Movie Recommendation

Junlin Lu
Rutgers University
jl2364@rutgers.edu

Shengjie Li
Rutgers University
sl1560@rutgers.edu

ABSTRACT

The goal of our project is about applying different skills and algorithms to do movie recommendation and rating prediction given a movie ratings dataset. In this project, we carefully perform data preprocessing. Then we build one content-based filtering model, two collaborative filtering models and one latent factor model to tackle this problem. The results of both movie recommendation and rating prediction are satisfying.

KEYWORDS

Movie Recommendation, Rating Prediction, Content-based Filtering, User-based Collaborative Filtering, Item-based Collaborative Filtering

1 INTRODUCTION

1.1 Background

A recommender system is a system that seeks to predict the “rating” or “preference” a user would give to an item [4]. Recommender systems are used in a variety of areas, e.g., playlist generator for music and video services, product recommendation for online shopping, content recommendation for news services.

1.2 Problem Setting

In this project, we are building a movie recommender system using a public dataset “MovieLens Latest Datasets-small” provided by MovieLens[1], which contains $100,000$ ratings and $3,600$ tag applications applied to $9,000$ movies by more than 600 users.

In this report, we will first introduce the data processing we do. Then we will list the experiments we have done. Finally, we will conclude the results and demonstrate future works.

1.3 Proposed Method

Because different approaches have their own advantages and disadvantages, We propose 4 models to tackle this problem.

1. Content-based model.
2. User-based collaborative filtering.
3. Item-based collaborative filtering.
4. Latent factor model.

Content-based filtering methods are based on a description of the item and a profile of the user’s preferences [3]. It is able to recommend unpopular items, but it’s not very effective in recommending items for new users.

Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past [2]. It doesn’t require any feature from items, but it tends to recommend popular items.

A latent factor model is a statistical model that relates a set of observable variables to a set of latent variables [5].

1.4 Results

After building the aforementioned models and doing experiments, we are able to do rating prediction and movie recommendation. Eventually, we are able to evaluate our models and get satisfying results.

2 PROBLEM FORMALIZATION

In this section, we will formalize our problem and provide a brief overview of our methodology. Our dataset contains 4 CSV files: ratings.csv, movies.csv, tags.csv and links.csv. Our goal is to recommend movies to users based on the existing rating that each user cast to the movies they watched.

We assume that movies with similar tags or in the similar genre are relevant. Therefore, for target user, we can take the movies that he watched, compute the similarities between the movies that share similar tags or similar genre with these movies, then recommend to target user.

We also assume that user behavior has some kinds of relevance, which means that users who watched the same movie have the same preference, and their similarities can be computed. Thus, for each user, if we can identify their ”similar” users, we can recommend the movies that similar users watched.

Furthermore, we assume that the set of items that target user has rated has some kinds of relevance to the sets of items that other users has rated, which means that movies have the same ratings have come common features. Thus, if we can compute the similarities between movies, we can recommend movies that are similar to the movies target user watched.

With these assumptions, we can propose different models that are built upon our assumptions and combine them to make a even better prediction if possible.

3 THE PROPOSED MODEL

3.1 Content-based Model

The main idea of content-based approach is to recommend movies to user x that are similar to previous movies rated highly by x.

3.1.1 Item profile. We choose ‘title’, ‘genres’ from `movies.csv` and ‘tag’ from `tags.csv` as the features of movies. Then we build item profile by calculting the TF-IDF score of features. We do log normalization in term frequency calculation to potentially get better performance.

$$TF_{ij} = log(f_{ij} + 1)$$

$$IDF_i = log\frac{N}{n_i}$$

$$TF\text{-}IDF_{ij} = TF_{ij} \times IDF_i$$

[1] https://grouplens.org/datasets/movielens/

Where f_{ij} is frequency of feature i in movie j, n_i is the number of movies that have feature i, N is the total number of movies.

3.1.2 User profile. We use two approaches to build user profiles.

(1) Weighted average of all rated item profiles
(2) Average of item profiles of the top 20 rated movies

3.1.3 Recommendation. For a given user, we use its user profile x to calculate cosine similarity score with all movies, and then find k movies with the highest cosine similarity scores:

$$\underset{i}{\operatorname{argmax}}\, cos(x, i) = \underset{i}{\operatorname{argmax}} \frac{x \cdot i}{\|x\| \cdot \|i\|}$$

3.1.4 Rating prediction. For rating prediction given user x and movie i, we find the most similar 10 movies to movie i that user x has watched and rated, and let the average rating of this 10 movies be the predicted rating for movie i.

