CSIT5800 -- Introduction to Big Data

Mini-project final report – Group 7

Project title: Movie Recommendation System

Group members:

Name	Student ID	Email
Supriya Parthiban	20567171	sparthiban@connect.ust.hk
Huang Yi	20538986	yhuangdf@connect.ust.hk
Li Yilin	2056 3096	ylihy@connect.ust.hk
Yan Ziqi	20543632	zyanal@connect.ust.hk
Lu Yao	20552578	ylubx@connect.ust.hk

1. Introduction

Background

Present movie recommendation systems are generally divided into tree types: popularity-based engine, content-based engine and collaborative filtering engine. Popularity-based engine is the easiest, but the most impersonal one. Content-based engine is based on the description of movies, usually extract features, to represent each movie, and compute their similarities by these features. Collaborative filtering engines will also take information from users into consideration, the recommendation will make use of the similarities between various users.

Problem Definition

Our system is designed to give user suggestion, what movies they might get interested in. User type in a movie he or she like, or he or she want to watch movies like this, the system will select 5 movies from the database to recommend, these 5 movies are similar to the movie input, but not so similar, to avoid the high probability user has known it.

Related Work

Libraries we have used in our project: Nltk, Matplotlib, Json, Pandas, Numpy, Seaborn, Sklearn, Fuzzywuzzy, Math, Wordcloud.
Nltk is a library which is used to process natural language, we use it to recognize same word in different form. Matplotlib and seaborn are libraries we used to visualize the data. Json is a library we use to process json file. Pandas and numpy are widely used libraries, we use these two to process data and do the matrix calculation.

2. Design and Implementation

Basic Design of Our System

This movie recommendation system is mainly based on content-based engine, that is to say, the most important part is to extract attribute from each movie's information, convert them to a vector to represent these movies, apply formula on these vector, and the result represent the similarities between these movies. However, this system recommend movies that are similar to the movie that user input, but would not only refer to the similarity between the movie that user input and movies in the dataset we used, cause users usually tend to expect movies he or she haven't heard before, thus additional fine tuning would be apply on the final result.

Exploration

The dataset we used has 26 attributes and 50,000 instances, some attributes might have missing values.
Two important features are visualized as pictures below.

Figure 1: Genres Popularity

According to figure 1, the genres of most films in the database we used in this project is drama, then comedy, third thriller and romance.

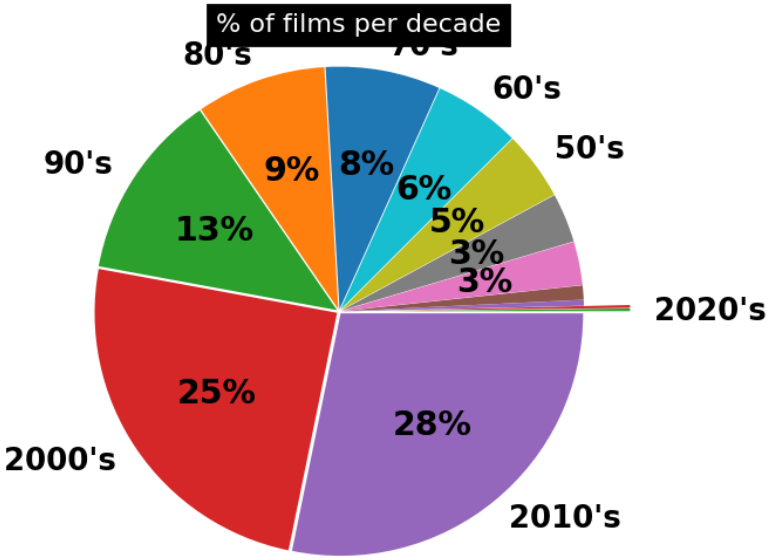

Figure 2: Releasing Years Distribution

The releasing year of films is combined into a decade to reduce complexity. According to picture, most films was published in 2010's, more than half of the film, are published in 20th century. There's a film published in 2020's, it is a movie which is underway, in other word, haven't been published yet.

