




 taingocbui / **phase4\_project**



















 **Code**


 Issues


 Pull requests


 Actions

 Projects

 Wiki

 Security

 Insights

 Settings

 **phase4\_project** Public

---

1 of 10

11/25/2024, 10:24 AM

1 Branch 0 Tags

**About** Code

**taingocbui** update presentation 5e55938 · 13 hours ago

photos	Update README and fin...	last week
.gitignore	update project4 index file	2 weeks ago
README.md	Update README	last week
notebook.pdf	Update README and fin...	last week
presentation.pdf	update presentation	13 hours ago
project4.ipynb	Update README	last week

**README**

# Collaborative Filtering Recommendation System

---

Author: Tai Ngoc Bui  
Date Completion: November 10th, 2024

## 1.Business Understanding

---

This project focuses on developing a robust recommendation system based on

Building a Recommendation System

- [Readme](#)
- [Activity](#)
- 0 stars**
- 1 watching**
- 0 forks**

Releases

No releases published  
[Create a new release](#)

Packages

No packages published  
[Publish your first package](#)

Languages

- Jupyter Notebook** 100.0%

the MovieLens dataset to provide top 5 movie recommendations to a user, based on their ratings of other movie. This dataset, from the GroupLens research lab at the University of Minnesota, contains tens of millions of movie rankings (a combination of a movie ID, a user ID, and a numeric rating), although the dataset used in this project only contains 100,000 of them.

This model will use root mean squared error (RMSE) as the main metric to ensure the accuracy of our prediction. This is perhaps the most popular metric of evaluation for recommender system. RMSE is a metric which measure how much the signal and the noise is explained by the model. It penalises way less when close to actual prediction and way more when far from actual prediction compared to Mean Absolute Error metric, another popular metric of evaluation.

## 2.Data Understanding

This dataset (ml-latest-small) describes 5-star rating and free-text tagging activity from [MovieLens dataset](#), a movie recommendation service. It contains 100836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996 and September 24, 2018. This dataset was generated on September 26, 2018.

All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided. The data are contained in the files links.csv, movies.csv, ratings.csv and tags.csv.

The dataset files are written as comma-separated values files with a single header row. Columns that contain commas (,) are escaped using double-quotes (").

### a. User Ids

.. user: user

MovieLens users were selected at random for inclusion. Their ids have been anonymized. User ids are consistent between ratings.csv and tags.csv (i.e., the same id refers to the same user across the two files).

## **b .Movie Ids**

Only movies with at least one rating or tag are included in the dataset. These movie ids are consistent with those used on the MovieLens web site (e.g., id 1 corresponds to the URL <https://movielens.org/movies/1>). Movie ids are consistent between ratings.csv, tags.csv, movies.csv, and links.csv (i.e., the same id refers to the same movie across these four data files).

## **c .Ratings Data File Structure (ratings.csv)**

All ratings are contained in the file ratings.csv. Each line of this file after the header row represents one rating of one movie by one user, and has the following format: userId, movieId, rating, timestamp

The lines within this file are ordered first by userId, then, within user, by movieId. Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars). Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

## **d. Tags Data File Structure (tags.csv)**

All tags are contained in the file tags.csv. Each line of this file after the header row represents one tag applied to one movie by one user, and has the following format: userId, movieId, tag, timestamp

The lines within this file are ordered first by userId, then, within user, by movieId.

Tags are user-generated metadata about movies. Each tag is typically a single

tags are user-generated metadata about movies. Each tag is typically a single word or short phrase. The meaning, value, and purpose of a particular tag is determined by each user.

Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

#### **e. Movies Data File Structure (movies.csv)**

Movie information is contained in the file movies.csv. Each line of this file after the header row represents one movie, and has the following format: movieId, title, genres

Movie titles are entered manually or imported from <https://www.themoviedb.org/>, and include the year of release in parentheses. Errors and inconsistencies may exist in these titles.

Genres are a pipe-separated list, and are selected from the following: Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western, (no genres listed)

#### **f. Links Data File Structure (links.csv)**

Identifiers that can be used to link to other sources of movie data are contained in the file links.csv. Each line of this file after the header row represents one movie, and has the following format: movieId, imdbId, tmdbId

- movieId is an identifier for movies used by <https://movielens.org>. E.g., the movie Toy Story has the link <https://movielens.org/movies/1>.
- imdbId is an identifier for movies used by <http://www.imdb.com>. E.g., the movie Toy Story has the link <http://www.imdb.com/title/tt0114709/>.
- tmdbId is an identifier for movies used by <https://www.themoviedb.org>.

- tmdbid is an identifier for movies used by <https://www.themoviedb.org>.  
E.g., the movie Toy Story has the link <https://www.themoviedb.org/movie/862>.

Use of the resources listed above is subject to the terms of each provider.

### 3. Objectives

---

Develop a Collaborative Filtering recommendation system to provides top 5 movie recommendations to a user, based on their ratings of other movies. This recommendation system should be scalable and achieve an acceptable low root mean squared error metric.