3.2 User-based Collaborative Filtering

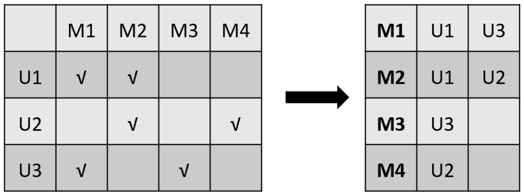
There are two main steps in user-based collaborative filtering:

(1) Find similar users set N for target user.
(2) From set N, find the items that these users like and recommend those items that target user has never seen to target user.

In this project, we use cosine similarity to calculate user similarities:

$$Sim_{uv} = \frac{|M(u) \cap M(v)|}{\sqrt{|M(u)||M(v)|}}$$

Where *M(u)* represents the movies user u watched. Note that a pair of users may not watch the same movie, a user-movie table is very sparse and consumes a lot of time to loop through it, so we use movie-user table to represent who watched the same movie, as shown in Figure 1.

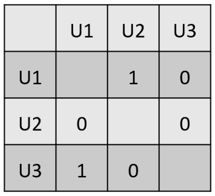


	M1	M2	M3	M4
U1	√	√		
U2		√		√
U3	√		√	

M1	U1	U3
M2	U1	U2
M3	U3	
M4	U2	

Figure 1: Table Transformation

Then we use a co-occurrence matrix to represent the number of movies that each pair of users watched, as shown in Figure 2.



	U1	U2	U3
U1		1	0
U2	0		0
U3	1	0	

Figure 2: Co-occurrence Matrix

The interest of user u for movie m can be represent as:

$$Interest(u, m) = \sum_{v \subset K} Sim_{uv}$$

Where K is the set of top k similar users. Among the top k similar users that watched movie m, sum up their similarities with user u for movie m. Then select the top n interesting movies and recommend to user u.

To predict user u's rating for movie i, we simply sum up the rating of top k similar users for movie i and then take the average:

$$r(u, i) = \frac{1}{k} \sum_{v \subset K} r(v, i)$$

3.3 Item-based Collaborative Filtering

There are also two main steps in item-based collaborative filtering[1]:

(1) Find similar items based on Jaccard similarity, cosine similarity or adjusted cosine similarity.
(2) Rate the item based on the weighted sum and recommend the top k items with the highest predicted rating.

Initially, we try to use adjusted cosine to calculate item similarities, but end up costing a lot of time and poor result because of the high sparsity of the matrix. Then we turned to use cosine similarity to represent the similarities between movies:

$$Sim_{mn} = \frac{|V(m) \cdot V(n)|}{\sqrt{|V(m)||V(n)|}}$$

Where V is the rating vector of a movie. The rating for movie m is defined as:

$$r(m) = \sum_{n \subset K} r(n) \times Sim_{uv}$$

Where K is the set of top k similar movies. Then among all the unrated movies for user u, take the top n movies with the highest predicted rating and recommend to user u.

3.4 Latent Factor Model

Other than the three models we construct, we also try to implement a simple Latent Factor Model. The basics of latent factor model is Single Value Decomposition. The key idea behind this is that the users' interests are embedded in a small number of hidden factors. With the user-movie matrix, we produce dimensional reduction on it in order to:

(1) Remove noisy features;
(2) Find hidden correlations between users and movies;

By reducing the dimension of the user-movie matrix, we are deriving user preferences from the raw data.

After we generate the training data set and construct user-movie matrix A, we simply fill the empty value with 0, which actually affect the rating prediction. Then we perform SVD algorithm on it and decompose it into the best lower rank approximation of A:

$$A_{m \times n} = U_{m \times m} \Sigma_{m \times n} V^T_{n \times n}$$

Where U is the user feature matrix, represents the users' interest; Σ is the diagonal matrix of singular values; V^T is the movie feature matrix, represents each feature's relevance to each movie.

After dimensional reduction, we take the dot product of U, Σ and V^T to predict the rating and recommend the movies with the highest rating.

