Movie recommendation system

Yongxing NIE, Jia Xu, Yating Li
College of Engineering
Northeastern University
Toronto, ON
nie.yo@northeastern.edu
xu.jia2@northeastern.edu
li.yat@northeastern.edu

Abstract

In this project, three recommendation methods will be used to recommend movies to users. They are respectively demographic filtering, user-based/movie-based collaborative filtering, and a hybrid algorithm including content-based filtering and user-based collaborative filtering. The advantages and disadvantages of each recommendation method are explained. And the suitable recommender for a specific scenario is explored.

1 Introduction

In this project, we explore recommending systems by studying three different recommending mechanisms. Recommendation systems are interesting because they can help users discover new products or content that they may like, based on their previous preferences. This can be useful for businesses to help increase sales or engagement, and for individuals to help them find new things that they may enjoy. By building demographic recommender, collaborative filtering recommender, and a hybrid recommender including content-based filtering and user-based collaborative filtering on MovieLens dataset, we hope to grab a sense of how these popular recommenders are applied and adjusted in business.

2 Dataset

1.1 Data characteristics and data visualization

The MovieLens dataset contains rating data of multiple movies by multiple users. It contains four main csv files. This project is mainly based on two files: ratings. csv and movies. csv.

In movies.csv: movieId: The ID of the movie; **title**: The title of the movie; **genres**: Movie genres.

In ratings.csv: userId: The ID of the use; **movieId:** The ID of the movie; **rating**: The rating the user gave the movie; **timestamp**: The time the movie was rated.

1. The distribution map of movie lens dataset

The left picture shows that most of the users' rate movies only a few times. The right picture shows that most of the movies get only a few rates.

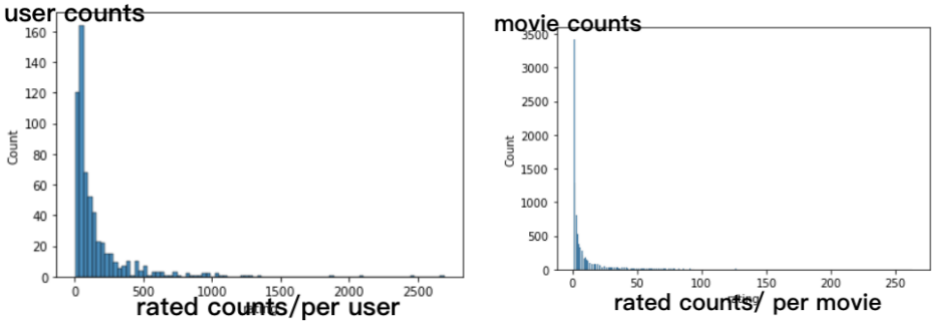

Fig. The distribution of the number of each user rated movies(left), the distribution of the number of the movies being rated(right).

1.2 Data processing

1. The data processing of movie lens dataset

To reduce the dimensionality of the dataset, I removed users whose movie rating actions are below 50 times. And I removed movies whose rating records are below 25. After the data processing, most movies have ratings of 3 or 4. And over 15K movies have ratings of 4, over 10K movies have ratings of 3.

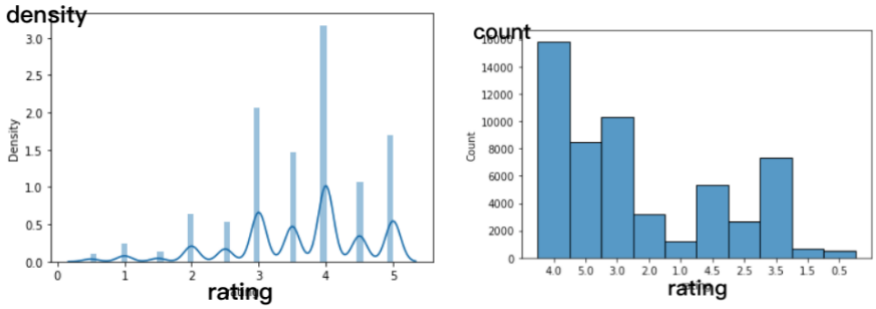

Fig. The distribution of the movie's ratings.

3 Methods （Algorithms and steps）

3.1 Demographic-based Recommendation

Demographic-based recommendation is a recommendation method that finds the relevance of the recommended product to the user based on the user's information, and then recommends the relevant products to the current user.

When recommending movies, the popularity of a movie is often an important indicator, and we can see the ranking of movie popularity on many movie-related websites, which is usually based on the number of viewers, user ratings or website searches. According to the dataset used in this project, the rating will be the basis of this recommendation model.

First filter out the movies with high ratings, i.e., high popularity

Then extract those movies that have not been rated by the given user ID as recommendations for that user.

