Movie Recommendation system

Group 26 Yan Gu, Xinyang Wang

Oct 23, 2019

1 Project Proposal

1.1 General system description

Movie recommendation systems are movie information filtering system that help recommend movies that users are likely interested in. This technology is widely used in today's online e-commerce websites like Amazon, Tmall and so on. There are many techniques that can be used to build such a system. And in this project, we plan to build a simple movie recommendation system model and by using this model, we can generate movies for our users.

1.2 Project Time line

1.2.1 Date:09/04–09/25

Search for topics that we are interested in and decide which topic to choose;

Search for some documents and thesis and have a basic understanding of what techniques we are going to use.

1.2.2 Date:09/26–10/16

Choose the data we are going to work on from open source data base, describe the data format and data processing flow.

Describe the data content using spark.

Describe the next step that we will proceed with on processed data.

1.2.3 Date:10/17–12/14

Achieve a simple movie recommendation system model. Use this model to do some tests to verify the capacity of our model and evaluate the results numerically or via visualizations.

Write the final report.

2 Data Source & Format & Preprocessing

2.1 Data source

The data we use to do experiments and to draw conclusion from in this project is MovieLens 1M Dataset (link: https://grouplens.org/datasets/movielens/), which contains 1 million ratings from 6000 users on 4000 movies(Released 2/2003). It is from MovieLens data sets, which are collected and made available from MovieLens web site (link: http://movielens.org) by GroupLens Research. MovieLens data sets are the most popular sets used for relevant scientific researches for their accuracy and authenticity.

2.2 Data Format

MovieLens 1M Dataset contains three major files, which are movies.dat, ratings.dat and users.dat. Below we will show the data format of each file.

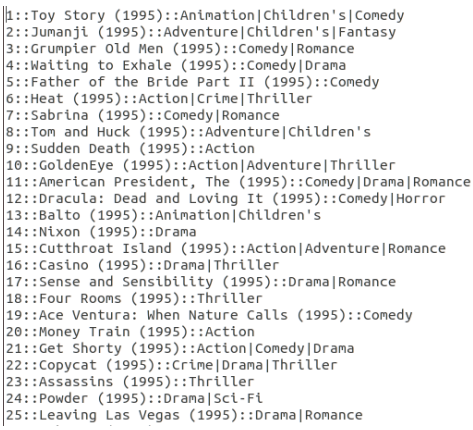
1 movies.dat

Movie information is in the file "movies.dat" and is in the following format:

MovieID::Title::Genres

Below is a screenshot of the data for clearer illustration:

Figure 1: data in movies.dat



```
1::Toy Story (1995)::Animation|Children's|Comedy
2::Jumanji (1995)::Adventure|Children's|Fantasy
3::Grumpier Old Men (1995)::Comedy|Romance
4::Waiting to Exhale (1995)::Comedy|Drama
5::Father of the Bride Part II (1995)::Comedy
6::Heat (1995)::Action|Crime|Thriller
7::Sabrina (1995)::Comedy|Romance
8::Tom and Huck (1995)::Adventure|Children's
9::Sudden Death (1995)::Action
10::GoldenEye (1995)::Action|Adventure|Thriller
11::American President, The (1995)::Comedy|Drama|Romance
12::Dracula: Dead and Loving It (1995)::Comedy|Horror
13::Balto (1995)::Animation|Children's
14::Nixon (1995)::Drama
15::Cutthroat Island (1995)::Action|Adventure|Romance
16::Casino (1995)::Drama|Thriller
17::Sense and Sensibility (1995)::Drama|Romance
18::Four Rooms (1995)::Thriller
19::Ace Ventura: When Nature Calls (1995)::Comedy
20::Money Train (1995)::Action
21::Get Shorty (1995)::Action|Comedy|Drama
22::Copycat (1995)::Crime|Drama|Thriller
23::Assassins (1995)::Thriller
24::Powder (1995)::Drama|Sci-Fi
25::Leaving Las Vegas (1995)::Drama|Romance
```

Some further explanations below:

(1) Titles are identical to titles provided by the IMDB (including year of release)

(2) Genres are pipe-separated and are selected from the following genres:

* Action * Adventure * Animation * Children's * Comedy * Crime * Documentary * Drama * Fantasy * Film-Noir * Horror * Musical * Mystery * Romance * Sci-Fi * Thriller * War * Western

