

## **Design of movie recommendation systems based on collaborative filtering algorithm**

Team members:

Yuke Li	z5176261
Zanning Wang	z5224151
Yuankai Zhao	z5233940
Zhenming Wang	z5140192

All files including dataset, code and readme:

<https://drive.google.com/drive/folders/1E0E0zxnqutquUiL5yKR-iaI2IjoHr53?usp=sharing>

### **Introduction**

Collaborative Filtering (CF) algorithm is currently the most successful recommendation system algorithm, which is widely used in many well-known recommendation systems, such as Netflix, YouTube, Amazon, etc. In practical situations. Based on various users 'habit and unpredictable rating conditions, a single model movie recommendation system tends not to meet users' diverse demands. Therefore, to tackle with the singularity of movie recommendation system, this report implements three algorithms to satisfy different group of users.

In view of the different characteristics of users, we propose two types of models, namely user-based CF and item-based CF. Among them, the item-based CF class contains two models, namely Item-item CF recommendation based on K-Nearest Neighbor (KNN) and User-item CF recommendation based on Alternating Least Squares (ALS). This model is mainly oriented to make recommendations when users have few comments in the database. In the user-based CF category, user-user CF recommendation is mainly for users who give more rates in the database to make recommendations for the user. When the user is a new user, we can use the cold start model for recommendation.

### **Model Design**

According to different working principles, the recommended systems can be divided into the following types: item-based collaborative filtering recommendation, user-based collaborator filtering recommendation, Content-based Recommendation, and demographic-based recommendation. In this report, we mainly focus on two types of models, namely user-based CF and item-based CF.

The construction process of the recommended model is divided into the following steps :

(1) Construct training set and prediction set

- (2) Train the recommendation system model
- (3) Evaluation recommendation results

## ITEM-BASED CF:

### ◆ Item-item collaborative filtering recommendation (File: KNN.py)

The main idea of Item-based collaborative filtering systems is to calculate the ratings of different users for different movies to obtain the relationship between items and obtain the similarity between movies and movies so that we could recommend the similar movie to the user.<sup>[1]</sup> In this recommended system, we will train a KNN models to cluster similar movies based on user's ratings and develop our own evaluation system to estimate the accuracy of recommendation.<sup>[1]</sup> According the questions we mentioned at the beginning of report, this model make recommendations when users have few comments in the database.

## Implementation

### (1) Construct training set and exploratory data analysis

Through the preliminary observation of the original data, we first need to clean the data and process the basic evaluation parameters, such as the frequency of each movie's rating and the number of each segmented movie.

When dealing with the number of occurrences of each rating, due to the uneven user ratings, we found that most of the movies were not rated, and the number of 0 ratings was much higher than the number of other segmented movies. To solve this problem, we take the logarithm of the number of movies in each segment to compare the number of segments. The final result is shown below. By observing the picture below, we find that most users tend to rate movies between 3 and 4.

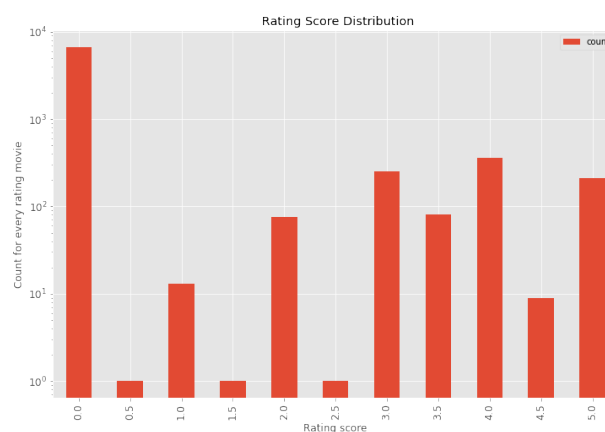


Figure.1.Rating Score Distribution

### (2) Train the recommendation system model based on KNN model

For the KNN model, we first need to build one, each row of user Id, each column is a matrix of different movie ratings. Since it is impossible for users to watch every movie, most of the ratings are empty. In order to calculate the distance of the item, we need to process the empty part of the

data. By weighting the users' rated movies, we can get the user's rating expectations for unknown movies, use this number as the missing observation value, and perform linear algebra operations.

