. The steps of the entire project are probably to collect the required data from IMBD, and then apply the data to the three recommendation algorithms, and finally recommend the top ten movie list.

1. **Details of all the steps**
   1. **Data acquisition**

Our data is mainly obtained from IMBD Most Popular Movies (IMBD, 2020) web page. We write a data acquisition python file to collect original data from the website. The logic of data collection is to select 4 most useful reviews from the top 100 most popular movies and get 5 reviews of other movies based on each of authors of the 4 reviews. Therefore, we can roughly get about 100 \* 4 \* 5 = 2000 data from IMBD. This data collection step is mainly carried out with the BeautifulSoup module. According to the tag on the positioning web page, the data useful for the recommendation algorithm like user id, movie name, movie id and movie description are obtained.

Since our data collection is carried out through web crawling, in the future to avoid excessive access to IMBD and lead to prohibited access, our data collection file adds functions to slow down access speed and use random IP to access. Specifically, in order to slow down access speed, we import a sleep function after each data collection and crawl a data for every 3 seconds. For example, it could spend over 6000 seconds for getting 2000 data. In terms of random IP function, we create a function that before requesting a URL, it will run the function and randomly select an IP as a proxy to access the URL.

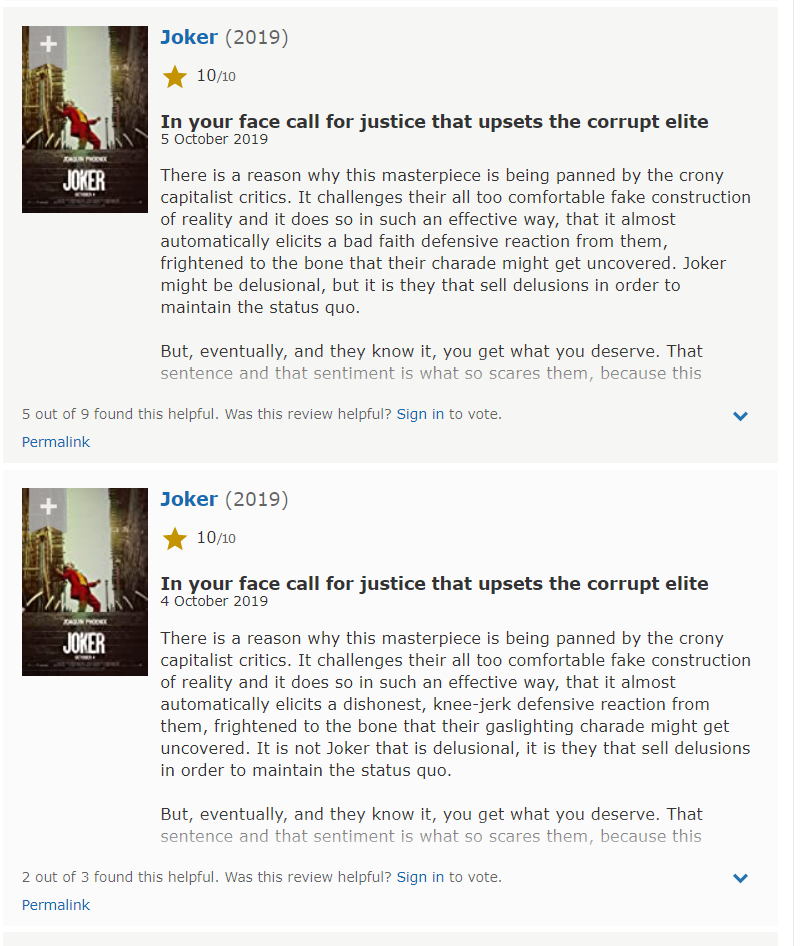
What’s more, we also download a movie rating package for item-based and user-based from the website (jinhuakst, 2018). This data package was acquired by others from the Douban movie. In total, the Douban data package has collected 57424 movies, 28718 users, and 2828500 rating records. This large data package will be a reference for our own data set. The Douban movie data package is at “./ez\_douban/movies.csv” and Douban ratings data package is at the address “./ez\_douban/ratings.csv”. The range of ratings in IMDB data is [1, 10] and the range of ratings in Douban data is [1, 5].

* 1. **Data pre-processing for original data**

After running the data acquisition file, we finally get 2093 original data. However, there are many duplicate values(1352) and missing(27) values in the original data. Since there are too many duplicate values, we visit one of the user pages based on user id.