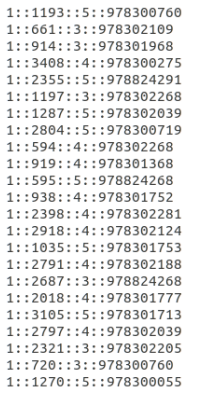
2 ratings.dat

All ratings are contained in the file "ratings.dat" and are in the following format:

UserID::MovieID::Rating::Timestamp

Below is a screenshot of the data for clearer illustration:

Figure 2: data in ratings.dat



```
1::1193::5::978300760
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
1::1287::5::978302039
1::2804::5::978300719
1::594::4::978302268
1::919::4::978301368
1::595::5::978824268
1::938::4::978301752
1::2398::4::978302281
1::2918::4::978302124
1::1035::5::978301753
1::2791::4::978302188
1::2687::3::978824268
1::2018::4::978301777
1::3105::5::978301713
1::2797::4::978302039
1::2321::3::978302205
1::720::3::978300760
1::1270::5::978300055
```

Some further explanation:

(1) UserIDs range between 1 and 6040

(2) MovieIDs range between 1 and 3952

(3) Ratings are made on a 5-star scale (whole-star ratings only)

(4) Timestamp is represented in seconds since the epoch as returned by time(2)

3 users.dat

User information is in the file "users.dat" and is in the following format:

UserID::Gender::Age::Occupation::Zip-code

Below is a screenshot of the data for clearer illustration:

Figure 3: data in users.dat

```
1::F::1::10::48067
2::M::56::16::70072
3::M::25::15::55117
4::M::45::7::02460
5::M::25::20::55455
6::F::50::9::55117
7::M::35::1::06810
8::M::25::12::11413
9::M::25::17::61614
10::F::35::1::95370
11::F::25::1::04093
12::M::25::12::32793
13::M::45::1::93304
14::M::35::0::60126
15::M::25::7::22903
16::F::35::0::20670
17::M::50::1::95350
18::F::18::3::95825
19::M::1::10::48073
20::M::25::14::55113
21::M::18::16::99353
22::M::18::15::53706
23::M::35::0::90049
24::F::25::7::10023
25::M::18::4::01609
```

Some further explanation:

(1) Gender is denoted by a "M" for male and "F" for female

(2) Age is chosen from the following ranges: * 1: "Under 18" * 18: "18-24" * 25: "25-34" * 35: "35-44" * 45: "45-49" * 50: "50-55" * 56: "56+"

(3) ccupation is chosen from the following choices: * 0: "other" or not specified * 1: "academic/educator" * 2: "artist" * 3: "clerical/admin" * 4: "college/grad student" * 5: "customer service" * 6: "doctor/health care" * 7: "executive/managerial" * 8: "farmer" * 9: "homemaker" * 10: "K-12 student" * 11: "lawyer" * 12: "programmer" * 13: "retired" * 14: "sales/marketing" * 15: "scientist" * 16: "self-employed" * 17: "technician/engineer" * 18: "tradesman/craftsman" * 19: "unemployed" * 20: "writer"

2.3 Data preprocessing

The data stored in the "*.dat" is a number of rows of byte string and the data in the "*.csv" is a Dataframe with column indices. Since the data structure used in the Spark is Dataframe, in order to simplify the data analysis process when exploring the data, we choose to transfer the raw byte data in "*.dat" to data form stored in the "*.csv" files and replace the tags of occupations and ages with their corresponding name of occupation or range of ages.

First, for each row of bytes string, we decode them to be char string and then split the string to columns. For example, for each row in "ratings.dat", we use "::" as separator and transfer the string to a list with format "list([userid, movieid, rating, timestamp])".

Then, the next step is to use a map to replace the tags of occupations and ages in the Dataframe with their corresponding values such as "writer", "56+" and so on.

After all that have been done, the final step is to output the Dataframe to a csv file and add the column indices which specify the labels of each column. Then, the well defined Dataframe object is ready to be used in the next steps.

3 Data content

All the data in above three data files are discrete and scattered, we can not draw patterns from them. So this is where spark comes into pictures. By using spark, we can simply and quickly get many interesting result. Below we will show the experiments result we harvest from the raw data.