Then we can substitute the existing test data into the model, and we can get the cosine value of every two points. The larger the cosine value, the higher the similarity between the two movies, and the closer the distance in the coordinate system. After completing the above work, we have completed the preliminary model establishment, and we can already use the above model to make personalized recommendations based on the given movies.

### (3) Evaluation recommendation systems

#### (a) According rating frequency of all movies filter the recommended movies

According to the existing recommendation system, we found that due to the existence of a large number of unpopular movies, the cos distance between the movies is far, and the accuracy of the movies pushed to the user is low. So, we decided to screen some user as a new dataset. By plotting the grade frequency of each user, we found that there is a long-tail property in the movie evaluation system. That is, only a small number of user have given a certain number of reviews. In order to narrow the gap in the number of rating for different users, we also log the number of movie scores.

By observing the table below, we can find that the number of the reviews are concentrated in the top 120 users. These users who give at least 120 ratings, are believed to be able to give ratings more fairly.

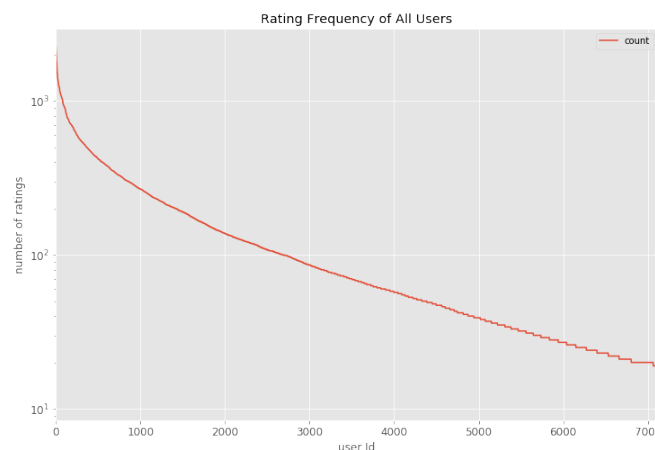


Figure.2.Rating Frequency of All Users

#### (a) Establish an evaluation mechanism

Since the algorithm we adopted is an assisted filtering model based on KNN algorithm, which is an unsupervised learning model, it is difficult to have an evaluation mechanism to directly evaluate the quality of movies recommended by this model due to its own model characteristics. We designed a score based on two dimensions to reflect whether the movie is suitable for users and evaluate whether the recommended movie is accurate or not.

The evaluation mechanism is mainly based on the following two dimensions:

- 1) The overall quality of the movie recommended to the user, that is, the score of the movie recommended to the user. In most cases, movies with relatively high scores have a better reputation, and the quality of the movies themselves is relatively strong. Compared with movies with low scores, they are more likely to be liked by users. At the same time, in order to reflect the relationship between recommended movies and user preferences, we weight the

movies that are similar to the user's preferences, that is, the closer the cos distance to a given movie, the higher the weight.

- 2) Consistency of the types of movies recommended to users : That is, we hope that our recommendation system, the classification of recommended movies is as concentrated as possible. However, because a movie may carry multiple tags due to its own characteristics, we only count the first three tags and weight the top three tags separately, to reflect the concentration and dispersion of recommended types.

Based on the above two dimensions, we prefer our model to suit the preference of users. Therefore, for these two dimensions, we assign a higher weight to the latter, that is, whether the movie types are consistent. In summary, we can initially obtain the evaluation criteria of our model, the formula is as follows:

$$Model_{score} = 0.7Part_1 + 0.3Part_2$$

$$Part_1 = \frac{\sum_i^k Score_{Movies_i}(10-i)}{i}$$

$$Part_2 = 7 \cdot \sigma_1 + 5 \cdot \sigma_2 + 2 \cdot \sigma_3$$

Among them, part1 and part2 respectively represent two evaluation dimensions,  $Score_{Movies_i}$

Indicates the score of the recommended movie,  $\sigma_1$ ,  $\sigma_2$ ,  $\sigma_3$  Respectively indicate the number of occurrences of the top three tags.

## Results

### 1) Adjust and optimize K in the KNN model

Adjust and optimize K in the KNN model. Based on the evaluation mechanism that has been proposed, we adjust the k value to calculate the performance of average scores of movies recommended. K value which makes most recommendations best will be our best K. The following table is obtained, and the result shows that when k=9, our recommendation is best.