4 EXPERIMENTS

4.1 Data Preprocessing

4.1.1 Bias Removal. We consider the ratings to be inconsistent because every user has their own habits of rating. Some users are harsh at giving high ratings while other are generous. We want to eliminate this issue so that the same rating score for two users means the same level of preference. To formalize, we have:

$$r_{xi} = b_x + b_i + r'_{xi}$$

Where r_{xi} is user x's rating of movie i, $b_x = \overline{r_x} - \overline{r}$ is the rating deviation of user x ((avg. rating of user x) – (overall mean movie rating)), $b_i = \overline{r_i} - \overline{r}$ is the rating deviation of movie i, r'_{xi} is the real unbiased rating. So we have:

$$r'_{xi} = r_{xi} - b_x - b_i$$
$$r'_{xi} = r_{xi} + 2 * \overline{r} - \overline{r_i} - \overline{r_x}$$

4.1.2 Time Correction. In the movieLens dataset, timestamp represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970. This doesn't have direct meanings to our study. We consider the ratings of movies would change over time, starting from the moment when a movie is released. We don't have the release date of movies in the dataset, so we set the release timestamp to be the earliest timestamp of movies. Then, we subtract the original timestamp by this release timestamp. At last, we scale every timestamp from seconds to weeks, as we think ratings wouldn't change rapidly. Figure 3 shows how the rating changes over time for some selected movies.

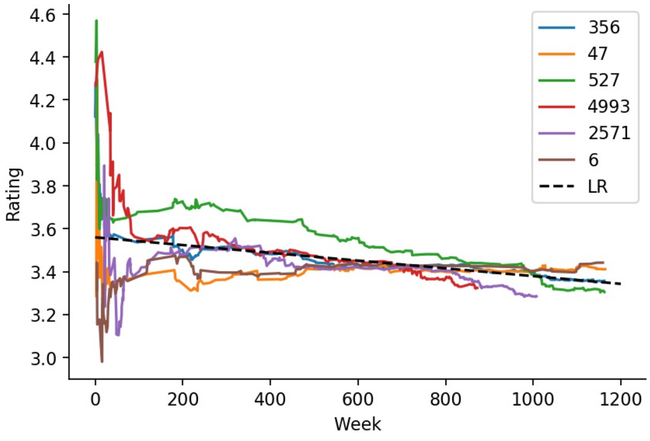


Figure 3: Ratings of some movies over time

We also fit a linear regression model (dash line in Figure 3) to see the correlation between ratings and time, by minimizing mean square error using the following equations:

$$r = kt + b$$
$$k = \frac{\sum_{t,r}(t - \bar{t})(r - \bar{r})}{\sum_r (r - \bar{r})^2} = -4.5739 \times 10^{-20}$$
$$b = \bar{r} - k\bar{t} = 3.5596$$

Where t is the timestamp.

As we can see in Figure 3, the fitted line shows that the average rating tend to decrease over time. To get the time-independent ratings, we can do a scaling according to this linear regression model:

$$r' = r - kt$$

Where r' is the time-independent ratings, r is the original ratings.

The results are shown in Figure 4. As we can see, the ratings tend to stabilize after certain amount of time.

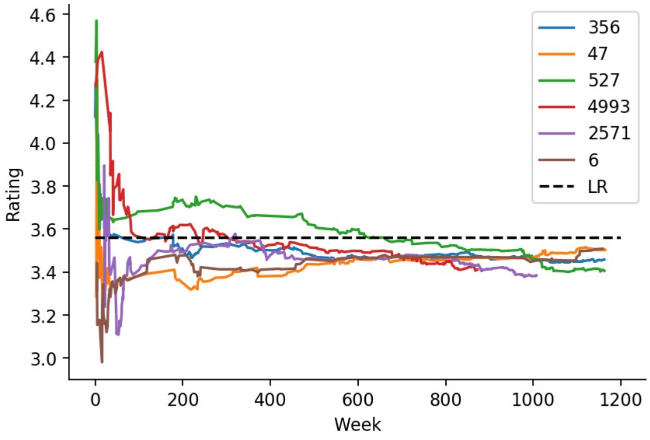


Figure 4: Ratings of movies over time after time correction

4.1.3 Dataset Splitting. We are doing a 5-fold cross-validation while doing experiments, so we split the dataset into 5 equal sized parts. Of these 5 parts, a single part is retained as the testing dataset, and the remaining 4 parts are used as the training dataset. This process is repeated for 5 times, so that every part is used as the testing dataset for exactly once.