Data pre-processing

Duplicated entries

```
Num of entries that have the same movie title as some other entries: 5460
The details:
                        movie_title  title_year            director_name
42660   10 Minutes                    2014.0      Iván Dariel Ortiz
24402   10 Minutes                    2002.0      Stefan Fjeldmark
15194   12 Angry Men                  1997.0      Yosuke Fujita
1161    12 Angry Men                  1957.0      Sidney Lumet
32754   12 Chairs                     1976.0      Xavier Palud
32768   12 Chairs                     1971.0      Charlie Levi
12999   20,000 Leagues Under the Sea  1916.0      Jonathan Demme
24092   20,000 Leagues Under the Sea  1997.0      Jeremiah Zagar
20838   20,000 Leagues Under the Sea  1997.0      Kasper Barfoed
990     20,000 Leagues Under the Sea  1954.0      Richard Fleischer
20639   2:22                          2008.0      Jane Anderson
44924   2:22                          2017.0      Dziga Vertov
5165    3:10 to Yuma                  1957.0      Joel Zwick
12039   3:10 to Yuma                  2007.0      Marcello Fondato
```

Figure 3: Duplicated Entries

Some movies remake after a long time but may keep the same title to pay tribute to the classics. It is necessary to compare the director and main actors' name to verify whether the entry is valid or not.

Cleaning of the keywords

Keywords play a significant role in the function of this recommendation system. Although keywords describing the film could be subjective and multivariate, in order to quantize the similarity between films, processing on keywords values much to gauge the similarity.

a) Grouping by roots

Root is the core of the vocabulary that derivation keywords could be substituted by more popular roots to reduce the computing complexity. The list of keywords is cleaned with the NLTK library taking advantage of its natural language processing concept.

```
Num of keywords in variable(feature) 'plot_keywords': 18571
get a feel of a sample of keywords that appear in close varieties
1 {'toy', 'toys'} 2
2 {'boy', 'boys'} 2
3 {'friend', 'friends'} 2
4 {'fish', 'fishing'} 2
5 {'best friends', 'best friend'} 2
6 {'divorced', 'divorce'} 2
7 {'confidence', 'confidant'} 2
8 {'banking', 'bank'} 2
9 {'murders', 'murderer', 'murder'} 3
10 {'suspension', 'suspense'} 2
11 {'betrayal', 'betrayed'} 2
12 {'gangs', 'gang'} 2
13 {'explosions', 'explosive', 'explosion', 'explosives'} 4
14 {'widow', 'widower'} 2
```

Figure 4: Grouping by Roots

b) Groups of synonyms

Cleaning the synonyms is divided into two steps. As the first step, keywords that appear less than five times would be replaced by a synonym of high frequency. As the second step, all the keywords suppress again to delete those keywords that appear in less than three films. The replacement only concerns less than 5% of the keywords while the total number of meaning keywords drops from 19052 to 6009 to save a lot of computing time.

```
Keywords with frequency lower than 6 are replaced bytheir synonymes with the most frequency.
For examples(output is different every time the program runs)...
foodie       -> gourmet      (init: [('gourmet', 1), ('foodie', 1)])
sink         -> sinkhole     (init: [('sinkhole', 2), ('sink', 1)])
network      -> net          (init: [('net', 2), ('network', 1)])
belly        -> stomach      (init: [('stomach', 1), ('belly', 1)])
steel        -> sword        (init: [('sword', 69), ('blade', 8), ('steel', 1)])
dehydration  -> drying up    (init: [('drying up', 1), ('dehydration', 1)])
fissure      -> crack        (init: [('crack', 6), ('fissure', 1)])
--------------------------------------------------------------------------------
The replacement concerns 4.74% of the keywords
```

