COMP7630 Group Project Report

Movie Recommendation base on IMBD

19403844 Deng Guoquan

Abstract

In this project, we explore what we lean about the recommendation system on the lectures. As beginner of the designer of recommendation system, we will apply the content base recommend system, collaborative recommendation system and explore some new idea. All the data in this project about movies is crawled from the official website IMBD.com through code.

1, Introduction

Nowadays, a feasible and excellent recommendation system often means clients, money, technical asset so on, which means recommendation system contain great potential business value. So how to create a high performance recommendation system become a big topic on us. In this project, we focus on how to reproduce a good enough recommendation system from the beginning and explore each detail or difficulties of a recommendation system project.

2, relate work

The whole project contains three tasks, each for one student.

1. The first task focus on user rating prediction. The main part of this task is, according to the previous ratings of a specific user, predicting the rating value given by this user when he watch a new movie. It is a task similar to CTR task and usually used in a lot of business condition.
2. The second task is related to NLP method, it pay more attention to how to make the recommendation through description and summary of movie. It is a challenge task because feature engineering is difficult and critical.

(3)The third task is relate to content-base recommendation system base on genres and key words of movies. Using search algorithm to recommend item to client is also an useful and general method which widely used in business work.

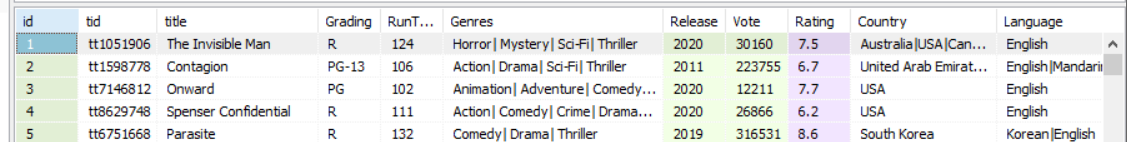
3,Rating Prediction Task

3.1, Data collection

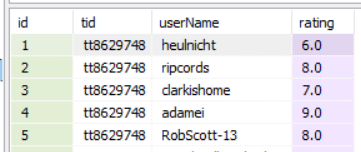
The first thing I need to consider is data of collection. IMBD website is not complex HTML structure website, so it is suitable to crawl information from IMDB though it is long request time to get response from IMDB in the mainland of China. I use a framework call Scrapy, Unlike the normal tool of python for example requests package, scrapy can make even 64 request in one second, and it is asynchronous so when it faces the block which may waste a lone time, it can solve other tasks at first. Instead of using package beautifulsoup, I use package lxml which can be used to parse xml structure directly and faster than beautifulsoup. The data crawling from the website will be stored by easy data base sqlite3. Before crawling the information, template data base which used to store the status information can not be ignore. I create a template table to store the request and response status for each url I crawl preventing loss information when some bug or connection break happen during crawling data. If some thing wrong appear, I can continue the task from the break point again. Since IMBD has not IP limit so I do not use proxy. But in order to avoid detection from website, each request should be pack with different user-agent and remove the cookie information.

After nearly one day crawling, I crawl basic information of 11556 movies. Two table haven been created to stored all data, here I will provide a short brief.

1. Movie Table: title, move\_id, Grading, Vote, Rating, Language, Country, Release\_Data, Genres, Production\_company, Director, Writter, Cast, Summary, Keywords. It contain 11556 movies’ basic information.

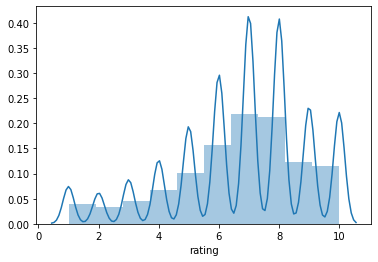


1. UserRating Table: userName, movie\_id, rating. It contain 1258702 rating records.

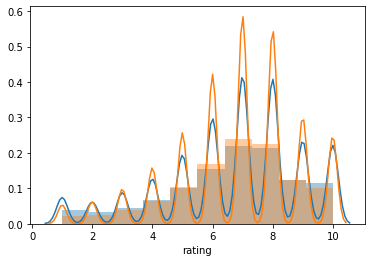


3.2, Data cleaning

Because I focus on rating prediction task, so the first step for me is to clean the rating data. The first step of data cleaning is to remove the row with empty or null. Since the data I can crawl is small compare with the whole data from the website, what is more, there are not any explicit relationship between pages I crawl, so the data set is very sparse. Some movies just have only one rating, some client just rating one movie, all these data is noise and most of them are out of real situation because I can not gain complete data. So I remove all these data. Some clients give all the movies they watch the same rating, I also remove all these data. Now I get the data based on distribution shown below:



Since the computation of my computer is not good enough, I sample higher density data from this data set, so the distribution of mini data set I use is shown below:



Extract suitable data set is very important, you will need long time and larger model to over fit the challenge data set, to some degree, if the data set contain a certain percentage of noise data, even the complex and larger model can not be trained well.