*Here we assume that if a movie has not been rated by a user, then the user has not watched the movie.

3.2 Collaborative Filtering

There are two types of collaborative filtering (CF), one is to find users with similar movie patterns as the target user (user-based CF); the other one uses similarity between the movies to determine whether a target user would like it or not (movie-based CF). One advantage of movie-based CF is its stability since the ratings on a given movie will not change significantly overtime. The project intends to explore both CFs.

- 8 algorithms (some are suitable for user-based CF, some are suitable for movie-based CF) are calculated to find algorithms with lowest root-mean-square deviation (RMSE).
- Among the RMSE result, choosing 1 lowest algorithm for the model of user-based CF, and 1 for movie-based CF.
- When training user-based CF, cross validated randomized search was used to determine the best model used in selecting closest users and optimal parameters. When training movie-based CF, cross validated randomized search was used to tune the best parameters.
- After trained the model, cross validation was applied on the test data to get the mean RMSE result for each CF.

3.3 Hybrid Recommendation Algorithm -- Collaborative Filtering + Content-based Filtering

A good movie needs two conditions: high popularity and high rating. Therefore, in this algorithm, the rating will be an important judgment condition, and movies with a rating less than four points will not be recommended. The hybrid algorithm uses pipelined mixing to run the recommendation algorithm in a certain order, so as to gradually optimize the recommendation results.

This algorithm is based on the combination of the following two:

1. User based collaborative filtering; the potential assumption is that similar users may have similar preferences for similar movies.

2. Content based filtering, recommended according to genres popularity of each user.

Algorithm process:

1. First, use the user based collaborative filtering algorithm for the dataset to calculate the cos similarity between each user and others, select the user with the highest similarity and recommend the movies the user has seen.

2. If the highest similarity is less than 0.2, it is considered that no one is similar to the user, and the content-based filtering algorithm is used instead. If the most similar user hasn't seen a movie that is different from the user and has a rating of more than 4 points, it will also use the content-based filtering algorithm to recommend.

3. Content-based filtering algorithm, get the top two genres of each user's genres popularity ranking and their weights. When the movie rating is less than 4 points, it doesn't participate in the calculation of genres popularity. Then calculate the number of movies recommended by the genres according to weights. Also, the movies with different genres are sorted according to their ratings, and the top movies under the user's favorite genres are recommended.

4 Experiments and Results

4.1 Demographic-based Recommendation

	movieId	score	Rank	userId
274	318	247	2.0	628
416	480	236	4.0	628
98	110	213	7.0	628
933	1196	189	12.0	628
2308	2959	165	20.0	628
507	588	162	23.0	628
1004	1270	155	29.0	628
1233	1580	148	32.0	628
333	380	138	35.0	628
823	1036	137	37.0	628

Fig. The result of the recommendation for user ID 628

Because no historical data about the user is required, unlike the collaborative filtering algorithm, there is no "Cold Start" problem for new users. And this method does not rely on the data of the items themselves, so, it can be used in different fields of items.

However, this recommendation method does not pay enough attention to the user's preferences, so it cannot make personalized and fine recommendations. Also, if the recommendation is based on some users' personal information, it may not be easy to obtain the relevant data due to privacy reasons.

4.2 Collaborative Filtering

(1) The algorithm vs. RMSE result shows that, as a representative of user-based CF, KNN-Baseline has the lowest RMSE (0.827), while as the representative of movie-based CF, SVD has the lowest RMSE (0.836）.

Algorithm	test_rmse	fit_time	test_time
KNNBaseline	0.827600	0.087441	0.978913
KNNWithZScore	0.837072	0.071049	0.886953
SVD	0.837131	0.379715	0.070142
BaselineOnly	0.837329	0.055243	0.053302
KNNWithMeans	0.837449	0.050830	0.833833
NMF	0.857275	0.625134	0.048399
CoClustering	0.878268	0.561211	0.038611
KNNBasic	0.888124	0.039041	0.752947

Fig. The RMSE result of 8 different algorithms.

(2) The cross validated result of KNN-Baseline shows that the mean RMSE is 0.817. The SVD cross validated result shows that the mean RMSE is 0.847. When compared the recommending list, the recommending list from SVD and that from KNN-Baseline overlap a little bit. It can be concluded that both methods give good performance.