We make sure that every user and every movie to appear in the training dataset of every fold, so that our models have some background knowledge for every user and every movie during the testing phase.

Eventually, the statistics are shown in Table 1. $fold_i$ means the i-th experiment of 5-fold cross validation. Each training dataset has all 610 users and 9724 movies.

4.2 Experiment Setup and Evaluation

We set the length recommendation list to 10. All experiments are run using this length.

For evaluating the performance of models, we choose 6 metrics. 4 of them are for evaluating recommendation, while the other 2 are for evaluating rating prediction.

For evaluating recommendation:

	# of rows in train set	# of rows in test set	# of users in test set	# of movies in test set
$fold_1$	82729	18107	609	3991
$fold_2$	82729	18107	608	3950
$fold_3$	82729	18107	609	3985
$fold_4$	82730	18106	608	3946
$fold_5$	82730	18106	610	3963

Table 1: Some statistics information of our dataset

(1) Precision: the fraction of correct recommendations over all recommendations.

$$Precision = \frac{1}{|X|} \sum_{x \in X} \frac{|R_x \cap T_x|}{|R_x|}$$

(2) Recall: the fraction of correct recommendations over the number of movies that users actually watched in the testing dataset.

$$Recall = \frac{1}{|X|} \sum_{x \in X} \frac{|R_x \cap T_x|}{|T_x|}$$

(3) F-measure: the harmonic mean of precision and recall.

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

(4) NDCG: the normalized discounted cumulative gain.

$$NDCG = \frac{\sum_{i=1}^{|R_x|} \frac{\mathbb{1}\{R_{xi} \in T_x\}}{log_2(i+1)}}{\sum_{i=1}^{|R_x \cap T_x|} \frac{1}{log_2(i+1)}}$$

For evaluating rating prediction:

(5) MAE: mean absolute error.

$$MAE = \frac{1}{|X|} \sum_{x \in X} \sum_{i=1}^{|R_x \cap T_x|} \frac{|r_{xi} - t_{xi}|}{|R_x \cap T_x|}$$

(6) RMSE: root-mean-square error.

$$RMSE = \frac{1}{|X|} \sum_{x \in X} \sqrt{\sum_{i=1}^{|R_x \cap T_x|} \frac{(r_{xi} - t_{xi})^2}{|R_x \cap T_x|}}$$

Where D_x is the set of recommended movies for user x, T_x is the set of movies in testing dataset that user x actually watched, r_{xi} is the predicted rating of user x to movie i, t_{xi} is the actual rating of user x to movie i in the testing dataset (ground truth).

4.3 Results

The 5-fold cross validation results are shown in Table 2, 3, 4, 5, Where $fold_i$ means the i-th experiment of 5-fold cross validation.

After doing some experiments, we notice that the second approach of building user profile has better performance. So we only report the results of the second approach in Table 2.

	Precision	Recall	F-measure	NDCG	MAE	RMSE
$fold_1$	0.043	0.014	0.022	0.017	0.717	0.940
$fold_2$	0.048	0.016	0.024	0.020	0.707	0.920
$fold_3$	0.042	0.014	0.021	0.017	0.667	0.869
$fold_4$	0.043	0.014	0.022	0.018	0.751	0.969
$fold_5$	0.043	0.015	0.022	0.018	0.640	0.820
avg.	0.044	0.015	0.022	0.018	0.696	0.904

Table 2: 5-fold cross validation results of content-based approach

	Precision	Recall	F-measure	NDCG	MAE	RMSE
$fold_1$	0.247	0.083	0.124	0.677	0.611	0.794
$fold_2$	0.238	0.080	0.120	0.670	0.629	0.817
$fold_3$	0.231	0.080	0.117	0.671	0.650	0.853
$fold_4$	0.240	0.080	0.121	0.685	0.622	0.802
$fold_5$	0.243	0.082	0.122	0.671	0.642	0.840
avg.	0.240	0.081	0.121	0.675	0.631	0.821