3.3, Estimation method

Estimation consist of two part: 1, using RMSE and MAE as the error function, calculate the RMSE and MAE between real rating and predict rating, RMSE using for training, MAE using for directly compare. 2, Choosing RMSE and MAE between mean rating group by user and real rating as the base line of models. In this data set, the base line is 1.46. The formula to gain the base line shown below:



Reason of user’s mean rating as base line: to some degree, the base line contain some prior information of clients, only the clients are interested in the movie, they will watch it, if they do not like the movie, they will not even watch it or give a rating. So that is why we can see the distribution of data set is right skew and imbalanced.

3.4, Model Exploration

3.4.1 Cosine similarity model:

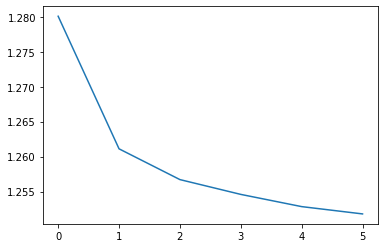


it is a easy model, assume similar movie for same user having the same rating value. Value of cosine distance is -1 to 1, because the movie totally different with the estimated movie will not provide negative value so I compress the cosine distance into 0 to 1. And for the purpose to simple the calculation process, I regular the vector length of embedding into 1.



It sound reasonable, but there are some disadvantage in the model. 1, the predicting value can not excess or less than the history highest point and lowest point. 2, movie with similarity of 1 get the highest score is not reasonable, assume some one love topic like Action, if a movie contain more content about action it should be given a higher rating although the similarity is not 1.

The result I got for this model shown below, y-axis is MAE error.

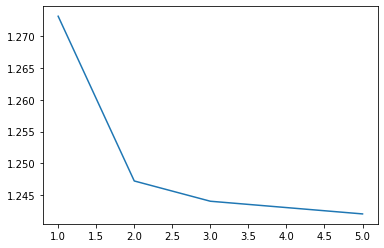


3.4.2 Advance similarity model

For the purpose of resolving the disadvantage of Cosine similarity model, I use a embedding linear regression model. Imaging movie as embedding of size 50, each point of embedding represent percentage of one topic, more percentage of topic client liking, the rating will be higher.



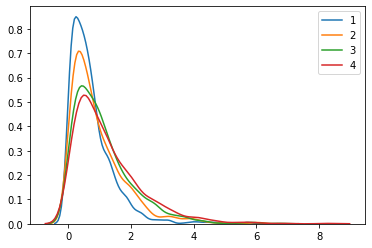
The MAE cost can be seen below:



Unfortunately there is no improvement for this model.

3.4.3 Collaborative Filtering

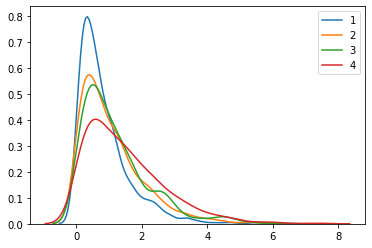
Collaborative Filtering is normal method, I test user-based collaborative Filtering and item-based collaborative Filtering. And for both two algorithm, I test two type of distance function, one kind of distance function A just consider the Intersection set of rated movies from two client, the other kind of distance function B, I consist the union set of movies which has been rated by the compared two customers.



Here is the MAE cost distribution, line 1 represent user-based collaborative Filtering with distance function B, line 2 represent item-based collaborative Filtering with distance function B, line 3 represent user-based collaborative Filtering with distance function A, line 4 represent item-based collaborative Filtering with distance function A. The following expect cost are 0.73, 0.93, 1.10, 1.24.

The cause of the collaborative Filtering contain the mean rating of client, so it work better than it is in the real world. Mean rating of client will provide prior information. In collaborative Filtering, we assume mean of rating represent bias of client though it is nor correct after I try to use rating value minus user’s mean rating as parameters of distance function and it fail.

