Customized Movie Recommender System

SI 650 Information Retrieval Final Report

Qi Zhang
University of Michigan
Ann Arbor, Michigan, United States
angieqiq@umichi.edu

Yuying Li
University of Michigan
Ann Arbor, Michigan, United States
yuyingli@umichi.edu

Xuanyu Wang
University of Michigan
Ann Arbor, Michigan, United States
olivwang@umichi.edu

ABSTRACT

In this report, we describe the approaches to a customized movie recommendation. We use content filtering and collaborative filtering to implement the recommendation mechanism. In terms of content-based filtering, we choose TfidfVectorizer to extract TF-IDF feature from overview of movies, and we choose CountVectorizer for a collection of director, cast, keyword and genre. We use SVD model for collaborative-based filtering. Also, we also use Flask as the web frame, together with html to implement the front end. User can type in their ID in the first search box, or type a keyword in the second search box and select a radio to choose whether you want to search based on overview or other dimension such as director, and the system will return top eight recommended movies based on your profile, or your keyword type. With Root-mean-square error(RMSE) method, our recommendation system can reach good evaluation result of 0.9.

KEYWORDS

TF-IDF, Vecterizer, SVD, RMSE, collaborative filtering

ACM Reference Format:
Qi Zhang, Yuying Li, and Xuanyu Wang. 1997. Customized Movie Recommender System: SI 650 Information Retrieval Final Report. In *Proceedings of ACM Woodstock conference (WOODSTOCK'18).* ACM, New York, NY, USA, Article 4, 3 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

1.1 Problem

There are too many movies for people to choose from. Such system adds value to everyone's daily life due to its ability to provide enhanced entertainment, because it can suggest a customized set of movies to users based on their interests, rating history, or the popularity of the movies. Besides, movie is now a common entertainment, and there is a large market demand where people wants to efficiently find a movie matching their tastes and query needs. Therefore it is important to have accurate and customized movie recommendation systems to match customers' taste and better people's lives.

WOODSTOCK'18, December 2018, Ann arbor, Michigan USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

1.2 DataSet

We use the dataset called *The Movie Dataset* posted on Kaggle. The dataset contains metadata on over 45,000 movies and 26 million ratings from over 270,000 users. The main data attributes we used for movie are the movie ID, title, overview, director, cast, crew, genre, keyword. For user watching history data, we have three attributes to use, user ID, movie ID, and the rating this user once gave to this movie.

2 METHOD

Since there are three functions, recommender function, overview search , feature (function, cast,genre,plot) search function in our recommender system. The implement method will be split into there parts according to functions.

2.1 Recommender Function

The recommender function is based on the rating dataset, which contains information of users' rating on movies. In order to build a movie recommender system which capture the personal tastes and biases of a user, a technique called **Collaborative Filtering** will be used to make recommendations to Movie Watchers. It is basically of two types: User based filtering and Item Based Collaborative filtering[2]. The former one let systems recommend products to user that similar users have liked and the latter one help system recommend items based on their similarity with the items that target user rated. Similarity in both types will be computed with Cosine Similarity. To utilize both types of filtering technique and handle the scalability and sparsity issue created of collaborative filtering, the **single value decomposition(SVD)**, which finds latent factor model of the matrix, will be used to capture the similarity between users and items. The matrix will be separated into three matrix. And the first and the end matrix will represent the user-latent factor and item-latent. By finding similarity in this two matrix, the original matrix will be filled by the similarity of items and users. For evaluation this function, we use the criteria called RMSE to evaluate the recommender function's performance. And the *RMSE* for our function is 0.9, which means for every predicted user's rating, the mean error is plus-minus 1 creidt in total 5 credit.

$$RMSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - y_i)^2$$

2.2 Overview Search Function

For better understand customers' preference, querying certain keywords, the movie will be recommended according to the overview feature. In order to implement this function, the word needs to be

	userId	movieId	rating	predict
19090	128	1028	5.0	4.114526
99678	665	4736	1.0	3.380532
18455	120	4002	3.0	3.346646
35755	257	1274	4.0	3.509386
66536	468	6440	4.0	3.153289
58156	423	1732	5.0	3.815770
45934	330	265	5.0	3.835023
67241	471	44301	4.0	3.602277
84837	569	914	5.0	4.128679

