Hybrid Movie Recommendation System

Guanyu Chen, Zhuowei Cheng, Jingyi Xiao, Jing Xu

Introduction

One prominent application of the recommendation system is movie recommender. There are several primary types of movie recommender system. A popularity-based recommendation system offers recommendation based on overall user ratings and suggests movies with high ratings to users. Content based recommender suggests similar movies to users according to a given movie. Similarity is often measured based on movie meta data such as genres, actors, directors, description and so on. Another type, collaborative filtering provides recommendation based on the match between users' interest. These recommendation systems have their own strength and weakness.

Here, a hybrid system which contain the three of the mentioned recommenders were proposed to overcome drawbacks of a single recommendation system.

Data and Exploratory Data Analysis

The data is the TMDB datasets which contain around 9000 movies and TV series. The TMDB datasets includes movie metadata (such as title, overview, genres, release date, revenues and so on), credits (including cast and crew information) and user ratings.

vote_count and *vote_average* are two of the variables that represent uses behavior in this dataset. Histograms of these two variables show that *vote_count* has a zero inflated poisson distribution and vote_average has a normal distribution. This shows that a large amount of movies didn't receive much votes. But that doesn't necessarily mean the votes they receive will be a low rating.

Figure 1. Histograms of *vote_count* and *vote_average*

The dataset also has descriptive information like *overview* and the *tagline* of the movie. Using keywords from those descriptive information can help us to find movies with similar contents. The bigger the word is in the figure, the higher frequency it shows up in *overview* or *tagline*.

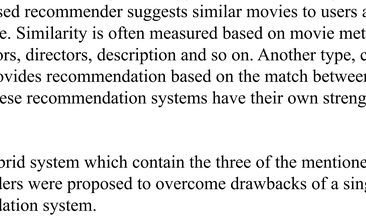
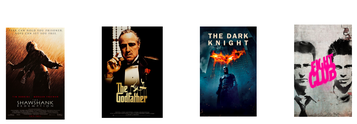


Figure 2. Wordcloud representation of keyword frequency

Another characteristic of the rating part of the dataset is the sparsity. Users in general rate only a limited number of movies. So the rating matrix is sparse.



User A	5	?	?	3
User B	?	?	3	4
User C	4	3	1	?
User D	?	?	3	?

Figure 3. The sparsity of the rating matrix

Methodology

A hybrid recommendation system that incorporates popularity, content-based and collaborative filtering is built for movie recommendation. The methodological framework is shown in Figure 4.

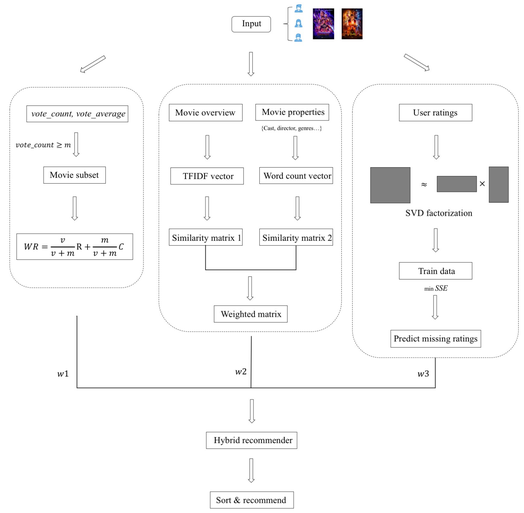


Figure 4. Methodological framework

Popularity based recommender

Popularity based recommendation system is an engine that simply recommends movies to every user based on popularity. That is, suggesting movies with high average ratings to users (Figure ?).

Instead of ranking movies only with their own ratings, a weighted rating of a movie, *WR* is formulated as the equation in Figure 5.

Figure 5. Popularity based recommender

First, a movie will only be a candidate if its rating count is above a threshold (i.e. m). Because a movie with a high average rating might not be an good option if it has few ratings, for example 10 ratings. Second, a weighted rating for each candidate movie that considers a movie's average rating (i.e R), vote count (i.e. v) and all movies' average ratings (i.e. C) is computed. Last, movies can be sorted and recommended based on their weighted ratings.