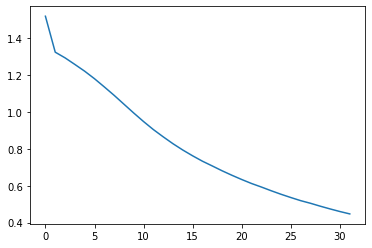
After removing prior information the distribution of MAE will become:



3.4.4 Simple Correlation Model

Simple model can make a good prediction. In this model, I suppose the there is a correlation matrix between each pair of movie, somebody love one movie may love the other specific one. But when the number of movies is larger, it will exponentially increasing, the size of matrix equal to . but we can use a embedding space to represent this matrix because this matrix is symmetry:





3.4.5 Content Base model

Some content and basic information is also very useful. I choose genres as the content of movie, there are 36 genres in this data set. The first step is define the user profile and movie feature haven been show below:

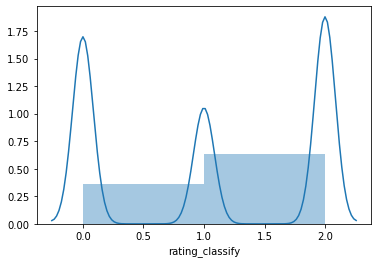


When I have movie feature and user profile, we need to create cross feature.

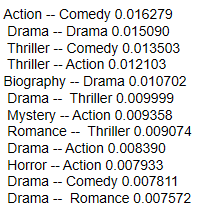


Because the computation process (1) of cross feature is not symmetry, I need to transform the cross feature matrix into a symmetry matrix, so I use formula (2), and then select the upper triangle of this matrix as the feature.

For the purpose of training the model, I adjust the train data set, I classify all the rating which larger or equal 8 to be true positive recommendation as 2, all the rating which less or equal 5 as true negative recommendation labeled 1, the other are 0. And the distribution of data set become:



In order to solve the imbalance problem, I use re-sample method to ensure the positive samples and negative samples are both 50%. for this binary classification problem, two modes have been trained, logistic regression model with accuracy 63.07%. Random forest model: 82.06%. Some interest and importance cross feature shown here:



1. Second Task:Recommendation based on Description of Movie

4.1 Data cleaning

I used four sets of data to train my model. Key columns are listed below. For data cleaning, I focus on summary column. This column is description of movies. I want to use this column to build TF-IDF matrix, so that I remove all the stop words to

improve accuracy. There are difffferent number of descriptions for difffferent movies.

To make it fair, I only leave one description for each movie.

Movie Table: id, movieID, title, Genres

Keyword: id, movieID, keyword

Summary: id, movieID, summary

UserRating: userName, movieID, rating

4.2 Model

The model I used is content base recommend. First, I found the movie with the

highest rating by each user and record the rating. The rating is considered as user

profifile. I only keep rating higher than 6.0, because it can not tell the interest of user

if the rating is too low.

I keep one description for each movie. I collect all the descriptions and build TF

IDF matrix. Then, I used keyword of movies to calculate cosine similarities to fifind

target movies. For example, if I have user who’s highest rating is 8.0. The movie

called X. I will collect all the keyword of X. For each keyword, I will calculate cosine

similarity using the TF-IDF matrix. Then, I will record the movie with highest cosine

similarity. After iterating all the keywords, I get one movie for each keyword. At this

stage, I have some target movies. To make it more accurate, I remove all the

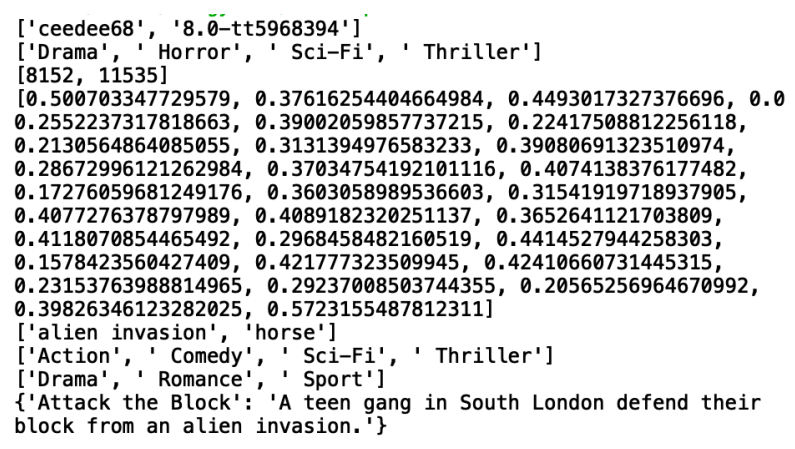
movies with similarities lower than 0.5. If the similarity is too low, it is useless. After

this step, only few targeted movies remains.