Figure 5: Groups of Synonyms

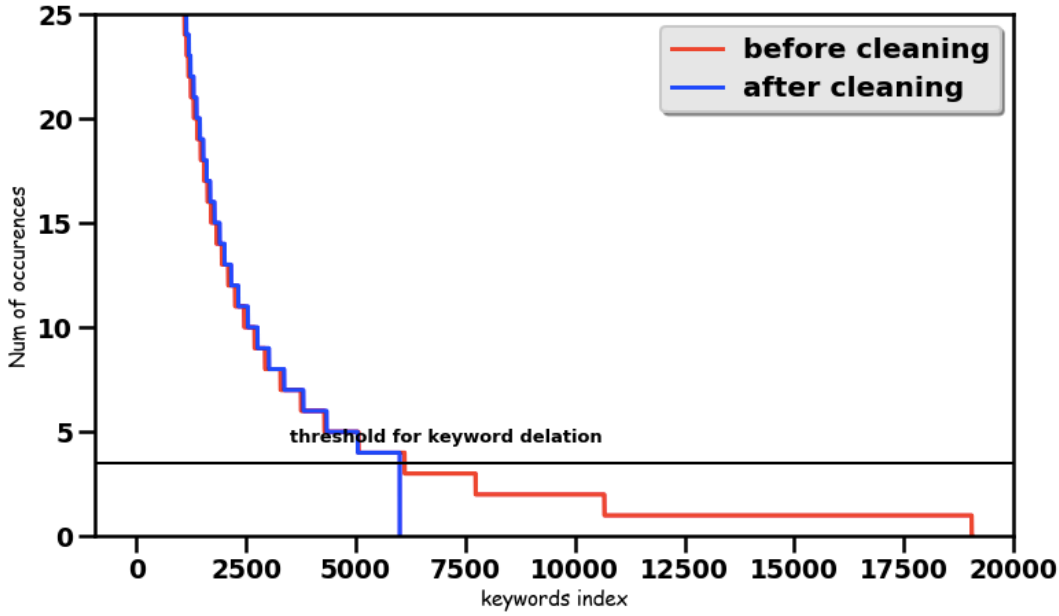

Figure 6: Number of Occurrences

Correlations

Processing the correlation results shows the internal relationship of different factors. By eliminating part of the low correlation variables from the data frame, the final diagram presents below.