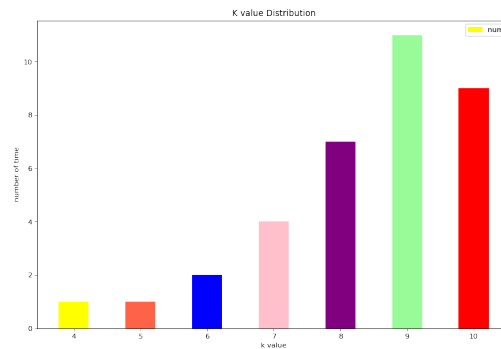


Figure.3. K value Distribution

### 2) Change the way of measuring distance in the KNN model

Regarding the parameters that measure the distance between items in the model, we tried the following five distance calculation methods to count the scores respectively, and get the following table. The cosine distance has the highest score, so the cosine distance is selected.

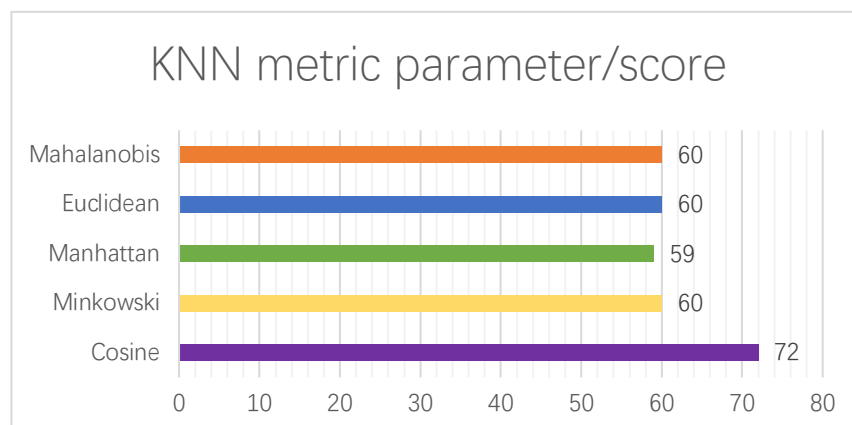


Figure.4. KNN metric parameter/score

### 3) Optimize model results by filtering valid data

Because movies with fewer raters and user who rates less are not of reference value. Therefore, according to the previously established evaluation model, the score with different threshold of different users and movies are calculated. The following table is obtained. When top 40% rated movies and top 25% raters are taken, the score is the highest, which is the best proportion for this model.

Table.1. Different Threshold of Different Users and Movies

User_Thres \ Movie_Thres	10%	15%	20%	25%	30%	40%	45%	50%	55%
10%	62	62	55	67	67	60	60	53	59
15%	55	60	53	53	55	55	55	55	55
20%	67	65	62	60	53	65	66	60	60
25%	58	59	54	52	55	56	54	54	53
30%	55	55	55	52	52	56	61	69	62
35%	55	48	48	60	67	68	66	66	54
40%	67	64	63	72	60	64	65	65	66

### ◆ Item-user collaborative filtering recommendation (File: ALS.py)

This model is based on the ALS algorithm. By observing the scores of all users on the product, using alternating least squares to solve, and then determine the movie with the highest score for the user, and recommend the most suitable product to the user.

## Implementation

### (1) Process the database and initialize the spark environment

Process the original data and transform it into the first column as the user number, the second column as the movie number, and the third column as the tuples scored by the user. And Initialize

spark.

## (2) Train the recommendation system model based on ALS model

### 1) Determine whether the model is over-fitting.

As the number of iterations increases, the model may have over-fitting. We set the rank and reg\_parm parameters to the conclusion of the previous question. By observing the change of RMSE and the model fitting situation in different iteration times. According to the figure below, we can see that when the number of iterations is 3, the RMSE has become relatively flat. Therefore we set iteration as 10.