To do second fifiltration, I record the genres of X and those targeted movies. I will recommend the movie with most number of same genres. This shows that the

movies I recommend is closer to user’s interest.

4.3 Results Analysis



This is a piece of results. The fifirst line shows that there is a user called ceedee68 and his rated a movie 8.0 which is the highest rating. The second line is the genres of that movie. The flfloat point numbers in the list are cosine similarities. ‘alien invasion’ and ‘horse’ are two key words with cosine similarities higher than 0.5. The next two lines are genres of targeted movies. The last line is the movie I want to recommend to ceedee68. The movies is ‘attack the Black’, and it comes from

the keyword ‘alien invasion’.

4.4 Evaluation

There are some limitations of this model. The fifirst is that some users only rated

movies with low scores and some users did not give rating to any movies. Therefore, I have no ideas about the preference of these users. The second limitation is that it will be inaccurate results if all cosine similarities are low. The third limitation is that I have to calculate similarities for all keywords. There are too much calculating, resulting in low effiffifficiency. Although for the time being I have not thought of a good way to solve these limitations, but when I have more knowledge, I will modify the model better.Recommendation system is complex. I think we can not simply say which model is the best. In the future, I think it is feasible to combine multiple algorithms. We can take advantage of each algorithm and reduce the limits of each algorithm.

5, Third Task:Content Base Recommendation System

In the database every movie has a set of keywords which describe its main content. By comparing the cosine similarities of keywords between different movies we can know what kind of movies should be recommended to the user.

5.1 Retrieve Data

The first step is to retrieve data from the database file. We exported ‘Movie’ table and ‘keywords’ table from the database and save them into pickle files. The ‘Movie’ table contains the title of movies and ‘keywords’ table contains the keyword for each movie, and the same movie in different tables share the same index number.

We only need title, genres and keywords, therefore we export those data from the database into the form of dataframe.

5.2 Data Processing

Then we extract keywords from movies in the profile. In the project, for each test run we generated 5 random numbers as index number of movies to form a profile. From the profile we can extract keywords, split these words and save them in a list. However, using all the keywords as a search query to compare with all the movies in the database is time consuming, therefore we need some methods to improve efficiency. One way is to identify relevant movies from irrelevant movies, the other way is to remove words that are not good at classifying movies.

We use a simple way to distinguish relevant movies from irrelevant movies: for each movie just check if there is a same type of movie in the profile. If there is one, then this movie is relevant. The next step is to identify effective keywords. There are a lot of redundant words in the saved keywords so we need to dump them first. Then we need to clear stopwords.

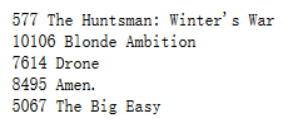
Though stopwords show in contents frequently, they do not help classifying movies, therefore they need to be removed. Even without redundant words and stopwords, the remaining words we get still form a large list. To optimize the words used in search query, we need to measure the performance of words classifying movies. So we use each word to search for contents containing this word and calculate the similarities between the contents. If the similarity values in the results are low, it means that contents containing this word do not have tight connections, therefore this word is not suitable to be used to classify movies. After calculating the results, we use a least misery method to remove words whose maximum similarity of the search results is significant lower and deviant from the mean value and form the rest of words into a query list.

5.3 Calculate Similarities

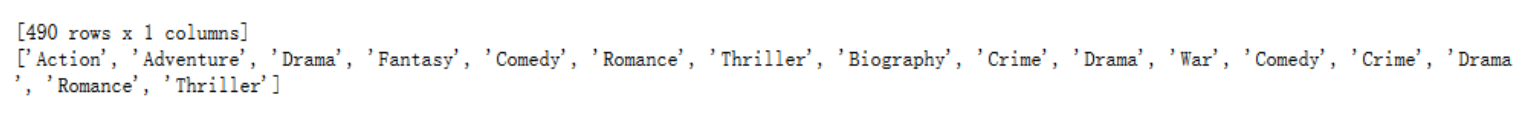
At this step, we have already got a list of query words and a collection of relevant movies. Then we can compute cosine similarities between the list and each movie in the collection, and eventually list the results in descending sequence.