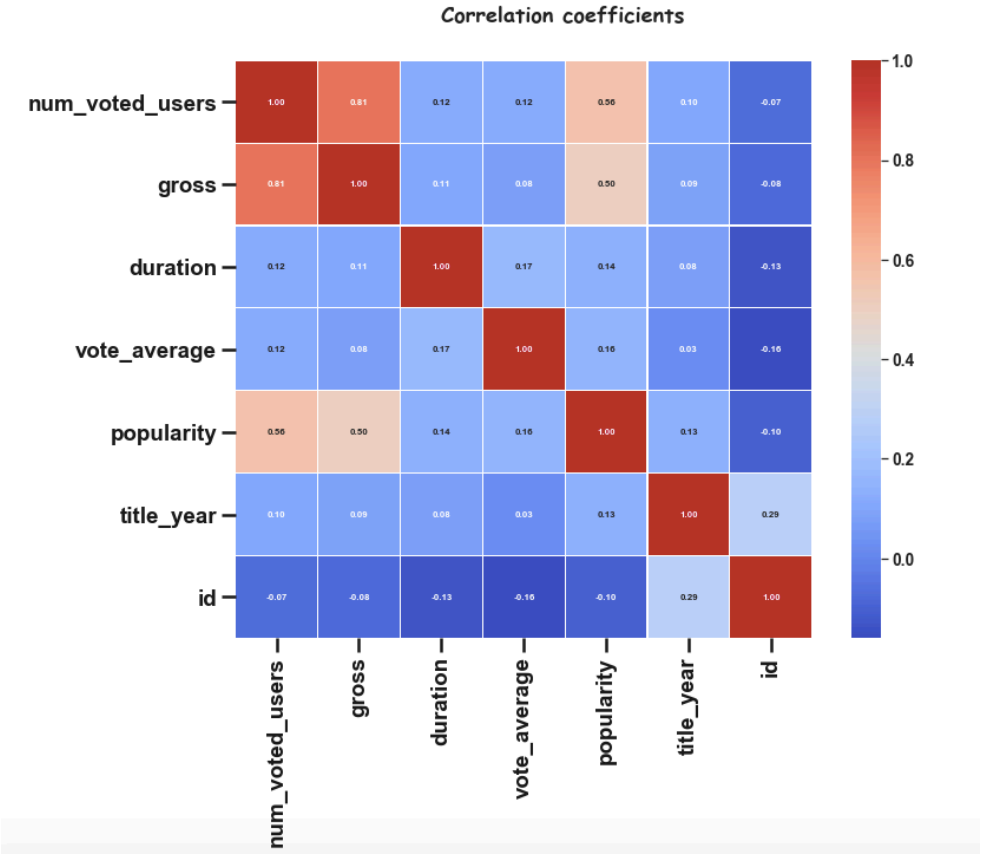

Figure 7: Correlations

Missing Values

a) Setting missing title years

For some of the films missing the title years, we use the mean year of activity for list of actors and directors in current dataset to estimate the title year.

b) Extracting keywords from the title

As previously outlined, keywords gauge the similarity. For films lack keywords, we use words of the titles to fill the missing values. First, we create the list of synonyms of all the words contained in the title and check if any of these synonyms are already in the keyword list to turn out to be a better result.

c) Imputing from regressions

According to the diagram presented above, gross and num_voted_users show a higher degree of correlation with a Pearson's coefficient >0.5. Therefore, we use this finding to fill the missing values by making regressions on pairs of correlated variables.

Recommendation Engine

a) Basic functioning of the engine

Basically, our algorithm can be divided into two steps:

(1) In the first step, we determine *N* movies with a content that is considered as similar to the movie provided by the user.
(2) In the second step, we select the 5 most popular films among these *N* movies.

Actually, we perform the second step based on two assumptions. One is that users are not only interested in a similar movie but also interested in a good movie in turns of quality. Another one is that a movie with high quality tends to have high popularity. Therefore, we assess the quality of a movie by measuring its popularity.

b) Algorithm

(1) Similarity

In this step, we need to define the criteria that tells us how similar two movies are. To do so, we start from the description of the film that was selected by the user. From it, we get the director name, the names of the actors and its keywords and genres. Based on these features, we build a matrix where each row corresponds to a film of the dataset and each column corresponds to the previous features value. An example of the similarity matrix is shown below:

movie title	director	actor 1	actor 2	actor 3	keyword 1	keyword 2	genre 1	genre 2	...	genre k
Film 1	a_{11}	a_{12}			...					a_{1q}
...					...					
Film i	a_{i1}	a_{i2}			a_{ij}					a_{iq}
...					...					
Film p	a_{p1}	a_{p2}			...					a_{pq}

Figure 8: Similarity Matrix

In the matrix shown in the figure above, each entry takes either the value 0 or 1 depending on the correspondence between its' column and its' row. For example, if the director of the given movie is Nolan, then we check the director of the first movie, if it's also Nolan, then the entry a_{11} will be marked as 1, otherwise it will be 0. After computing every entry, we get the whole matrix. Once we get the whole matrix, each movie can be represented by a corresponding vector. Using these vectors and the formula shown below, we can calculate the distance between each of these movies and the given one.

$$d_{m,n} = \sqrt{\sum_{i=1}^{N}(a_{m,i} - a_{n,i})^2}$$

N is the number of the columns of the similarity matrix. $a_{m,i}$ and $a_{n,i}$ is the value the of the i-th entry of movie m and movie n separately and $d_{m,n}$ is the final distance between movie m and movie n.
In the end, we sort all the movies by their distance from the given one in increasing order and select the N most similar movies to perform the second step.

(2) Popularity
In this step, as mentioned above, we need to measure the popularity of each movie. We decide to compute the popularity score based on 3 criteria — the IMDB score, the number of votes and the release year. As we can see, the first two criteria are direct measurements of the popularity. The reason why we also choose the third criteria, the release year, is that we have made another assumption that people will prefer the movies from the same epoch, in our case, from the same decade. Therefore, we took the releasing year into our consideration. Below is our formula to compute the similarity score.

$$score = IMDB^2 \times \phi_{\sigma 1,c1} \times \phi_{\sigma 2,c2}$$

where ø is a gaussian function:

$$\phi_{\sigma,c}(x) \propto \exp\left(-\frac{(x-c)^2}{2\sigma^2}\right)$$

For votes, we get the maximum number of votes among the N films and we set $\sigma_1 = c_1$

= m. For years, we put $\sigma_1 = 20$ and we center the gaussian on the title year of the film selected by the user. With the gaussians, I put more weight to the entries with a large number of votes and to the films whose release year is close to the title selected by the user.
After computing the popularity score of all the N movies that we get from step one, we choose 5 movies with the highest score and return it to the user.