Table 3: 5-fold cross validation results of user-based collaborative filtering

	Precision	Recall	F-measure	NDCG	MAE	RMSE
$fold_1$	0.234	0.077	0.116	0.669	0.721	0.825
$fold_2$	0.231	0.071	0.162	0.666	0.737	0.838
$fold_3$	0.228	0.071	0.108	0.667	0.779	0.879
$fold_4$	0.232	0.072	0.110	0.671	0.732	0.830
$fold_5$	0.233	0.073	0.111	0.668	0.756	0.871
avg.	0.232	0.073	0.121	0.668	0.745	0.849

Table 4: 5-fold cross validation results of item-based collaborative filtering

	Precision	Recall	F-measure	NDCG	MAE	RMSE
$fold_1$	0.270	0.091	0.136	0.727	1.682	1.944
$fold_2$	0.265	0.089	0.133	0.705	1.713	1.973
$fold_3$	0.263	0.089	0.133	0.721	1.670	1.966
$fold_4$	0.261	0.088	0.132	0.723	1.664	1.937
$fold_5$	0.262	0.088	0.132	0.721	1.710	1.977
avg.	0.264	0.089	0.133	0.719	1.688	1.959

Table 5: 5-fold cross validation results of latent factor model

Model	Precision	Recall	F-measure	NDCG	MAE	RMSE
CB	0.035	0.012	0.018	0.014	0.689	0.899
UserCF	0.240	0.081	0.121	0.675	**0.631**	**0.821**
ItemCF	0.232	0.073	0.121	0.668	0.745	0.849
LF	**0.264**	**0.089**	**0.133**	**0.719**	1.688	1.959

Table 6: Average Performance of Each Model

The overall average performance of our models is shown in Table 6. As we can see, the latent factor model has the best performance in the movie recommendation task, while the user-based collaborative filtering model has the best performance in the rating prediction task. The content-based model doesn't have a competitive performance in the movie recommendation task, but has a good performance in the rating prediction task.

5 CONCLUSIONS AND FUTURE WORK

The goal of our project is to utilize what we have learned in class and put into practice. Movie recommendation is such a popular problem in building recommendation systems. Building a movie recommendation system involves different kinds of skills and knowledge. We first need to clean the data and do preprocessing to extract useful information. Then, we are able to build different models to make predictions and evaluate their performance. In this project, we implemented a content-based model, two collaborate filtering models and a latent factor model. Our understandings to these models are further improved during the implementation process.

There's still a lot of space for improvement. Actually, the most obvious one is to use more data. The movieLens website provides multiple datasets of various sizes, but we only used the smallest one which has 0.1 million records. We weren't able to use more due to the limitations of time. We can build a hybrid model of what we have right now, for better performance, if we had more time (or more team members). Another thing we have noticed is that, the movieLens dataset also provides a file `links.csv` which stores a mapping from `movieId` to `imdbId` and `tmdbId` (They are identifiers for movies used by MovieLens, IMDb, and TMDb respectively). IMDb[2] and TMDb[3] are two well-known public database of movies and ratings, we might be able to utilize the data on these two websites, i.e., extracting more features for our content-based model.

ACKNOWLEDGEMENT

Our deepest gratitude goes first and foremost to Dr. Yongfeng Zhang, instructor of massive data mining. His profound knowledge and interesting lectures inspired our experiments. We would also like to express our gratitude to our TA: Shuyuan Xu. His fair grading and positive working attitude make our study smoother.

High tribute shall be paid to the authors of these libraries: numpy, pandas, matplotlib, scipy.

REFERENCES

[1] Joseph Konstan Badrul Sarwar, George Karypis and John Riedl. 2001. *Item-based Collaborative Filtering Recommendation*. Vol. 1-15. GroupLensfiles.grouplens.org, Minnesota, MN, USA, 1–15.

[2] John S. Breese, David Heckerman, and Carl Kadie. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI'98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 43–52.

[3] Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl. 2007. *The Adaptive Web - Methods and Strategies of Web Personalization*. Springer-Verlag, Berlin, Heidelberg, 325. DOI:http://dx.doi.org/10.1007/978-3-540-72079-9

[4] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2010. *Recommender Systems Handbook*. Vol. 1-35. Springer, Boston, MA, USA, 1–35. DOI:http://dx.doi.org/10.1007/978-0-387-85820-3_1

[5] Wikipedia contributors. 2020. Latent variable model — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Latent%20variable%20model&oldid=908250118. (2020). [Online; accessed 22-April-2020].

[2] http://www.imdb.com

[3] https://www.themoviedb.org/