Movie Recommendation System

CISC5950 Big Data Programming

Final Project

Team Members

Guo Tian

Meilan Jin

Jiayu Zhou

Xiaoyang Ren

Han Wu

Abstract

Recommendation system has been widely used in all aspects of our lives. Yet, currently, they are far from optimal. We want to understand recommendation systems and compare their performance on the prediction of recommendations. In this project, the datasets we examined are extracted from Netflix and The Movie Database (TMDB) website. We attempt to build a movie recommendation system which could provide personalized recommendations based on user profile and item profile. This was achieved by the following process:

- Use data preprocessing techniques to improve the accuracy of prediction on the datasets.
- Use Collaborative Filtering method to build movie recommendation systems based on customer habits.
- Run K-Means algorithms to analyze the similar between each movie on TMDB 5000 dataset.
- Build personalized customers recommendation engine by combining different solution such as User-Based Recommendation System and Item-Based Recommendation System

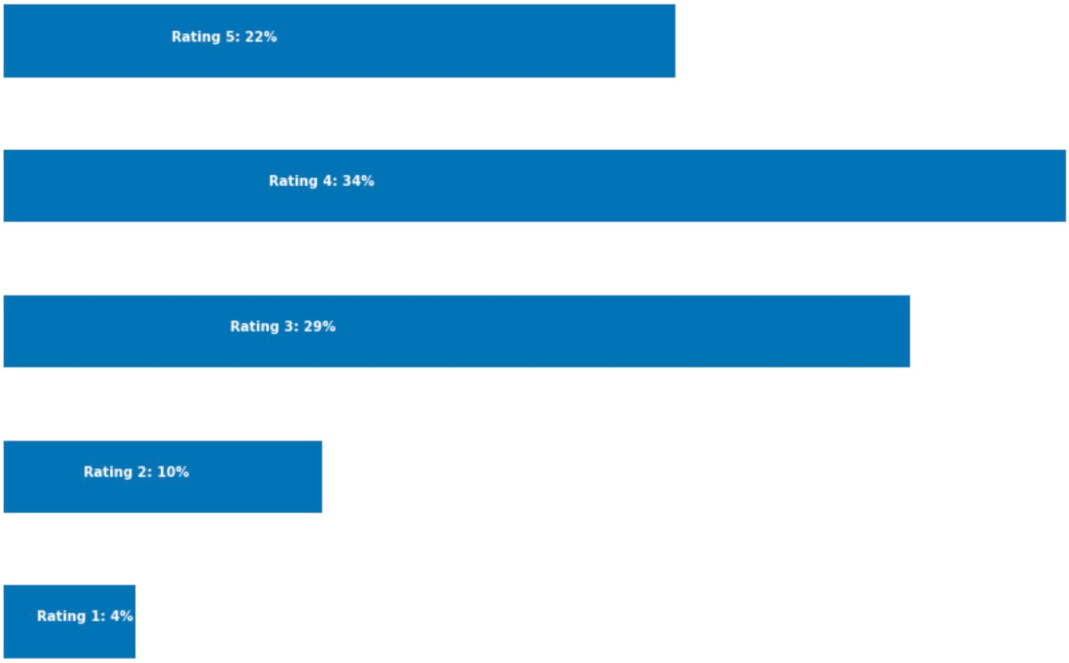
Data Preprocessing and Transformation

In this project, the datasets we used are collected from Netflix Prize data and The Movie Database 5000 dataset. We have two parts of datasets. The Netflix datasets contains ratings in a scale of 1 to 5, and time of given rating for more than 1,7000 movies. The TMDB 5000 dataset contains a lot of features such as actor name, director name, keywords, language, country, budget, title year, movie facebook likes, and etc. Data preprocessing process was achieved by the following processing:

- Extract netflix prize data files and then combine together
- Drop date feature, and then add a new feature of movie id to each rating observation

- Filtering movies that have movie features in TMDB data
- Give movies feature data ID that match with those in netflix prize data

Total pool: 2,237 Movies, 478,278 customers, 58,522,356 ratings given



- Remove movie with too less reviews (they are relatively not popular)
- Remove customer who give too less reviews (they are relatively less active)

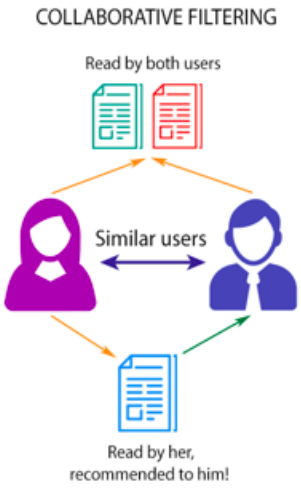
```
Original Shape: (58522356, 3)
After Trim Shape: (27017553, 3)
-Data Examples-
          Customer ID  Rating  Movie_Id
3346          2473170       5        30
11266748      2648181       5      3864
21774447      1404690       1      6844
32552780       956035       4     10607
43715749       176611       3     13728
54439292      1614125       5     16668
```