2. Evaluation

How do we evaluate our system?

Frankly speaking, it is difficult to evaluate a recommendation system without user data because there is no general standard to judge a result as good or bad. Whether the recommended result is good or not highly depends on the taste of users. A result that can satisfy one user might not be able to satisfy the others. In the end, we decided to evaluate our system by comparing its' results with the results of IMDB because we download the dataset from it. Since the IMDB are much more mature than our system, we can consider their results as standard to our system. Moreover, the IMDB actually get much more information than us by recording the viewing history of users, users' rating about other films, and the feedback of the users after recommendation, etc. We can see that the recommendation results of IMDB will be very different from ours. Therefore, we decide if our system recommend one movie that in the results of IMDB, we will call it a pass.

Evaluation results

We have randomly tested 50 movies to see if the results share common movies with that of IMDB. Here I show some examples below:

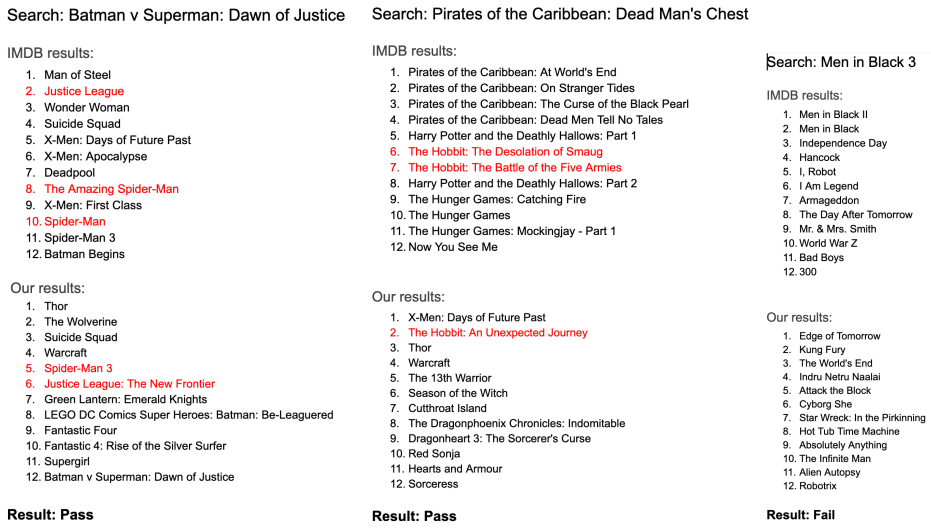

Figure 9: Recommendation with Sequels

As shown above, among three test cases, we got two passes.

3. Conclusion

Challenges

During the process of this project, we have met a lot of challenges:
(1) Missing values in the releasing year of films
In this case, we cannot use some standard approach to fill the missing values, like taking the average/mean value of that feature or using the most common value, because it does not make sense. Instead, we estimate the activity period of the director and the actors and use it to fill the missing values.