5.4 An example

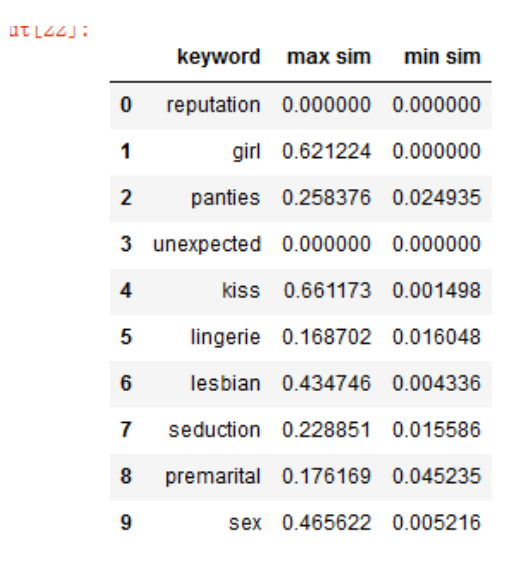
A profile generated using random number 577,10106,7614,8495,5076.



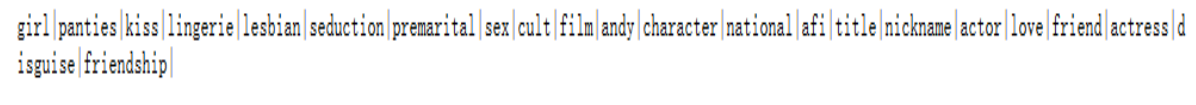
There are 490 keywords in total and the types of movies are shown below. We only search for movies within these types



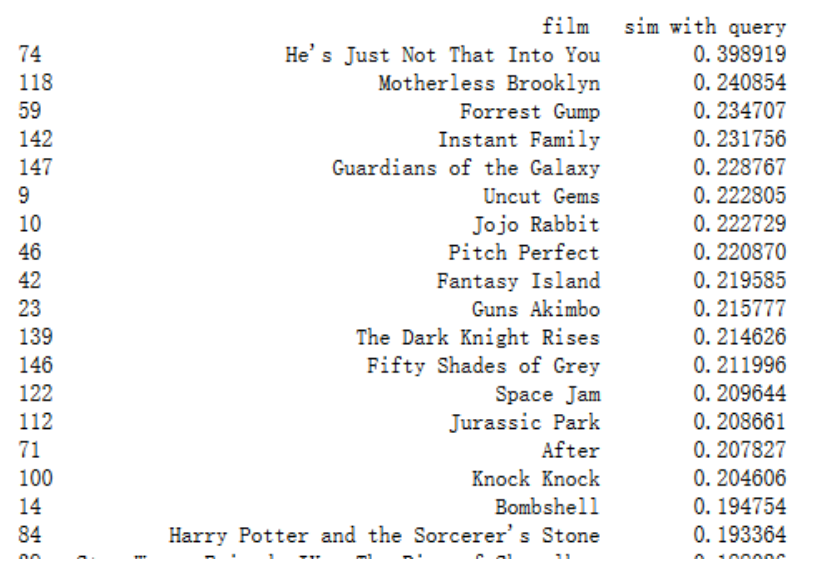
Portion of similarities between contents containing keywords. 0 similarity means this word do not appear in any movie’s keywords except for movies in the profile.



A set of chosen words used for query



List of results:



According to the results, ’He’s Just Not That Into You’ is the best recommendation.

Conclusion

In this project, we explore some classical recommendation methods such as Collaborative Filtering and design some new models on real world movie data set which crawled from IMDB official website. Base on some linear model testing in this report, it seem that the real world data is sparse and scale-free, that is the reason why recommendation system can work. Customers usually pay more attention to small number of items and do not care the others.

The further work

Reference：

1, 《推荐系统实践》，项亮编著， ISBN-978-7-115-28158-6,2012.6

2, 《Algorithms of the Intelligent Web 2nd-Edition》version 9, Douglas G.Mcllwraith, Haralambos Marmanis, Dmitry Babenko, 2016 copyright manning publication

1. 《Building Machine Learning Systems with Python》,Willi Richert, Luis Pedro Coelho, ISBN-978-7-115-35682-6,2014.7
2. 《The BellKor Solution to the Netflflix Grand Prize》，Yehuda Koren，2009.8

5, 《MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS》，Yehuda Koren，Yahoo Research， Robert Bell and Chris Volinsky, AT&T Labs-Research.