Contend-based movie recommendation system

Cai Ying & Chong Meng

Chong Meng - December 6, 2018

Introduction

With the accumulation of the online records, currently abundant datasets are established for all kinds of activities such as shopping and watching movies online. These large dataset give us the possibility to analyze these activities and knowing the important features of those activities can further help improve the services of online merchant.

For example, if we found the personalized preference of a customer form his history of watching movies, we can recommend new movies closer to their appetite. Obviously this kind of recommendation will enhance the amount of movies the customer will watch, and also accurate recommendation will win customers' loyalty since they get a wonderful experience in your services.

In addition to recommend movies to user, we can also aim at a specific movie and find the potential customer who is more likely to watch this movie. This symmetrical recommendation will dramatically decrease size of the targeted customers for some activity such as advertisement for new movies, which will improve the effect of advertisement and reduce its cost.

The main goal of our project is to build such recommendation system based on MovieLens dataset. The version of MovieLens dataset we are working with (265MB) contains about 27,000,000 ratings applied to 58,000 movies by 280,000 users. it also contain information such as each movie's genres which is very important for calculating the similarity between two movies.

Our movie recommender is an item-based recommendation system. We computed the Cosine similarity between movies first. Based on the cosine similarity, we do recommendation incorporating with k nearest neighbors (KNN). Besides that, we also take the rating of users into our consideration when we recommend movies to user.

This report consists of 5 parts. The first part explains the procedure we preprocess the data to make the data more valuable. After that, the method we used in this project is demonstrated. The third part will be the discussion about the results of recommendation. Following that, We conclude our work and some aspects could be improved will be mentioned. The last part we will show how to run our code to get the results we used in this report.

Data preprocessing

The MovieLens dataset contains complete record, but directly use all of these data may not be a good choice. Since some extreme cases in the dataset will not help our recommendation, it will mislead the recommendation and introduce extra meaningless computation cost conversely. In addition to that, recommendation regards to these movies or users is not meaningful because these users are not likely to watch many movies and these movies can merely attract more audiences.

Therefore we eliminate extreme cases for both movies and users. We remove movies watched by less than 25 audience and users who watched less than 25 movies out from the dataset.

Before the data processing, we have 27,753,444 ratings applied to 58,098 movies by 283,228 users.

After refining the data, we have 25,876,299 ratings applied to 16655 movies by just 153524 users. Apparently the result shows that most rating records are given by nearly a half of audiences and only for nearly one forth of all movies. So, our data preprocessing eliminate large amount inactive users and movies with a slightly decreasing of whole rating record.

