Movie Recommendation System

Chu-Yun Hsiao, Chia-Yen Ho, Shu-Yun Liu, Yu-Chun Peng

2022.01.31

Executive Summary

As movie streaming platforms being prosperous for the past decade, the recommendation engine behind each platform has become even more important in order to retain users by providing more precise movie recommendations. The recommendation engine can be broadly applied in various aspects of our everyday life, such as media, E-commerce, retail etc.

As passionate data scientists who love watching movies in our leisure time, we decided to explore how these streaming platforms recommend movies based on our movie preferences. We created four movie recommendation systems, including demographic filtering, content-based filtering, collaborative filtering, and hybrid engine. Each of which provides movie recommendations based on user preferences or movie features. The dataset we used is "TMDB 5000 Movie Dataset" and "The Movies Dataset" from Kaggle.

1. Data Description

Before diving into the process of modeling, here is some basic information about the data to provide a better understanding. Among all the variables in the original data file, 11 variables were used. The table below shows the name, data type and description of each feature. Four features (cast, crew, keywords and genres) are in the JSON format.

Features	***Data Type***	***Description***
title	Categorical	The official title of the movie
overview	Categorical	A brief blurb of the movie
tagline	Categorical	The tagline of the movie
popularity	Numeric	The Popularity Score assigned by TMDB
cast	Categorical	The JSON format data that list out casts of the movie
crew	Categorical	The JSON format data that list out crews of the movie
keywords	Categorical	The JSON format data that contain the movie plot keywords for movies

genres	Categorical	The JSON format data that list out all the genres associated with the movie
vote_average	Numeric	The average rating of the movie
vote_count	Numeric	The number of votes by users
ratings	Numeric	The rating of a movie from users

2. Preparing Data

• Reformat Data Type

Since the form of features like cast, crew, genres and keywords are in the JSON format, we extracted the director and three most important actors, genres, and keywords associated with that movie.

• Text Pre-processing

We removed stop words, aligned words in lowercase, and conducted lemmatization to clean the textual data. In order to evaluate how relevant a word is, we converted the textual data into a TF-IDF matrix, which will give us a matrix where each column represents a word in the overview vocabulary and each row represents a movie. This helps group together the inflected forms of a word so they can be analyzed as a single item.

• Calculate Cosine Similarity

We used the cosine similarity to calculate the angle between two data points in the high dimensional metric space that denotes the similarity between two movies. Mathematically, it is defined as follows:

$$\text{Cosine}(x, y) = (x*y)/(|(|x|)|*|(|y|)|)$$

Since we used the TF-IDF Vectorizer, calculating the dot product will directly give the cosine similarity score. Therefore, we used scikit-learn's linear_kernel instead of cosine_similarities since it can significantly improve computation time. Eventually, we got the pairwise cosine similarity for all the movies in our dataset.

3. Types of Recommender Systems

• Demographic Filtering

Demographic filtering is to make recommendations for users based on movie popularity.

The basic idea behind this recommendation is that movies that are more popular and more critically acclaimed will have a higher probability of being liked by the average audience. This model does not give personalized recommendations based on the user.

This model is simple. All we did is to sort all movies based on popularity and ratings, then showed the top movies on the list. However, if we directly used average ratings of the movie as the score, it would not be accurate because some ratings did not have many votes. Thus, we used IMDB's weighted rating:

where, Weighted Rating = $(\frac{v}{v+m} \times R) + (\frac{m}{v+m} \times C)$

- v is the number of votes for the movie
- m is the minimum votes required to be listed in the chart
- R is the average rating of the movie
- C is the mean vote across the whole report

Furthermore, we set a cutoff point to make sure all the movies listed in the chart have more votes than at least 90% of the movies in the list. To be more specific about how the demographic filtering recommendation works, here's an example. For all users, the recommendation will give the same list of movie recommendations. From our dataset, the first movie recommended will be "The Shawshank Redemption", and the second one will be "Fight Club". This recommendation applies to all users.