(1) number of movies of each genre + average ratings for each genre

Figure 4: number of movies of each genre

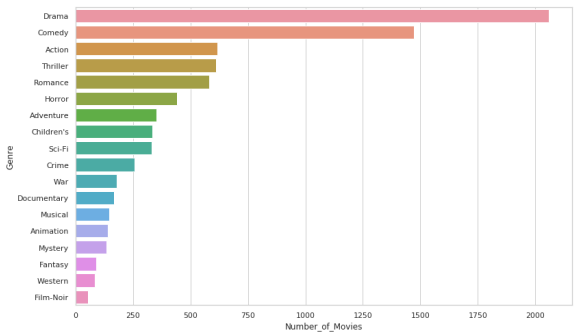
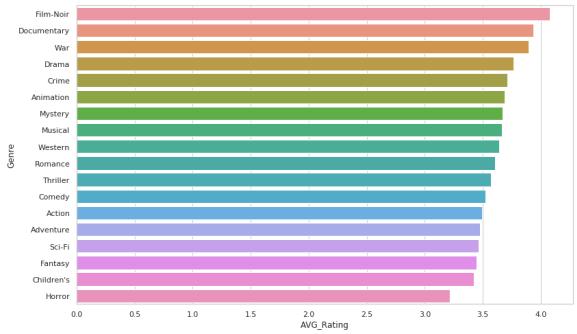


Figure 5: average ratings for each genre



We can find out some interesting results from above two figures. For example, Film-Noir is the least popular, but its average rating is the highest. This is reasonable in fact. Because Film-Noir is not popular, people who watch them must be fond of this type, therefore they are very likely to give high ratings.

(2) average number of rated movies according to career + average rating according to career

Figure 6: average number of rated movies according to career

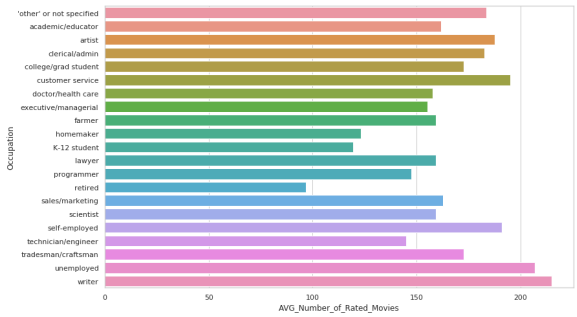
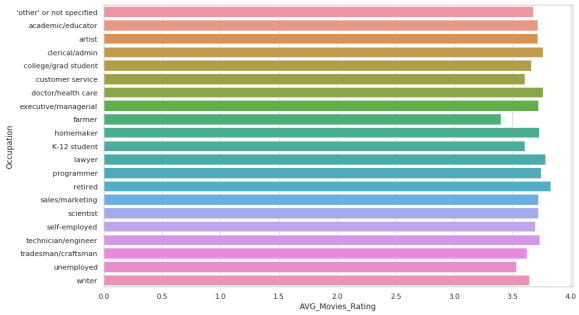


Figure 7: average rating according to career



From above, we can also draw some reasonable results. From the first chart, we can see that writers watch the most movies. That may be because they watch movies to get inspiration on their writings. Also unemployed people watch a lot of movie because they got lots of time to consume.

From the second chart, we can see retired people tend to give high ranks. That also comply with reality. elder people are mostly tolerant, even of a bad movie.

(3) average number of rated movies according to age distribution + average rating according to age distribution

Figure 8: average number of rated movies according to age distribution

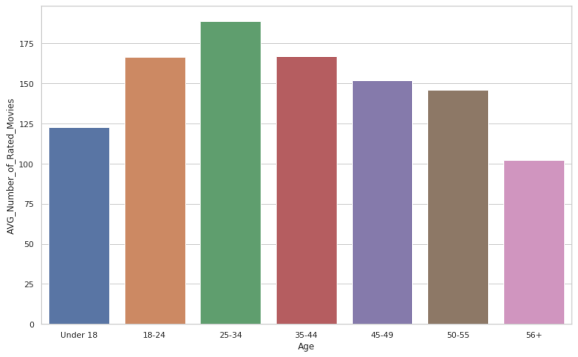
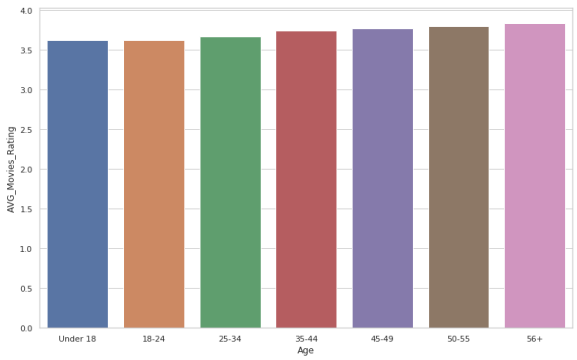


Figure 9: average rating according to age distribution



As we can see, people in age span 25-34 watch the most movies, while the number of movies rated decreases as the age of people increases. That is because older people are not as interested as younger people in fashion staff like movies.