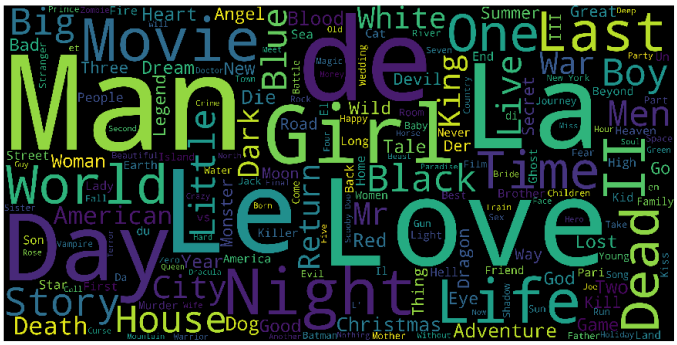

Fig 1. word cloud of movie title

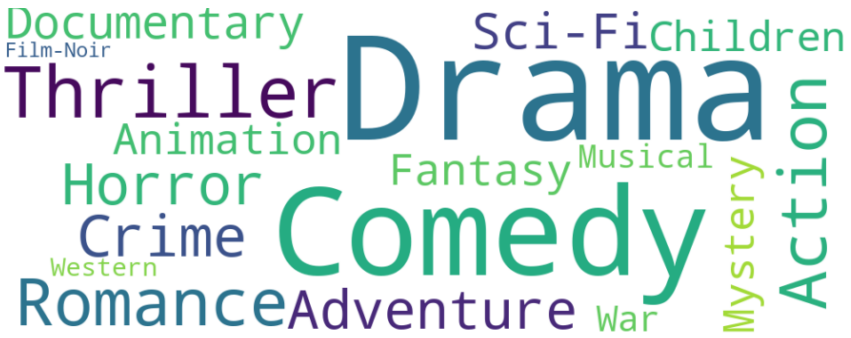

Fig 2. word cloud of movie genres

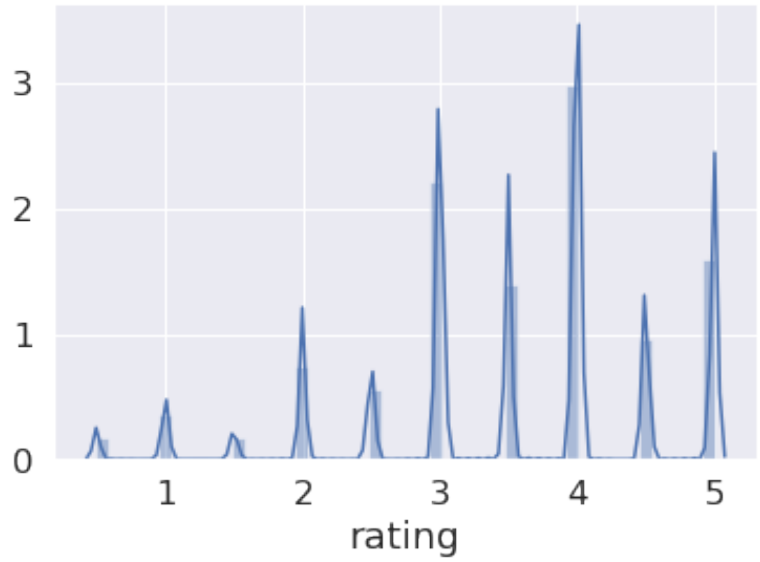

Fig 3. User rating distribution

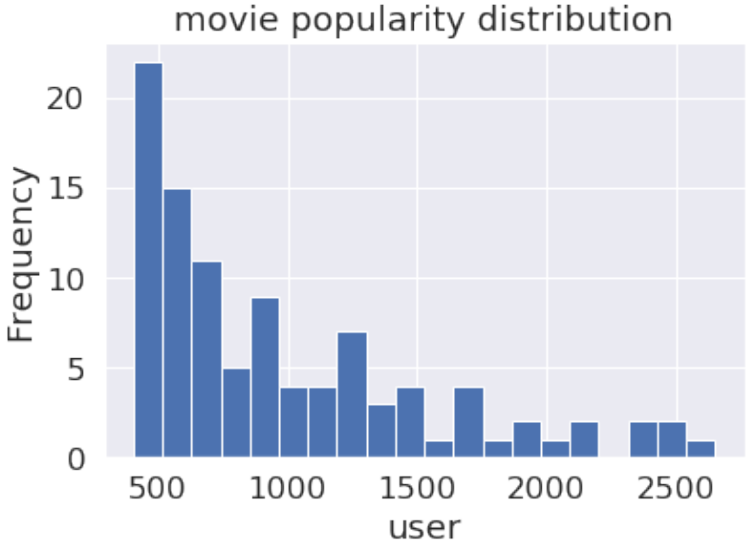

Fig 4. Movies' popularity

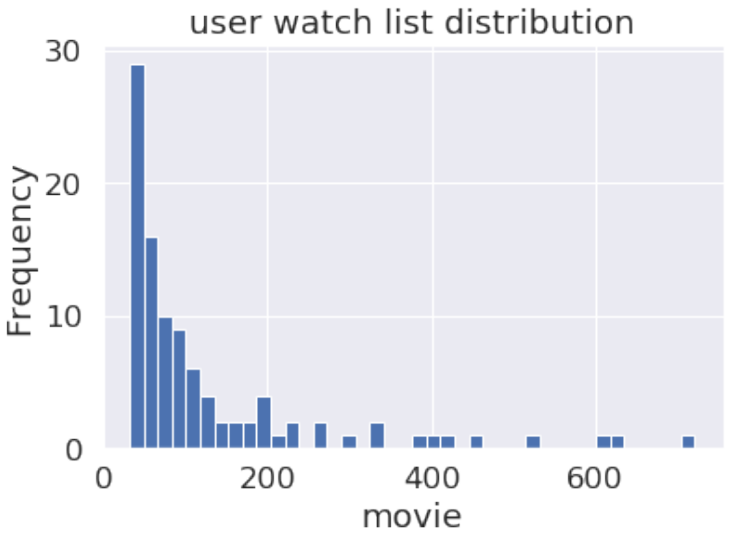

Fig 5. User watch history distribution

We also takes a first glance of the refined data. First we take a word cloud of movies' title, shown in fig 1 with the size of font represent the frequency. The word cloud of movies' genres is shown in fig 2, which tells us the most popular genres of movie are 'Comedy' and 'Drama'. The distribution of ratings is depicted in fig 3, it was normalized by 3.5 which is mean value of all ratings. We can see that most users are very general to give a high rating.

For movies' popularity, we plot the histogram of number of users who watched a special amount of movies. Fig 4 show the results starting from 26 movies, our data tells us 2,660

users have watched 26 movies and 2,500 users have watched 27 movies. Here we just plot the top 100 value of number of users. Fig 5 shows the results of movies watched by a special amount of users. For example ,there are 277 movies watched by 27 users and 200 movies watched by 30 users. Histogram plotting of top 100 value sorted by number of movies are given in fig 5.

To start our prediction, we also split our data to two parts, for test and training respectively. We take 20% from the refined dataset of ratings for test. For one random sample, we got 5,175,260 records applied to 16,653 movies by 153,488 users. The remaining 80%, which contains 20,701,039 ratings applied to 16,655 movies by 153524 users, is for training.

Run my code

Download the dataset:

The dataset we are working with is pretty large, you can download it from the link below
http://files.grouplens.org/datasets/movielens/ml-latest.zip

After download, please unzip it to 'code' folder. So there should be a 'ml-latest' folder now.

About our code:

There are 3 .py files of our code.
DataProcess.py: Read the original data from 'ml-latest' fold and write pruned data to current path.
movie.py: Read the pruned data and save mapping information for late usage.
recom.py: Calculate the recommendation and evaluate the results, results will be stored in .json file.

Run our code

You can run our code by submitting the batch job directly in palmetto cluster:
qsub recom.pbs
It will load the anaconda3/5.1.0 module and activate root python environment.
Then it will execute these three python files one by one.
After the job, you can find our results stored in *.json file.

We plot our results by excel and R code, producing figures in this report