User-Based Movie Recommendation System

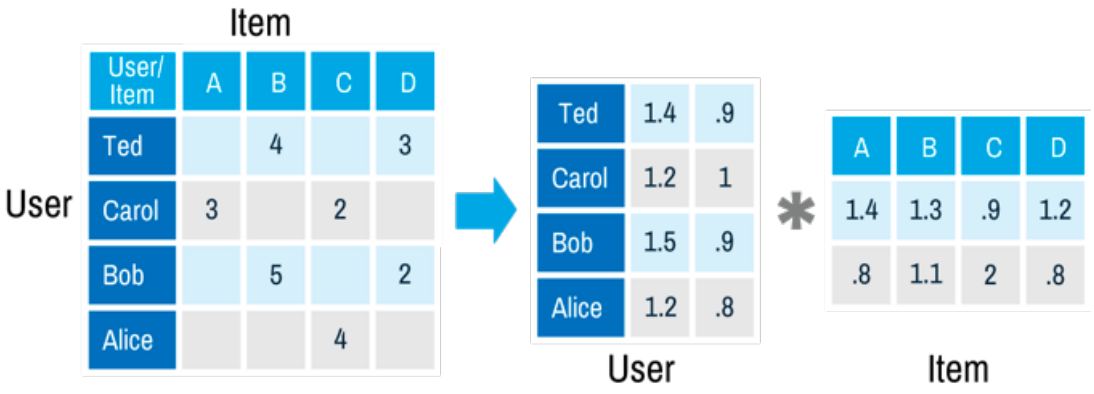
1. Collaborative Filtering

In this project, we mainly use Collaborative Filtering method to predict user's

movie rating. Collaborative Filtering techniques make recommendations for a user based on ratings and users' preferences. The main idea behind collaborative filtering is based on the similarity of users. If two users both like one movie, then the movie that one user likes but other users have not watched can be recommended to him.



When making movie recommendations, we create movie ratings matrices. Each column represents a user and each row represents a particular movie. However, there are many missing values in the entries since not all users have rated all movies. The main underlying idea of collaborative filtering is to fill the movie rating matrix by factoring it as the product of two subset matrices. one matrix describes properties of each user, the other describes properties of each movie. We use two algorithms for Collaborative filtering, SVD and ALS, to approximating these movie rating matrices.



2. SVD Method

SVD is a common matrix factorization method that is usually used to reduce the number of features of a dataset by reducing space dimensions from N to K where K < N. The main underlying idea of SVD is that it decomposes the rating matrix A into a product of 3 subset matrices(USV):

A = USV

U：each row represents one character of a user

V：each column represents one character of a movie

S：represent the correlation of user and movie

The result of the decomposition leaves us with an ordered matrix of singular values which encompass the variance associated with every direction. We assume larger variances means less redundancy and less correlation and encode more structure about the data. This allows us to use a representative subset of user rating directions or principal components to recommend movies.

Our user-based collaborative filtering has respectively RMSE and MAE when running 3-folds cross validation as below:

```
Evaluating RMSE, MAE of algorithm SVD.

------------
Fold 1
RMSE: 0.8098
MAE:  0.6316
------------
Fold 2
RMSE: 0.8100
MAE:  0.6315
------------
Fold 3
RMSE: 0.8098
MAE:  0.6314
------------
------------
Mean RMSE: 0.8099
Mean MAE : 0.6315
------------
```

3. ALS Method

ALS is also a matrix factorization algorithm that uses Alternating Least Squares with Weighted-Lambda-Regularization (ALS-WR). It factors the movie rating matrix A into the user-to-feature matrix U and the item-to-feature matrix M. The Alternating Least Squares algorithm does this by first randomly filling the user's matrix with values and then optimizing the value of the movies with the error that is minimized. Then, it holds the movies matrix constant and optimizes the value of the user's matrix. Given a fixed set of user factors, we use the known ratings to find the best values for the movie factors using the optimization. Then we "alternate" and pick the best user factors given fixed movie factors.

User-based recommendation system prediction example:

Movies prediction for user ID 262478

Year	Name	Estimate_Score
2004.0	Six Feet Under: Season 4	4.439262
2002.0	Gilmore Girls: Season 3	4.284325
1989.0	The Simpsons: Season 1	4.258923
2001.0	Alias: Season 1	4.249497
2001.0	Wallace & Gromit in Three Amazing Adventures	4.223088
1954.0	Seven Samurai	4.211227
2003.0	Stargate SG-1: Season 7	4.205133
1959.0	North by Northwest	4.204525
1978.0	MASH: Season 7	4.203603
1993.0	Star Trek: The Next Generation: Season 7	4.192976

By observing the model performance of the SVD and ALS, we find that our RMSE and MAE are all around 0.8 and 0.6, which indicate that there are small variances between predict value and users' true preference. From the example result of our movie recommendation, we could get the conclusion that user 262478 prefers TV drama with long series other than films.