This is a review capture of user 89494061:

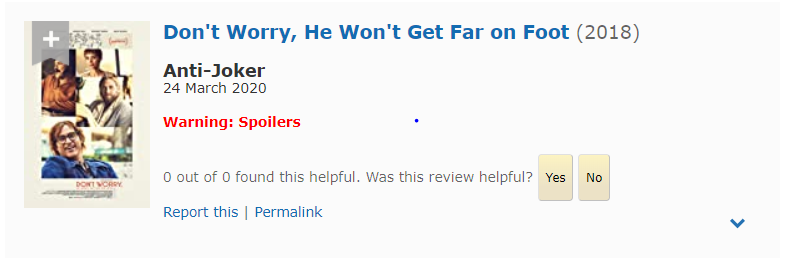


We can find an interesting phenomenon that many users have comment a movie for many times. Imagine that if the recommendation system does not eliminate duplicate values, movies with a large number of repetitive reviews are easier to recommend to users. Therefore, we believe that it is necessary to delete all duplicate values ​​in our data, which is conducive to improving the accuracy of our recommendation system.

For the missing values, we select a user with missing user rate value and view his comment page.



This is a capture based on the above data:



It turned out that this user chose to hide his comments so as not to spoil other viewers, so we could not get this user rate, resulting in the existence of missing values. Since our original data has only 27 missing values, even deleting it has little effect on the overall data, so we chose to delete the missing value data.

After deleting duplicate values and missing values, we get our processed data with 705 rating records, 555 movies and 86 users. Here is the rating distribution of processed data, we can see that most of rating score are over 7 out of 10 point, which means that most of movies in our data are popular movies.

手机屏幕截图

描述已自动生成

* 1. **Data pre-processing for processed data**

The original summary plot grabbed from the webpage are noisy because there are some HTML tags inside the contents. In order to obtain a better summary description quality, the data cleaning process is necessary.

As we mentioned before, we acquired some user’s rating data to movies. This part of the data is used for Item-based RS and User-based RS. To generate a pivot table like **Figure**\_**1**, it is necessary to remove all duplicated data by excel.

**图片包含 游戏机, 电子, 键盘, 电脑

描述已自动生成**

**Figure\_1 pivot table**

Somehow, some movies on IMDb have different space gaps in their movie’s name like “Witcher:” and “Witcher :”. It costs us a lot of time to find the messy data and recleans the whole rating data again. To be easily calculated by the program, we write a python file named dataProcessor.py to transform the original IMDb user and movie ID to normal user and movie ID that starts from 0. For an instant, we transform IMDb user-id “59014” to user id “0”. So, the user id begins at 0 and ends at 85. Our movie data package is at “./data/begin/movies7.csv” and ratings data package is at the address “./data/begin/ratings7.csv”.

We also find that the pivot table has the sparsity problem since most of the users we acquired from websites have not watched most of the movies in our data. So, we simply set default value 0 to handle the sparsity problems because of several reasons. First, it is convenient for programming and calculation. Second, we want to set penalties for those movies that are too few people to watch it because if a movie is just watched by a few people, it indirectly represents its word-of-mouth and reputation is low too. Then, this movie should not be recommended to other users. Third, the RS that we designed recommends movies according to the ranking of predicted ratings. The specific predicted rating is not very important.

Besides, we provide another data cleansing method in our program and we call it the filter-minority-data method. That is to filter movies with less than a threshold view and filter users with less than a threshold view. For example, input a threshold 5, if a movie has only been seen by 4 users, this movie will be removed out of the calculation of RS automatically by the program. If a user has only watched 4 movies, then it will be removed too. The specific threshold will be input at the beginning.

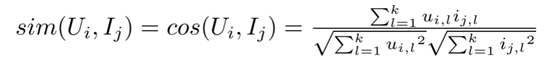
* 1. **RAKE\_nltk**

The natural language toolkit is a set of python-based natural language processing toolset for extracting the keywords form text. The RAKE stands for Rapid Automatic Keyword Extraction algorithm. RAKE can manage the original text to keyword by analyzing the occurrence and pairing frequency of terms in a document. In the project, use RAKE to process all summary plots, thus extract the keywords of the corresponding movie. In most situations, the keywords can contain the content of the movie. Rather than the original content, the key words are more suitable for content-based recommend system. The order of extracting keywords form RAKE is from the utmost essential to least important words.

* 1. **Data analysis**
     1. **Content-based RS**

The idea of content-based recommend method is to acquisition the analogous items in descriptions or profiles. Content-based filtering are commonly applied and receive an exceptional effect on general webpages, for example, applying the users’ previous records to recommend the corresponding goods in the online shopping platform. When comparing two movies, the summary plot, actors, directors, genres and the movie rating can be used in the content based recommend system. In the project, only choosing in using the summary plot as the content-based factor, if the movie types, actors and directors are used, perhaps the result could be more accurate to what the users’ ideal outcome. Next, exploiting the cosine similarity method to compute the most akin movies.