Figure 1: The predicted rating for the test dataset

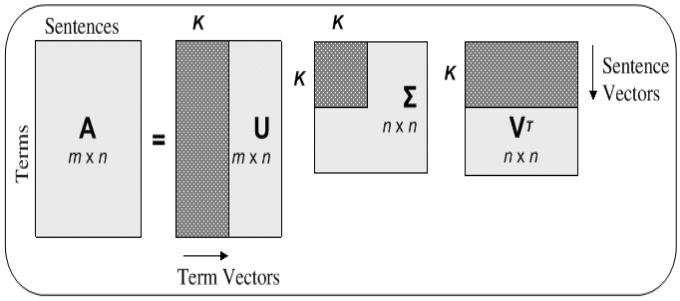

Figure 2: The-Singular-Value-Decomposition

convert by TF-IDF, then using the cosine similarity of the inner-product of the matrix to calculate the similarity rank for any movies[1].

	title	overview
0	Toy Story	Led by Woody, Andy's toys live happily in his ...
1	Jumanji	When siblings Judy and Peter discover an encha...
2	Grumpier Old Men	A family wedding reignites the ancient feud be...
3	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...
4	Father of the Bride Part II	Just when George Banks has recovered from his ...

Figure 3: The Overview

2.3 Feature Search Function

The feature search is similar to the overview search function, however, this time, the recommender will based on cast, genres and keywords, director. This function will give customers more chance to pick up their interested movie. And the way how to implement this function is the same as the overview search function. However, this time, the recommender system will recommender movies in different angles. If we adding weight for the director, it will recommender movies from the same director. For example:

3 RESULTS AND EVALUATION

Our recommendation of similar movies given a movie uses content based filtering. The essential mechanism of content based filtering is

```
get_recommendations('The Dark Knight Rises', cosine_sim2)
```

```
15651            Inception
23076          Interstellar
21804    Out of the Furnace
12589       The Dark Knight
10210         Batman Begins
26733      The Road Killers
10700                Havoc
26987    Thorne: Scaredycat
20530          Treacle Jr.
15704           Play Dirty
Name: title, dtype: object
```

Figure 4: The demo for the feature search function

to calculate cosine similarity of pairs of movies with the features we extracted to characterize each movie. There is not an objective and easy way to evaluate how good the calculated similarity scores are, therefore we decide to skip the evaluation for our movie overview based and credits, genres and keywords based movie recommender.

Our personalized recommender system uses collaborative filtering, which recommends movies based on similar users' tastes and similar movies' ratings. Basically, the system tries to predict users' ratings on movies. Since the actual ratings are present, we could calculate Root Mean Square Error (RMSE) between predicted ratings and the actual ones to evaluate effectiveness of our collaborative filtering model. The equation for RMSE is as shown below. Here, N represents total number of movie ratings being predicted, $\hat{R}$ stands for predicted rating, and is the corresponding actual rating.

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N} (\hat{r}_n - r_n)^2}{N}}$$

Figure 5: The function for RMSE

Specifically, we split our dataset into 5 folds with 4 folds being training set and 1 fold being testing set for each round of evaluation. In this way, we could get 5 RMSE scores. The RMSE scores are 0.9016, 0.8942, 0.8929, 0.9002, and 0.8972, with a mean score of 0.8972. The scores across 5 folds has little variance, which means that our model's performance is stable, and therefore the good prediction is not by chance. Also, the RMSE score less than 1 indicates that our collaborative filtering model is effective in predicting users' preference on unseen movies. Therefore, we conclude that our personalized recommender is stable and effective.

4 DISCUSSION

The main takeaway of our project is actually building a recommender system. We have gotten a good understanding of advantages and limitations of content based filtering and collaborative filtering. Content based filtering suggests similar movies based on a particular movie, but it is not capable of capturing tastes across genres. Meanwhile, collaborative filtering can provide recommendations base on users' tastes, but it suffers from scalability and data sparsity issues. Now, we have our content based filtering and collaborative filtering as two separate parts, so our next step would be combining two filtering model together. One possible approach

is to feed similar movies as a new feature into collaborative filtering model. That is, when we predicting rating of a movie, the ratings of similar movies by similar users also matter. By doing this, we could combine two filtering models to get a more powerful model.

We have also learned that it is not efficient to search in a large dataset. Therefore, one thing we could improve is to store our dataset in a database and query the database instead of searching a large dataset every time.

Also, we have learned to build user interface using python's flask library for our recommender system. We run our application on local host so we could not provide a link to our application. We will submit our front-end code along with back-end.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Qiaozhu Mei for lecture references and the help from GSI Shiyan Yan.

REFERENCES

[1] Fie Zhang et al. 2017. Improvement of Pearson similarity coefficient based on item frequency. Retrieved March 2, 2005 from https://ieeexplore.ieee.org/document/8076697

[2] Chengxiang Zhai and Sean Massung. 2016. *Text Data Management and Analysis* (1st. ed.). ACM, Champaign, Illinois.