And we can also find out older people gives higher ratings. This result complies with the result in the section above, where we get the result that retired people gives the highest ratings(reired people are usually older people).

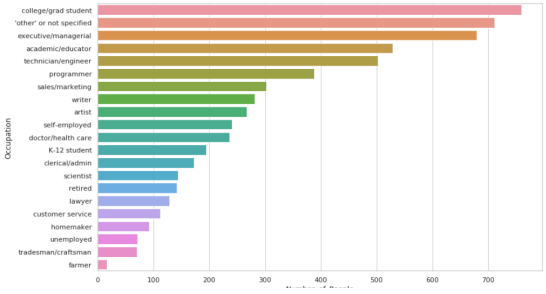
(4) Number of People for each occupation

Figure 10: Number of People for each occupation

Occupation	Number_of_People
college/grad student	759
'other' or not specified	711
executive/managerial	679
academic/educator	528
technician/engineer	502
programmer	388
sales/marketing	302
writer	281
artist	267
self-employed	241
doctor/health care	236
K-12 student	195
clerical/admin	173
scientist	144
retired	142
lawyer	129
customer service	112
homemaker	92
unemployed	72
tradesman/craftsman	70
farmer	17

To make it more intuitive, we draw it into a graph.

Figure 11: Number of People for each occupation



From the figure above, we can find out students are more engaged in this data collection while farmers are less engaged in, which are also comply with our expectation.

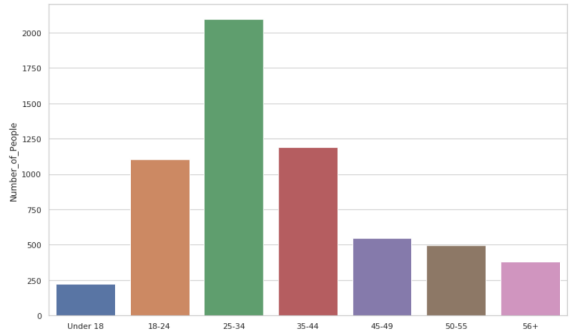
(5) Number of People for each age span

Figure 12: Number of People for each age span

Age	Number_of_People
Under 18	222
18-24	1103
25-34	2096
35-44	1193
45-49	550
50-55	496
56+	380

To make it more intuitive, we draw it into a graph.

Figure 13: Number of People for each age span



With the chart above, we can find out as the age increases, more people are engaged in data collection, reaching peak at 25-34. And then fewer people engaged in as age increases.

4 Possible Usage of Data

From the results we gain from the data, we might use these results as instructions when doing data collection. For example, we should pay more attention to younger and older people to get more data from them, so that we can get more balanced data.

Also because we got ratings in this data, we can try to design a real time movie recommendation system. When people type in the movie they watched with ratings, the system can give out the movies they might be interested in and expected ratings for people to choose from.

5 Collaborative Filtering

Like many other machine learning techniques, Collaborative Filtering makes prediction based on users' historical behaviors. Specifically, it's to predict user preference for a set of items based on past experience. To build a recommendation system using collaborative filtering, there are two approaches that are the most popular: user-based collaborative filtering and item-based collaborative filtering.

Let's first look at user-based collaborative filtering. The basic idea is to use the similarity between two different users. for example, we have an n*m matrix of ratings, with users u_i,i=1...n and movies u_j,j=1...m. Now if user i does not rate movie j, how do we calculate the expectation of rating $r_{i,j}$? The process is to calculate the similarities between target user i and all other users, select the top K similar users, and take the weighted average of ratings from these K users with similarities as weights.

Then let's go to item based collaborative filtering. Instead of computing the similarity between different users, we need to compute the similarity between different items. Suppose we also have the n*m matrix of ratings. Now if movie j is not rated by user i, how do we compute the expectation of rating $r_{i,j}$? The process is to calculate the similarities between target movie j and all other movies, select the top K similar movies, and take the weighted average of ratings from these K movies with similarities as weights.