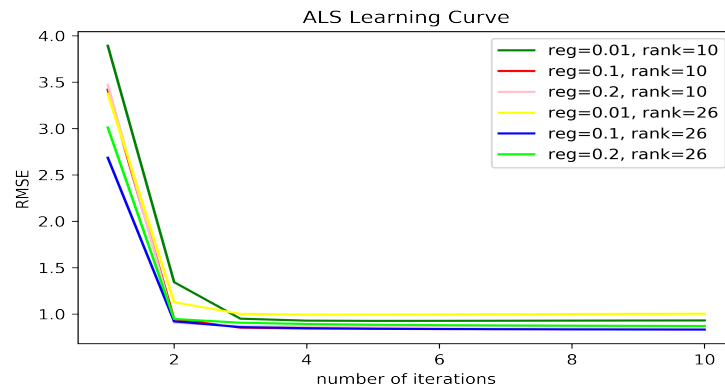


Figure.5.ALS Learning Curve

### 2) Find the best parameters.

Firstly, based on the ALS model, we use grid search to find the best parameters in the model. Including: iterations, reg\_parm, ranks. Among them, rank refers to the feature vector dimension. When the rank value is too small, the model is under-fitting, the error is large, the prediction is not accurate enough, however if the rank is too large, resulting in poor model generalization ability. Therefore, we initially set the value to 10-24. Reg\_parm refers to the parameters of regularization, and the initial setting range in the model is 0.01 to 0.2.

Set the fixed number of iterations to ten. When we solve for different ranks and reg\_parm, the root mean square error obtained. According to the root mean square error (RMSE) obtained, when the RMSE is smaller, the difference between the observed value and the true value is smaller, and the recommended result is relatively accurate. By adjusting ranks and reg\_parm to get the following table, we can see that when rank is 26 and reg\_parm is 0.1, the root mean square error is the smallest, the prediction is relatively accurate, and the model is relatively accurate.

Table.2. ALS training process (iteration number = 10)

latent factors \ Regularization	0.001	0.01	0.05	0.1	0.2	0.3
8	1.08614204	0.91227594	0.84285009	0.8334268	0.86754284	0.910064599
10	1.104750173	0.930928196	0.84645299	0.8330042	0.86686727	0.9099345
12	1.171099154	0.946000508	0.84913447	0.8347984	0.87158411	0.913635365
14	1.191530337	0.954689179	0.84751448	0.8333631	0.87110637	0.912685915
16	1.213797927	0.964634642	0.84984371	0.83378	0.87033019	0.912017056
18	1.239744218	0.975538539	0.85015799	0.8329827	0.87087262	0.912472631
20	1.259054526	0.984363188	0.85111173	0.8332503	0.87214816	0.913422641
22	1.286328944	0.995307798	0.85198152	0.8326721	0.87169809	0.913153273
24	1.298332523	1.000829849	0.85189455	0.8330613	0.87185278	0.913115193
26	1.310903175	1.003221919	0.84961665	0.8322009	0.87133256	0.912680496
28	1.308044861	1.007876674	0.85228596	0.8328284	0.87162867	0.912767593
30	1.312363014	1.007406512	0.85175005	0.8332061	0.87194762	0.91295076

### 3) Model testing and verification results

After preliminary completion of the model, check the validation data previously divided. What is the RMSE obtained 0.8291, Indicating that the model has a higher prediction accuracy. After completing the above work, personalized recommendations can be made for the movies input by the user.

## USER-BASED CF

### ◆ User-user collaborative filtering recommendation (File: movie\_recommender.py)

User-based collaborative filtering algorithm recommend the user's favorite movie or content through the user's historical behavior data (such as movie rating, content comment or sharing).<sup>[2]</sup> The main idea of user-based collaborative filtering recommendation is to find a group of "neighbors" users with similar preferences to the current user based on all users' preferences for the items.<sup>[2]</sup> The main process of User-based collaborative includes two steps:

- (1) Calculate the similarity between users based on the cosine value
- (2) Find a group of users with similar interests to the target user.
- (3) Recommend the items which the target users like most in this group, besides the target, users have not heard the recommended items.

Our model mainly recommends movies based on the user's portrait and matches users who are similar to the user by analyzing the movies that the user has watched. The system screens out movies that other users like but that the user has not watched and recommends them. This model focuses on analyzing user-level and is mainly suitable for the users who have many viewing records in the database. Recommend old users with this model to improve accuracy.