Content-based recommender

Content based recommender system is to recommend movies similar to what users have already watched.

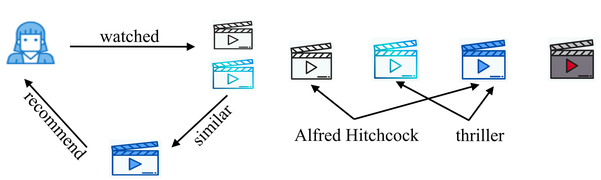
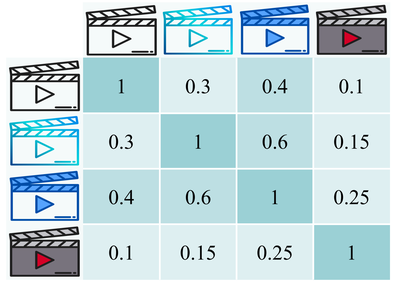


Figure 6. Illustration of content-based recommender system

Movies similarity can be defined in different ways. Similar plots, similar cast, same genre, same director, same production company and etc. can be used to measure similarity between movies.

Each movie can be represented by a vector of property. Cosine similarity can be computed for every two movies. An example of similarity matrix is shown below.



	1	0.3	0.4	0.1
	0.3	1	0.6	0.15
	0.4	0.6	1	0.25
	0.1	0.15	0.25	1

Figure 7. Illustration of cosine similarity matrix

Collaborative filtering recommender

- Create rating matrix with users' preferences
- By applying the algorithm, predict all missing values
- Recommend movies by sorting the estimated rating scores

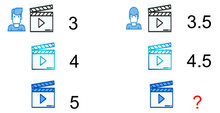


Figure 8. User based collaborative filter

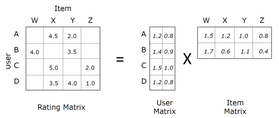


Figure 9. Funk SVD factorization

The latent factor model

1. Matrix factorization: the latent factor models are mostly based on the matrix factorization. In the example, user matrix (denote as p_u) has two column latent factors for each user; each column of item matrix (denote as q_i)has two latent factors. The dot product will produce the estimation of the rating matrix.

2. SGD Algorithm

For each user and movie, define the rating:

$$\hat{r_{ui}} = b_{ui} + p_u^T q_i = \bar{r} + b_u + b_i + p_u^T q_i$$

where $\bar{r}$ is the overall rating score, b is bias

Loss function :

$$L = \sum_{u,i}(r_{ui} - \hat{r}_{ui})^2 + \lambda_{b_u}\sum_u \|b_u\|^2 + \lambda_{b_i}\sum_i \|b_i\|^2 + \lambda_{x_u}\sum_u \|\mathbf{x}_u\|^2 + \lambda_{y_i}\sum_i \|\mathbf{y}_i\|^2$$

Learn all parameters by minimizing the loss function using stochastic gradient descent.

$$b_u^{t+1} = b_u^t + \eta(e_{ui} - \lambda_{b_u} b_u)$$
$$b_i^{t+1} = b_i^t + \eta(e_{ui} - \lambda_{b_i} b_i)$$
$$\mathbf{x}_u^{t+1} = \mathbf{x}_u^t + \eta(e_{ui}\mathbf{y}_i - \lambda_{x_u}\mathbf{x}_u)$$
$$\mathbf{y}_i^{t+1} = \mathbf{y}_i^t + \eta(e_{ui}\mathbf{x}_u - \lambda_{y_i}\mathbf{y}_i)$$

3. Process

In the project, the dataset is separated into two parts: training set and testing set. The training set is used in the SGD algorithm and calculate the RMSE in the testing set. The RMSE is defined as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \hat{r}_i)^2}$$

With smaller RMSE, the prediction has better performance.