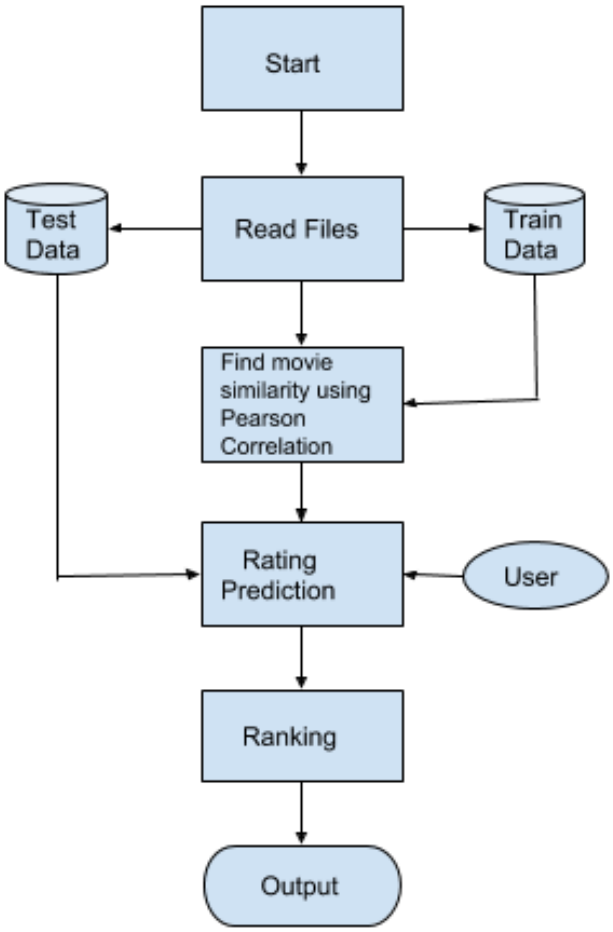
While both user-based collaborative filtering and item-based collaborative filtering are powerful in building recommendation system and the computing process is similar, there are some essential differences lying in between. Below we will explain the differences in detail.

Firstly, the core of collaborative filtering is to

compute the similarity matrix. In user-based collaborative filtering, when a new user comes into the recommendation system, we need to recompute the whole similarity matrix. The recommendation usually contains hundreds of thousands of users and movies. The computation of the similarity matrix is a time-consuming job. So the user-based collaborative filtering could not be used for real-time system.

Secondly, user-based collaborative filtering recommend movies according to similar users. While similar users may watch different genres, thus the recommendation result may go out of expectation. As to item-based collaborative filtering, the recommendation results are usually in same genre.

Due to the above reasons, we choose item-based collaborative filtering to build the recommendation system.



6 Flow diagram description

The flow diagram is shown as below, the point is database will be divided into Train data set and Test data set. Train data set will be used to compute movie similarity for every two movies/users and then get the similarity matrix in case of collaborative filtering.

7 High Level Pseudo Code

According to flow diagram, there will be three basic steps to get final recommended movies. Firstly, reading data into memory; then, calculating the similarity matrix using cosine or Pearson correlation; Lastly, getting recommended movies according to rating expectation.

Flow1: Reading file into program

```
set up two empty dictionary classes:
  TrainData={} and TestData={};
```

```
for each row in dataset:
  set up a random number used as
      possibility;
  according to the random number,choose a
      dictionary to put the value of this
      row in it
  split the row into for part namely UserID
      , MovieID, Rating and Timestamp;
  using UserID and MovieID as two keys, and
       put the ratings according to keys;
```

Flow2: calculate the similarity matrix

```
set up distance dictionary:Movie_Dis = {};
set up similarity matrix: Movie_sim = [];
for UserID in TrainData,
  calculate average ratings from each user
      Ru;
for UserID in TrainData,
  for m1 in TrainData[UserID],
    for m2 TrainData[UserID],
      if m1 != m2
           calculate the production of
               distance and put it in
               Movie_Dis [m1][m2];
for i to number of movies,
  for j to number of movies,
  using Movie_Dis to compute similarity of
      Movie_sim[i,j] (Pearson coefficient)
```

Flow3: recommend movies

```
given a user's data, traverse the movies he
     watched;
for each movie he watched,
  compute the expected rating of K(set by
      us) similar movie;
sort all similar movies and recommend the
    first L(set by us) movies;
```