## Implementation

- (1) Processing raw data and convert users' ratings of movies into vectors.
- (2) Quantify the similarity between two users through Euclidean distance
- (3) Find users who are most similar to user preferences based on similarity
- (4) According to other users' movie watching record, recommend movies that the current user hasn't watched as a recommendation

## Result

By randomly selecting 30 users and using the evaluation system established by the previous model to score the movies recommended to these 30 users, the average score is 86.4, as shown in the following table, which shows that the system has a good evaluation effect. Because the recommendation system relies on current users with as many as score records. We use this model to make recommendations to older users with more ratings



Figure.6. Recommendation score

### ◆ Cold start model (File: cold\_start.py)

#### Implementation

The model uses the following formula to recalculate the scores for each movie, and arrange them in descending order, recommending the top ten movies to users.

$$Movie_{score} = \frac{v \cdot R + n \cdot C}{v + n}$$

Where  $v$  is the number of times the movie has been rated;  $n$  is the least number of times the movie has been rated,  $R$  is the average score of all movies, and  $C$  is the average number of ratings.<sup>[4]</sup>

#### Final Conclusion

Based on the four models we established above, we have successfully proposed a brand-new movie recommendation system that recommends different users. Because the accuracy of the User-user collaborative filtering recommendation is high, but it relies on the user's existing rating in the system, therefore this model is mainly aimed at old users who give many ratings. The cold start model does not depend on user preferences and is suitable for brand new users, that is, the user has no record in the system. The item-user CF and item-item CF models are mainly based on the movie as an item for judgment. This model is less dependent on users' scoring in the system, and is suitable for some old users who give fewer ratings.

#### Future work

1. How to improve the running speed, such as using spark to implement parallel computing and processing of the KNN model.
2. Recommendation algorithm based on neural network and Machine learning.
3. The recommended privacy and security issues cannot be ignored.



4. With the gradual improvement of users' personalization requirements, exploration and utilization issues are also worthy of attention.

## Reference

1. <https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-1-knn-item-based-collaborative-filtering-637969614ea>
2. <https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1>
3. <https://medium.com/fnplus/user-based-and-item-based-collaborative-filtering-b73d9b2badba>
4. kaggle URL on cold start  
`df_movies_cnt = pd.DataFrame(rating_data.groupby('movieId').size(), columns=['count'])`
5. user\_based model website: <https://www.jianshu.com/p/e793e5785850>
6. <https://realpython.com/build-recommendation-engine-collaborative-filtering/>
7. Sarwar, B., Karypis, G., Konstan, J. and Riedl, J., 2001, April. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295).
8. Wei, S., Ye, N., Zhang, S., Huang, X. and Zhu, J., 2012, August. Item-based collaborative filtering recommendation algorithm combining item category with interestingness measure. In 2012 International Conference on Computer Science and Service System (pp. 2038-2041). IEEE.
9. [https://github.com/KevinLiao159/MyDataSciencePortfolio/tree/master/movie\\_recommender](https://github.com/KevinLiao159/MyDataSciencePortfolio/tree/master/movie_recommender)
10. Hu, Y., Koren, Y. and Volinsky, C., 2008, December. Collaborative filtering for implicit feedback datasets. In 2008 Eighth IEEE International Conference on Data Mining (pp. 263-272). Ieee.


## APPENDIX

1. the dataset we used above: <https://drive.google.com/drive/folders/1E0E0zxnqutquUiL5yKR-iaI2IjoHzr53?usp=sharing>

2. Problem chosen as Topic 0:

Forums / Lectures / Week 6 / Project (Questions) / Hi, Omar, our team plan to create a recommended sy...


### Comments



Zanning Wang about 5 hours ago

Hi, Omar, our team plan to create a recommended system, and we try to find a suitable competition on the kaggle, however, there is only several dataset meet our requirement. According to your requirement, we are only allowed to work on a competition instead of a dataset. therefore we want to consult if we can work on a dataset on kaggle instead of previous competition, cause we have a really good idea on this dataset. the dataset : <https://www.kaggle.com/shivamb/netflix-shows>  
 Thanks in advance.

Reply Edit Delete



Omar Ghattas about 3 hours ago

this is fine, we can consider it a topic 0 choice, please send me an outline of the approach you plan to take over email though

Reply