(2) Making meaningful recommendation
In the middle of the project, we found that if we only use similarity and popularity to give the recommendation without filtering the results, it will be highly possible that the system returns movies that are sequels of the given one. An example is shown in the figure below:

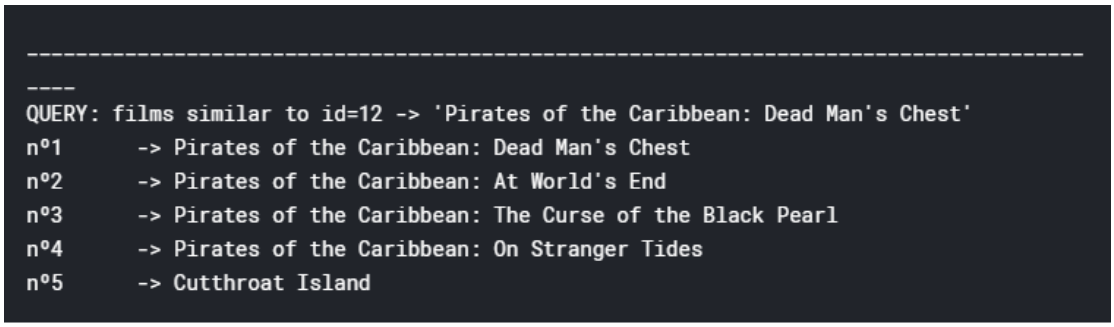

Figure 10: Recommendation with Sequels

However, we think that if one user has already watched one movie in one series, it is highly possible that he has also watched the other movies from the same series. At least, he would know these movies without recommendation. Therefore, we think that these results will not satisfy the user and we remove the sequels from the results. The modified result is shown below:

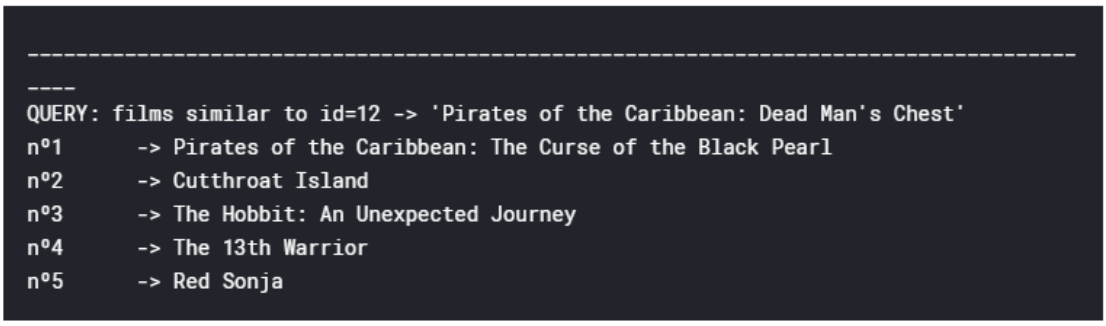

Figure 11: Recommendation without Sequels

Personal Thoughts

Through this mini-project, we have realized the big influence that the big data could bring to our lives. Even the recommendation system we have made is still immature, it can still make people's life more convenient to a certain extent, saving people's time struggling with the movies they do not like. During the process of this project, each member of our group has learned a lot, realizing the importance of flexible thinking. For example, when we were dealing with the missing values of the releasing years of movies, at first, we can only think of some standard approach like taking the average of the features, but it is obviously not correct. It took us some time to come out with the idea of taking the activity period of the director and the actors. This makes us realize that we cannot treat every feature the same and directly apply the same method to it. We need to think about its' meaning and find a reasonable approach. Also, teamwork is very important. Without the effort from every group member, we cannot finish this project.

Possible improvements

Our system is still immature. Here are some suggestions for the following work:
(1) Considering more features. The dataset has 27 attributes and we have not used all of them. Maybe there are some features that can hele the system make more meaningful recommendation.
(2) Try to reduce popularity bias. If we can introduce some user data, we can try to make personalized recommendation.
(3) Hybrid Approach. With user data, we can do collaborative filtering and combine it with content-based similarity to give a better result.
(4) Other techniques. We can introduce some techniques like regression and clustering to see if we can find something new.

4. References

http://sdsawtelle.github.io/blog/output/mean-average-precision-MAP-for-recommender-systems.html

https://www.nltk.org/_modules/nltk/stem/porter.html

https://www.imdb.com/list/ls052347299/

https://towardsdatascience.com/recommendation-systems-models-and-evaluation-84944a84fb8e

https://www.kdnuggets.com/2019/04/building-recommender-system.html

https://www.quora.com/What-metrics-are-used-for-evaluating-recommender-systems