8 Time and space complexity description

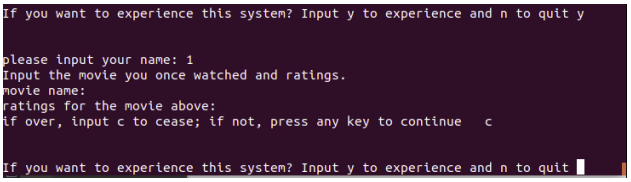
let's say we have n UserID, and m MovieID, the using of dictionary class decreases the indexing time from O(n)(using list) to O(1), so the we only need to pay attention to 'for' loops. And during the step from 'reading file' to 'recommend movies', the time complexity is $O(nm^2)$.

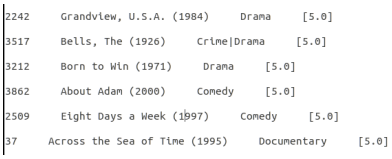
The biggest occupation of space lies in generating similarity matrix, which is O(mn).

9 Experiments results

In our experiments, we design two kinds of input:
–users in datasets
–new users with historical movies and ratings

Below are some screenshots of our experimental results. First let's see the recommendation results for users in datasets. We take user with userID 1 as an example. Because we use the user in datasets, we do not need to offer historical movies and ratings. So we just skip the 'input the movies you once watched and ratings' part and directly get the recommendation results as follows.





Then let's see recommendation results for new users. In this part, we need to fill out the movies and the ratings part. We choose the following movies and ratings:

86::White Squall (1996)::Adventure—Drama rating:5

590::Dances with Wolves (1990)::Adventure—Drama—Western rating: 3

3004::Bachelor, The (1999)::Comedy—Romance rating:2

The results are shown as below and comply with our expectation. In the inputs, we give 5 star rating to movie White Squall (1996), which belongs to adventure and drama. So we can expect that the recommendation system tends to give out movies of genre of adventure or drama. And as we can see, most of the recommendation results belong to drama genre.

```
If you want to experience this system? Input y to experience and n to quit y


please input your name: wxy
Input the movie you once watched and ratings.
movie name: 86
ratings for the movie above: 5
if over, input c to cease; if not, press any key to continue


Input the movie you once watched and ratings.
movie name: 590
ratings for the movie above: 3
if over, input c to cease; if not, press any key to continue


Input the movie you once watched and ratings.
movie name: 3004
ratings for the movie above: 2
if over, input c to cease; if not, press any key to continue   c


If you want to experience this system? Input y to experience and n to quit
```

```
##########################
1044    Surviving Picasso (1996)    Drama    [5.000000000000001]
614    Loaded (1994)    Drama|Thriller    [5.0]
1686    Red Corner (1997)    Crime|Thriller    [5.0]
662    Fear (1996)    Thriller    [5.0]
1401    Ghosts of Mississippi (1996)    Drama    [5.0]
1773    Tokyo Fist (1995)    Action|Drama    [5.0]
31    Dangerous Minds (1995)    Drama    [5.0]
1472    City of Industry (1997)    Crime|Thriller    [5.0]
1841    Gingerbread Man, The (1998)    Drama|Thriller    [5.0]
2893    Plunkett & MaCleane (1999)    Action|Drama    [5.0]
1570    Tetsuo II: Body Hammer (1992)    Sci-Fi    [5.0]
1930    Cavalcade (1933)    Drama    [5.0]
3743    Boys and Girls (2000)    Comedy|Romance    [5.0]
1647    Playing God (1997)    Crime|Thriller    [5.0]
2106    Swing Kids (1993)    Drama|War    [5.0]
2175    D◆j◆ Vu (1997)    Drama|Romance    [5.0]
1564    Roseanna's Grave (For Roseanna) (1997)    Comedy|Romance    [5.0]
2546    Deep End of the Ocean, The (1999)    Drama    [5.0]
2861    For Love of the Game (1999)    Comedy|Drama    [3.2841457191207812]
150    Apollo 13 (1995)    Drama    [3.0000000000000004]
```

10 Conclusion

In this report, we have used spark to analyze the original data and visualize the results in the first half part. In the second half part, we implement item-based collaborative filtering to achieve real-time movie recommendation (after computing the similarity matrix once). Item- based collaborative filtering is powerful and efficient in recommending movies, while there is still room for improvement. For example, using SVD algorithm to reduce the calculation in computing the similarity matrix.