To change the keyword to digital label vectors, the project implemented the function in CountVectorizer and Tfidf\_transformer from sklearn.feature\_extraction (Pedregosa, Varoquaux, & Gramfort, 2020). These two methods both could change the key words into number tokens, but the results are different. CountVectorizer focus on the occurrence counting, but the TfidfVectorizer rating the words by the items’ frequency displayed in the document. Because the data pre-process is fulfilled with RAKE, the text has been cleaned up as keywords and the frequency of the terms should not influence the weight of words. Therefore, elect CountVectorizer for the project. Because the keyword result is ranking, but in the project, we ignore this factor, so it should affect the recommend result.



**Figure\_2 cosine similarity**

Cosine similarity will get the similarity score between two vectors, the result close to 1 indicates that the two vectors are identical, on the country, score 0 represent for the two vectors are different. After calculating the similarity matrix of the movie datasets, a recommend model will be generated. Each row or column represents a movie, and the two most similar movie in the data set can be found based on the highest value in the row. When a query of detail content like “crime gun” passes through the recommend system, it need to add a new row of the query item. Next calculate the similarity matrix again, the newly added row becomes an array of similarity scores. The top 10 highest-scoring movie will be recommended. If query in movie title, just find the corresponding movie name and get the particular rows to display the most related movies

* + 1. **Item-based & User-based RS**

**Figure**手机屏幕截图

描述已自动生成**\_3 Similarity formula**

For item-based and user-based RS, we only use some other python packages to help us handle the data and complete the algorithms such as pandas, NumPy, the cosine method in SciPy. We write two different similarity calculation method for both item-based and user-based: cosine similarity and Pearson correlation coefficient. The formula is shown in **Figure\_3**. These two similarity methods have two different similarity output. The cosine similarity will output value in the range [0,1]. The Pearson correlation coefficient will centralize and output value in that range [-1, 1].

手机屏幕截图

描述已自动生成

**Figure\_4 User-based formula**

手机屏幕截图

描述已自动生成

**Figure\_5 Item-based formula**

The neighbor size also can be changed in our program. Therefore, the predicted rating value using the Pearson correlation coefficient will be more extreme than the cosine similarity in theory. **Figure\_4** shows the principle formula of user-based RS and **Figure\_5** represents the formula of item-based RS. The basic idea of our item-based and user-based RS is that first choose a user for whom you want to recommend movies in our data. Input the user id in the program. Then, the system will find what movies this user has not watched. And the system will calculate the predicted ratings based on the unwatched movies list. Finally, sort the predicted ratings in descending sequence. And it prints out the top 10 movies for the recommendation. Also, we record the calculation time on a different dataset.

* 1. **Visualization of analysis result**
     1. **Content-based RS result**

****

**Figure\_6 virtual user 86**

As the content-based result of the movie ‘Joker’ showed in **Figure\_6**, it matches only one recommend movie in the IMDb webpage. The classification accuracy of precision and recall are both with the low scores, the main reason is result may be due to insufficient experimental data, movie recommended on IMDb does not exist in the experimental data. Secondly, the recommendation result in the IMDb are more related to crime-type movies, science fiction, high-rating and popular in the recent days. These kinds of factors are not considered in the content-based recommend system. Next, apply the count vectorizer but the importance ranking is ignored, and the result used in the RAKE and all the vector become ‘1’ before calculated in the cosine similarity.

* + 1. **Item-based & User-based RS**

**手机屏幕截图

描述已自动生成**

**Figure\_7 virtual user 86**

To test the RS, in **Figure\_7** we set a virtual user whose id is 86, and user 86 loves to watch popular movies such as Ant-man, Avatar, Avengers: Endgame, and so on, and try to recommend movies for him. The predefined experiment parameters are that 20 neighbor size, cosine similarity method, prefilter the data with a threshold of 2, and movie data file named movies7.csv at the address “./data/begin/movies7.csv” and ratings data file named ratings7.csv at the address “./data/begin/ratings7.csv”.

手机屏幕截图

描述已自动生成

**Figure\_8 result without threshold**

手机屏幕截图

描述已自动生成

**Figure\_9 Item-based result with threshold**

**手机屏幕截图

描述已自动生成**

**Figure\_10 User-based result with threshold**

**Figure\_9** and **Figure\_10** are the top 10 recommendation results of item-based RS and user-based RS. **Figure\_8** is the result as the same in **Figure\_9** but without any filter-minority-data method used. And you can see that there are a lot of unpopular movies recommended in **Figure\_8**, which means that the filter-minority-data method is truly useful. After the recommendation, we also go to the internet to search whether the recommended movies are popular or not in terms of the online ratings, word of mouth, the type of movies, and the number of views. The consequences of item-based and user-based RS pretty satisfy the demands of user 86 in general. Almost famous action, drama, sci-fi, fantasy movies are all included except the movie named “The interviews” in the result of item-based RS. The interview is an unpopular short movie in 2019.