K-Means Clustering

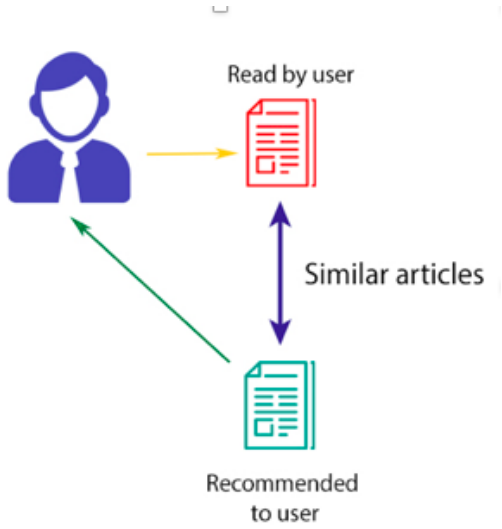
We also use K-Means Method to predict which cluster the movie belongs to, based on user ratings of movies. Then recommendation system will then suggest the highest rated movies from that cluster.

- Use K-Means to get central points and give label to each movie.
- Based on the user's rating of the movie, do majority vote.
- Find most of his favorite movies belong to which cluster and give labels to new movies.
- Recommend them to the users whose most of favorite movies are in the same cluster.

But User-based recommendation system commonly occurs with two problems. One is early rater problem. It occurs when a new user is introduced to the system and has yet to rate a selection of items significantly large enough for the service to start suggesting similar items. The other problem is sparsity problem. It occurs when there is insufficient information to work with in order to provide the user base with decent approximations as to which products they would likely prefer. So, we also build an item-based movie recommendation system.

Item-Based Movie Recommendation System

Item-Based Movie recommendation takes into account the likes and dislikes of the user and generates a User Profile. For generating a user profile, we take into account the item profiles and their corresponding user rating. The user profile is the weighted sum of the item profiles with weights being the ratings user rated. Once the user profile is generated, we calculate the similarity of the user profile with all the items in the dataset, which is calculated using cosine similarity between the user profile and item profile.



Challenges with Collaborative Filtering

Although there are different ways to do collaborative filtering, they all share some common drawbacks. Those drawbacks will be discussed in this section.

Cold start

Probably the most known and intuitive drawback with collaborative filtering is the cold start problem. Since collaborative filtering does its predictions based on the users' history, what will happen when the system tries to predict a movie without any rating history to a new user? Some solutions to this can be either asking the user to rate several movies upon account creation and/or the use of content-based filtering until enough data is gathered to do collaborative filtering.

The long tail

The addition of new items gives rise to another problem; new items have no ratings and therefore cannot be recommended to anyone. This creates the long tail problem with the rich-get-richer effect. Movies with few ratings will keep being ignored while movies with many ratings will keep getting more ratings. The problem is that the long tail of unpopular movies might contain preferable movies to certain users, but they stay unpopular because they never get recommended and not because it is a bad movie.

Indie Films Lover Recommendation System

To deal with this problem, we provide a solution to recommend movies to those users who love independent movies. To find out whether they are indie movie lovers, we filter the movies they give ratings of 5 and evaluate these movies' review times and measure the scale of whole dataset. After determining one user is indie films lover, we can calculate the Euclidean distance between the user's top rating movies and the whole movie dataset, then we can get 10 movies most similar to the movies that the user gives 5 rating.

Indie films lover recommendation system result example:

movie_id	movie_title
6274.0	The Hunt for Red October
16552.0	GoldenEye
1148.0	For Your Eyes Only
7910.0	The Living Daylights
NaN	Tomorrow Never Dies
12317.0	The Rock
14439.0	Live and Let Die
16952.0	Licence to Kill
11613.0	The Man with the Golden Gun
12160.0	Cliffhanger

New Customers Recommendation System (achieved by SparkSQL)

When there are some people becoming new users, we could not get any useful information of them, therefore our models are not able to offer some recommendation. In order to solve the problem, we offer a selection of what

kinds of movies the new users are interested in when they are registering. And then, we use SparkSQL to select movie title, the movie score coming from previous users and the kind of movies they like to generate top 10 movies. For example, a new user selects adventure movies, the system will provide 10 adventure movies to him including the score and review.

New customers recommendation system result example

movie_title	imdb_score	num_voted_users
Seven Samurai	8.7	229012
Spirited Away	8.6	417971
Back to the Future	8.5	732212
Raiders of the Lo...	8.5	661017
Das Boot	8.4	168203
Lawrence of Arabia	8.4	192775
Princess Mononoke	8.4	221552
2001: A Space Ody...	8.3	427357
Batman Begins	8.3	980946
Indiana Jones and...	8.3	515306

Discussion

This movie recommendation system is dedicated to providing customized service for each individual since each person has particular preference for movies. It gives out solutions for new movies, new customers, and indie films lover recommendation system, since collaborative filtering has certain drawbacks. However, there is still more aspects need improving. For example, it's hard to find a standard way to evaluate how well the model performs. Movie recommendation is very subjective for each individual. Besides, it needs more detailed data of users if the system tends to give recommendation for a group of people. If we want to improve the accuracy of this recommendation system, we could also include the leading actor information to each movie, since people tend to choose movie based on the cast they fond of.