DSO 593 Independent Research Report

Movie Recommend Engine -- Collaborative Filtering

Yufei Wang MSBA

* Project Introduction

The rapid growth of data collection has led to a new era of information. Data is being used to create more efficient systems and this is where Recommendation Systems come into play. Recommendation Systems are a type of **information filtering systems** as they improve the quality of search results and provides items that are more relevant to the search item or are related to the search history of the user.

They are used to predict the **rating** or **preference** that a user would give to an item. Almost every major tech company has applied them in some form or the other: Amazon uses it to suggest products to customers, YouTube uses it to decide which video to play next on auto-play, and Facebook uses it to recommend pages to like and people to follow. Moreover, companies like Netflix and Spotify depend highly on the effectiveness of their recommendation engines for their business and success.

For this project, I implemented a movie recommend engine based on collaborative filtering. Our data is user rating data from [The Movies Dataset](https://www.kaggle.com/rounakbanik/the-movies-dataset) which contains the TMDB and IMDB IDs of a small subset of 9,000 movies of the Full Dataset. This movie recommend engine can offer recommendations to one user on specific number of movies or provide 3 movies for each user.

* Process
* Data Visualization



This graph shows the most popular 30 movies based on number of viewers. It is clear that “Terminator 3: Rise of the Machines” is the Top 1 from all 10000 movies. However, when it comes to specific user, everyone has their preference. The following picture shows watch history for user 24 and 27, from which you can see that they only have 1 common item.



* Data Quality

|  |  |
| --- | --- |
| Variables | Description |
| UserId | There are 671 users in this rating table which ranges from 1 to 671. |
| MovieId | There are 9066 movies in this rating table which ranges from 1 to 163949. |
| Rating | Users’ rating ranges from 0.5 to 5. |

From following chat, we can see that most users would like to give a rate varying from 3.0~4.0 for a general film and a rate from 5.0~5.5 for an excellent movie.



* Methods of Recommendation

Basically, there are 3 types of recommendation.

**Demographic Filtering**: They offer generalized recommendations to every user, based on movie popularity and/or genre. The System recommends the same movies to users with similar demographic features. Since each user is different, this approach is considered to be too simple. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience.

**Content Based Filtering**: They suggest similar items based on a particular item. This system uses item metadata, such as genre, director, description, actors, etc. for movies, to make these recommendations. The general idea behind these recommender systems is that if a person liked a particular item, he or she will also like an item that is similar to it.

**Collaborative Filtering**: This system matches persons with similar interests and provides recommendations based on this matching. Collaborative filters do not require item metadata like its content-based counterparts.

I built this film recommender based on collaborative filtering. Steps as following:

1. Locality-sensitive Hashing

Apply minhash to obtain a signature of 20 values for each user by permuting the rows of characteristic matrix of movie-user matrix. LSH is to help speed up the process of finding similar users, where the signature is divided into 5 bands, with 4 values in each band.

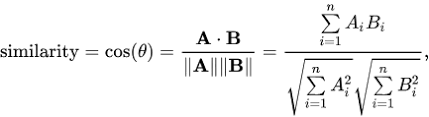
1. Normalizing Rating

Although some people have watched the same movie, they may have opposite comments which means they have different preferences. This information can be collected from their ratings and I use this following formula to normalize them. Then the positive rating means viewer likes it, while the negative rating means viewer doesn’t like it.

New rating = old rating – avg. rating by user

1. Cosine Similarity

I use cosine to calculate the similarity of two vectors (watch histories) that higher cosine means a higher similarity.



1. Making Recommendations

Based on the LSH result, for every user, I sum the similarity of each movie, finally return 5 recommendations with highest similarity.

* Recommendation Example

For user 2, there are 22 users who have a high similarity (>0.5) to her/him and 2109 movie recommendations from their watch history. The top 3 films are No.466(0.34), No.1022(0.21) and No.1020(0.21).

* Project Summary

Although this recommender can offer advice on next watch, the performance is not good enough. For the next step, content based filtering should be included as it always shows a good accuracy.