Table 1. KNN-Baseline and SVD recommending list comparison

KNNBaseline	userId	movieId	Rating	userId	movieId	Rating	userId	movieId	Rating
0	1	50	4.96746	200	1250	5	318	910	4.54979
1	1	1270	4.90385	200	913	4.73042	318	1193	4.36402
2	1	1210	4.87064	200	4973	4.64894	318	3147	4.34308
3	1	1275	4.84937	200	2248	4.62437	318	48774	4.30393
4	1	110	4.81799	200	4993	4.61164	318	1090	4.29314
SVD	**userId**	**movieId**	**Rating**	**userId**	**movieId**	**Rating**	**userId**	**movieId**	**Rating**
0	1	50	4.56032	200	1250	4.30688	318	1193	4.05400
1	1	1089	4.53399	200	1197	4.29700	318	57669	4.04599
2	1	1617	4.50060	200	4973	4.29668	318	1089	4.03566
3	1	110	4.49420	200	4993	4.29547	318	4011	4.01495
4	1	1210	4.44916	200	296	4.26947	318	58559	4.01452

(3) After the dimension reduction, there are 385 users and 992 movies, when creating a full matrix for SVD to fully understand the latent features, there should be around 381920 ratings. But there are only 55556 ratings (14.54%) in the ratings file. Therefore, the dataset is sparse. Since SVD retrieves latent features and latent features will be less compromised by sparse dataset, it would be better to apply SVD for this sparse dataset.

4.3 Hybrid Recommendation Algorithm -- Collaborative Filtering + Content-based Filtering

userID	userID_similar	similarity	recommend results
1	348	0.4138376737150969	4226,34405,52247
10	186	0.12743299694043467	-1(change to CB)
12	464	0.7535433834675822	-1(change to CB)
34	426	0.15523387124464555	-1(change to CB)
81	637	0.2379172048195114	110,247,457
150	422	0.27937724292592064	110,293,364

The above table shows the most similar users of some users and their similarity. We can see that the similarity between user 10 and 34 and their most similar users is less than 0.2, so they will adopt content-based filtering algorithm. Although the similarity between user 12 and its most similar user 464 is as high as 0.75, user 464 has not seen a movie that is different from user 12 and has a score of more than 4 points. Therefore, content-based filtering algorithm is also used for recommendation at this time.

userID	genres1, ratio	genres2, ratio
10	'Comedy', 0.512	'Drama', 0.488
12	'Comedy', 0.625	'Romance', 0.375
34	'Comedy', 0.638	'Drama', 0.362

The above table shows the remaining users who adopt the content-based filtering algorithm. The second and third columns are the top two genres in terms of genres popularity and their proportion. The algorithm will select the number of recommended movies according to the weight.

genres	movieID
Comedy	567, 583, 694, 1546, 2342
Drama	124, 418, 465, 567, 583
Romance	1757, 1925, 2620, 3559, 8294
Action	465, 1925, 4552, 5720, 25961

This table shows a list of the top movies in some genres. According to the top two genres of each user's genres popularity, the algorithm will recommend the top movies for users under these genres.

userID	recommend results movieID
1	4226, 34405, 55247
10	567, 583, 124
12	567, 583, 1757
34	567, 583, 124
81	110, 247, 457
150	110, 293, 364

This table shows the final movie recommendation results of the algorithm for these users.

We can see that content-based filtering can obtain users' genres preferences by mining feature vectors of data, and then make recommendations. If we use this kind of recommendation algorithm, we can find the unique niche preferences of users, and have a better explanation. The features or descriptions of the content can be listed to explain the recommended items, and the recommended results are more likely to be trusted by users. However, its recommendation results lack novelty, and tend to give the same recommendation. It will recommend items that are similar to those that have been evaluated by current users.

The user based collaborative filtering method uses the preferences of a group with similar interests to recommend movies that users are interested in. This model can help users find new interests. However, the recommended items can only be those evaluated by the nearest users, with great limitations. If there are no very similar users or have seen the favorite movies of neighboring users, you cannot make recommendations.

In order to improve the surprise degree of the recommendation list and avoid the unavailability of nearby users, we can combine the user based collaborative filtering with the content-based filtering algorithm.

5 Conclusions

Movie recommendation is a typical personalized recommendation scenario, and there are many related methods. In this report, we have analyzed the performance of 3 recommendation methods on MovieLens datasets. For collaborative filtering, when the dataset is sparse, it's better to choose SVD. While on other cases, RMSE can be used to measure performance and choose CF algorithms. The hybrid recommendation algorithm can better avoid the problems of current mainstream recommendation algorithms. When merging various recommendation algorithms, it should be noted that different recommendation algorithms can cover different scenarios.

References

[1] "Building and Testing Recommender Systems With Surprise, Step-By-Step". https://towardsdatascience.com/building-and-testing-recommender-systems-with-surprise-step-by-step-d4ba702ef80b

[2] "Various Implementations of Collaborative Filtering". https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

[3] Boström, P. and Filipsson, M., 2017. Comparison of user based and item based collaborative filtering recommendation services.

[4] MARIA, E. (2021). Popularity based Recommendation | Movielens. Kaggle. https://www.kaggle.com/code/esratmaria/popularity-based-recommendation-movielens/notebook