4. Result

Training RMSE: 0.6168

Testing RMSE: 0.9111

Hybrid recommender

Considering the fact that each recommender has its own strengths and weakness, a hybrid system that incorporates popularity based, content-based and collaborative filtering recommenders is built. With an input user information containing the user and his/her movie ratings, the system will compute three scores using these three recommenders and then calculate a weighted score based on given weights. Then, movies are recommended based on the weighted score.

Results

To test our system, a few hypothetic users are used.
User1 has a viewing history of *Toy Story*, *Grumpier Old Men* and *Waiting to Exhale* and user1 wants recommendation from just popularity.

	title	release_date	popularity_based	content_based	collaborative	score
40	The Shawshank Redemption	1994-09-23	1.000000	0.229114	1.000000	1.000000
690	The Godfather	1972-03-14	0.999602	0.139165	0.959630	0.999602
5766	The Dark Knight	2008-07-16	0.963637	0.097911	0.859295	0.963637

The results from our recommender recommend the overall most popular movies to use1.

User2 has a viewing history of *Toy Story* and user2 wants recommendation from similar movies.

	title	release_date	popularity_based	content_based	collaborative	score
7714	Toy Story of Terror!	2013-10-15	0.766340	0.260106	0.586172	0.260106
6427	Toy Story 3	2010-06-16	0.834567	0.257894	0.842690	0.257894
1766	Toy Story 2	1999-10-30	0.779526	0.234008	0.722859	0.234008
2469	Creature Comforts	1989-07-15	0.706139	0.197907	0.789798	0.197907
7743	Day & Night	2010-06-17	0.818601	0.176915	0.645539	0.176915

Our recommender recommends the most similar movies based on genres, keywords, overview, etc. The results show the movies from the same series and other cartoon movies.

User3 has a viewing history of *The Endless Summer, East of Eden, Rio Bravo, Bridges of Madison County,* and *Dead Man Walking,* and user3 gave the rating of 4, 3, 4, 3.5 and 2 for each one of the movies. User3 wants recommendation from the hybrid recommender with weights of 0.33, 0.33 and 0.34.

	title	release_date	popularity_based	content_based	collaborative	score
754	Bringing Up Baby	1938-02-18	0.813181	0.961417	0.756855	0.842948
40	The Shawshank Redemption	1994-09-23	1.000000	0.229114	0.887604	0.707393
690	The Godfather	1972-03-14	0.999602	0.139165	0.954873	0.700450

In this case, our recommender combined results from the three parts and recommended the movies with the highest score.

Conclusion

Emerging various data collection approaches expose people to massive information daily. The recommendation system that utilizes the power of machine was proposed to select most possibly important information targeted for specific.

This project is interested in constructing an informative movie recommender system to provide valuable movie suggestion tailored to specific users but also considering movie contents and popularity. We first pro-processed the data. After the data is cleaned, we first built a popularity based, a content-based and a collaborative recommendation system and then we built a hybrid system combining the above three systems with customized weights. Our results show that our recommender can combine the results from the three parts of the recommender and recommend movies based on users' preference.

Reference

[1] *The Movies Dataset*. (2017). Retrieved from https://www.kaggle.com/rounakbanik/the-movies-dataset.
[2] Daniel, F. (July 2017). *Film recommendation engine*. Retrieved from https://www.kaggle.com/fabiendaniel/film-recommendation-engine.
[3] Ahmed, I. (February 2019). *The Age of Recommender Systems*. Retrieved from https://www.kaggle.com/ibtesama/getting-started-with-a-movie-recommendation-system.
[4] Banik, R. (June 2017). *Movies Recommender System*. Retrieved from https://www.kaggle.com/rounakbanik/movie-recommender-systems
[5] Ghosh, S. (Mar 2018). *Simple Matrix Factorization example on the Movielens dataset using Pyspark*. Retrieved from https://medium.com/@connectwithghosh/simple-matrix-factorization-example-on-the-movielens-dataset-using-pyspark-9b7e3f567536