图片包含 游戏机, 文字

描述已自动生成

**Figure\_11 Item-based RS on the Douban data**

And **Figure\_11** shows the item-based RS result of running the Douban data package with Pearson similarity, a threshold of 200 filtered, neighbor size of 20, and recommend to the user with id 4. In this case, the data is much larger than our data and the result is much better too with higher accuracy. The data is so large that it is very time-consuming to cost me 2465s to complete the calculation, which is nearly 41 minutes. If run the Douban data set without any data filter method such as filter-minority-data, the predicted estimation of running time will cost more than 2000 hours to complete the calculation in normal computers, which is impossible. It requires big data technologies such as cloud computing, MapReduce, or multi-thread to speed up the time-consuming for corporations.

1. **Discussion of the results obtained and your interpretation**
   1. **Content-based Discussion**

The content-based recommend system has its own advantages. Content-based theory are straightforward, and it can conveniently demonstrate why such movies may be recommended to users. Because like movies always involve the identical contents and keywords. The recommendation system does not rely on the ratings of other users, each movie has its independent description. When a new movie enters the database, it can be recommended immediately according to the summary plot. New movies without rating are difficult to use in collaborative filtering recommendations.

However, there are many synonymy and polysemy words which will have a significant influence on the content-based results. If the most important unique keyword in the message cannot be extracted, other useless word may be appeared in the extremely different movie and be harmful to the recommend result. Furthermore, since the movie description is hardly change over time, so the content-based recommend system will always recommend the changeless movie. But the ideal recommend system want to find the potential interesting movie which will attract the users. A better recommend system should take consideration of new and popular movies.

* 1. **Item-based & User-based Discussion**

Based on the result and the experimental process, we figure out several limitations of our item-based and user-based RS.

Firstly, our data sample is still too small to fully reflect the real-world situation. For example, Avatar is a great popular movie in the world. But it only has one user watched it in our own crawled data, which makes no sense. Only 705 ratings data is still not large enough to reach higher accuracy.

Second, we understand that it is not thoughtful that only use default value 0 to handle the sparsity problem because everyone only watched 8.22 movies on average in our data. However, to some extent, the filter-minority-data data cleansing method is helpful to fix some sparsity problem and filter the minority movies and minority users.

Third, the limitation of the filter-minority-data data cleansing method is that the filtered minority movies cannot be included in the calculation and cannot be recommended to specific users. Similarly, cannot conduct recommendation for the users who have been filtered.

Fourth, the reason why the interview is recommended may be that our data is still not large enough to reflect the real similarity situation between movies. In specific, because the data is small, maybe some popular movies in our data are just similar to the unpopular movie “The interview” since the item-based RS conducts the recommendation according to the similarity of neighbor movie.

Fifth, when we use the Pearson correlation coefficient to conduct the recommendation, sometimes it will calculate a much larger predicted rating than maximum rating 10 in IMDB. The reason is the Pearson correlation coefficient can have negative similarity value. When a movie is very not similar to another movie, the RS formula may be that the numerator and the denominator are both negative values, which finally will become a positive rating. That is also why we tend to use the cosine similarity method instead of Pearson.

Nevertheless, it still has some advantages as well. Even the size of data sample is small, the top 10 recommendation is still recommended accurately through the data cleansing method filter-minority-data. The user-based RS has a better result than item-based RS. The collaborative filtering RS seems have a better recommendation result than the content-based RS. With the more ratings data, the collaborative filtering will be more accurate. This is logical in theory to a certain extent. We also implement many approaches to optimize the recommendation results by ourselves.

1. **Conclusion**

In conclusion, we conducted research on three types of recommendation systems. We built the RS through data acquisition, data preprocessing, data cleansing, data analysis, and result visualization. In general, the content-based RS is very suitable to be a guidance for the new user at the very beginning because it requires less information than the collaborative filtering. With the increase of user ratings, the collaborative filtering method will be a better choice for the recommendation. In real life, the sparsity problem is always serious because the great majority of people only watch a few movies in the real world. The average movies watched number 8.22 can indirectly reflect the real situation of people watching movies in the real world to a certain extent. Compared with the total number of films, the number of films watched by each audience is extremely small in reality, and the movies that users watch are highly concentrated in popular movies. Therefore, these parts of data will have more valuable and the corporations can consider spending more resources on this.

1. **References**

IMBD. (2020). Retrieved from https://www.imdb.com/chart/moviemeter/?ref\_=nv\_mv\_mpm

jinhuakst. (2018, 3 26). Retrieved from Github.com: https://github.com/SophonPlus/ChineseNlpCorpus/blob/master/datasets/ez\_douban/intro.ipynb

Pedregosa, F., Varoquaux, G., & Gramfort, A. (2020). *Feature extraction — scikit-learn 0.22.2 documentation*. Retrieved from scikit-learn: https://scikit-learn.org/stable/modules/feature\_extraction.html

Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010, 3 4). Automatic Keyword Extraction from Individual Documents.