- ## Content-Based Filtering

Content-based filtering makes recommendations by finding items with similar attributes. For instance, if a user likes item A, and we calculate that item A and item B are similar, we believe the user will like item B. Content-based filtering can be further divided into two types:

1. Movie Description

In this section, we computed similarity scores for all movies based on each movie's descriptions and tagline, and then recommended movies based on that similarity score. The way we computed similarity scores is to divide the number of times a word occurs in a description by measuring what proportion of all the descriptions a word occurs in. This has the effect of reducing the value of common words while increasing the weight of words that do not occur in many descriptions. For example, if you were comparing the description of one movie against the description of all the movies in our data, the term "hero" might get a low score as it probably occurs a lot in many movie descriptions. The term "Iron Man" on the other hand would get a high score as it is not as common in other movies' descriptions. Then we computed cosine similarity for each movie and returned the top 10 movies with the highest similarity score as a recommendation. Therefore, if we input "Iron Man" in our recommender, it will return the top 10 movies that are most related to "Iron Man" based on its movie description.

2. Genre, keywords, actors, and director

The logic of this recommender is very similar to what we created in the previous section, the only difference is that the features we use for the first part are based on each movie's description and tagline, while the features we use for this part are based on each movie's director and top three actors, keywords, and genres.

- **Collaborative Filtering**

Collaborative filtering is to make recommendations for items that might interest a user based on the preferences of similar users. To be more specific, users that are similar to user A can be used to predict how much user A will like a certain item that users have used but user A has not. This could provide more personalization compared to content-based filtering. Collaborative filtering can be further divided into two types:

User-Based Filtering

User-based recommendations compare among users. We could use user-based data to find similar users based on how they rated different movies. For example, Robin and Evelyn have a similar interest in movies. A new movie has been launched into the market, and Robin has seen and loved it. Therefore, it is highly likely that Evelyn will like it too. As a result, the system recommends this movie to Evelyn. There are some limitations in our project, as our steps of analysis pause after finding the rating of a new movie based on similar users. Further analysis would be needed to find out which top-rated movies to recommend the user. The advantages of this type are that it can provide more interesting patterns and insight, but its performance metric is usually lower than item-based recommendations.

Item-Based Filtering

Item-based recommendations compare different items. We could use item-based data to find similar movies based on how they have been rated by the users. For example, if Robin, Susan, and Evelyn have given 5 stars to "Harry Potter" and "The Hobbit", the system identifies the items as similar. Therefore, if someone watches "Harry Potter", the system also recommends "The Hobbit" to him or her. The pros of this type are that it is more consistent over time and can be pre-calculated, while the cons are that it provides more obvious suggestions, thus cannot provide strong insights.

1. KNN

We tried user-based filtering using KNN regression model. We can predict a user's potential preference for movies based on users who are similar to her. For example, we found that Robin, Evelyn, Vivian, and Susan have similar preferences in movies. If we

wanted to know Robin's potential favor toward the movie "Mockingbird", we can calculate the estimated rating based on the ratings from the other 3 people.

2. SVD

There are disadvantages to using KNN because there is a sparsity issue. That is, not all users rate all movies. Thus, we sought other methods to solve this problem. SVD (Single Value Decomposition) is a popular matrix factorization algorithm, which is a solution for this.
We used the Surprise library, which is based on SVD to minimize RMSE to provide recommendations. By applying the package, we could predict what rating would a user give for a certain movie. For example, we can predict that Robin will give "Don't Look Up" a rating of 4.5. The model itself doesn't directly recommend movies, but we can use this rating prediction to understand what movies a user will likely like, justifying the recommendations to him/her. After training the model, we got **RMSE (Root Mean Square Error) =0.896**, which is an outstanding result.

- ## Hybrid Engine

We brought together ideas from content-based filtering and collaborative filtering to build the hybrid recommendation engine. So, we can provide both the recommended movies and the estimated ratings to the user.
There are two major processes. First, we gave movie recommendations to a particular user using content-based filtering, in which we built the similarity metrics based on description or keywords, genres, actors, and directors. Second, we predicted the user's ratings for those movies using the SVD method mentioned above.

4. Conclusion

We built several recommendation models using demographic filtering, content-based filtering, collaborative filtering, and hybrid engine. Demographic filtering serves as a foundation as it offers generalized recommendations. It is simple but cannot be used to generate powerful insight. Content-based filtering and collaborative filtering are more advanced methods. Content-based filtering enables us to find similar movies based on content, while collaborative filtering enables us to estimate ratings for a given user and movie. The RMSE that we obtained from the model was 0.896. Finally, the hybrid engine brings the content-based filtering and collaborative filtering method together, integrating the advantages and shortcomings of the two methods as they are complementary. In conclusion, each method has its strengths and weaknesses, the best advice would be to use these recommendation models according to the context of the business problem that we are trying to solve.

This project is inspired by https://www.kaggle.com/ibtesama/getting-started-with-a-movie-recommendation-system