### 4. Collaborative Filtering

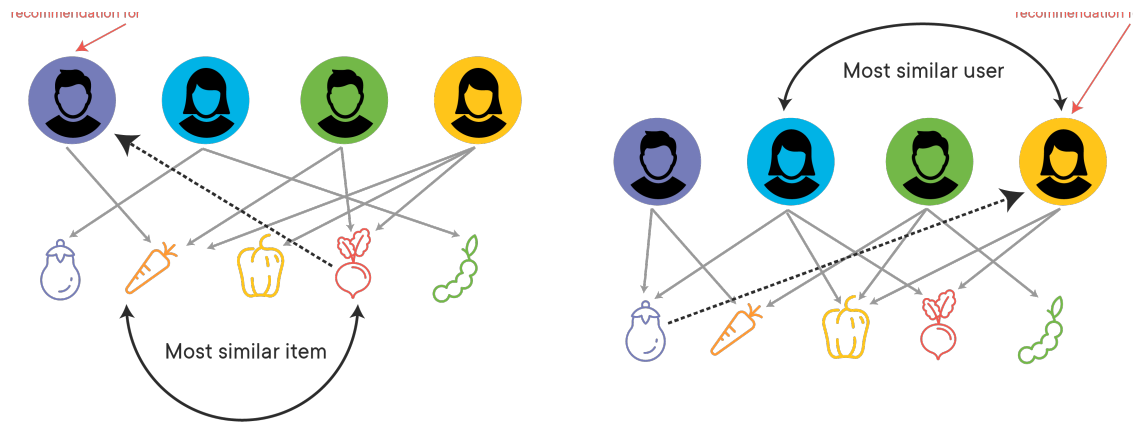
---

Two most ubiquitous types of personalized recommendation systems are Content-Based and Collaborative Filtering. The Collaborative Filtering Recommender is entirely based on the past behavior and not on the context. More specifically, it is based on the similarity in preferences, tastes and choices of two users. It analyses how similar the tastes of one user is to another and makes recommendations on the basis of that. In contrast, content-based recommendation systems focus on the attributes of the items and give you recommendations based on the similarity between them.

As the number of users are much smaller compared the number of movies in this particular project, I will use Collaborative Filtering as the main approach for this Recommender System. In general, collaborative filtering is the workhorse of recommender engines. It can be divided into Memory-Based Collaborative Filtering and Model-Based Collaborative filtering.

Item-Based Collaborative Filtering

User-Based Collaborative Filtering



### a. Base Model (KNNBasic)

To start with, I use KNNBasic as the baseline model with both cosine and pearson similarity function. It is easy to understand and implement. It uses the k-nearest neighbors approach to find similar users or items based on interaction data. Both model had RMSE of about 0.97, meaning that it was off by almost 1 point for each guess it made for ratings.

Cosine similarity and Pearson similarity are both metrics for measuring similarity but differ in focus and application. Cosine similarity measures the cosine of the angle between two vectors, focusing on their direction while ignoring magnitude, making it ideal for comparing high-dimensional or sparse data where the scale is irrelevant. In contrast, Pearson similarity evaluates the linear correlation between two vectors, accounting for their mean and relative deviations, making it suitable for analyzing relationships or trends in the data. While cosine similarity emphasizes vector orientation, Pearson similarity captures how well one variable's variations predict another's, regardless of scaling or translation.

### b. KNNBaseline

To further improve the neighborhood-based model, I will apply the KNNBaseline method, adding in bias term to reduce overfitting. KNNBasic directly computes similarity between users or items based on raw ratings, making it simple and intuitive but less effective in datasets with systematic biases, such as users who consistently rate items higher or lower than average. On the other hand, KNNBaseline accounts for these biases by incorporating baseline estimates for ratings, which adjust for global, user-specific, and item-specific biases. By adding in bias term, KNNBaseline method achieved the best RMSE of 0.87 with both Pearson and Cosine similarity.

Overall, Memory-based Collaborative Filtering is easy to implement and produce reasonable prediction quality. However, there are some drawback of this approach:

- It doesn't address the well-known cold-start problem, that is when new user or new item enters the system.
- It can't deal with sparse data, meaning it's hard to find users that have rated the same items.
- It suffers when new users or items that don't have any ratings enter the system.
- It tends to recommend popular items.

### c. Matrix Factorization (Singular Vector Decomposition)

Besides memory-based methods, model-based method is another popular approach of Collaborative Filtering. Unlike memory-based collaborative filtering, which directly calculates similarity scores between users or items (such as user-based or item-based nearest-neighbor algorithms), model-based approaches use a data-driven model to generalize patterns in the data.

In model-based collaborative filtering, a model is trained to learn hidden (latent)



factors that represent users and items. These factors capture preferences in a reduced feature space. Techniques like matrix factorization (e.g., Singular Value Decomposition (SVD), Non-negative Matrix Factorization (NMF)) are popular for learning these latent factors, effectively representing users and items as vectors in this space.

Matrix factorization is widely used for recommender systems where it can deal better with scalability and sparsity than Memory-based CF. The goal of MF is to learn the latent preferences of users and the latent attributes of items from known ratings (learn features that describe the characteristics of ratings) to then predict the unknown ratings through the dot product of the latent features of users and

While model-based collaborative filtering excels in scalability and predictive power, its drawbacks include:

- challenges with new users/items (similar to memory-based model)
- high initial computational cost for the decomposition
- overfitting risks
- requires periodic retraining to incorporate new interactions

With this MovieLens dataset, SVD is the best model with the lowest test set root squared mean error of 0.8519. The parameters used for tuning SVD is as follows: 'lr\_all': 0.005, 'n\_factors': 250, 'reg\_all': 0.05, 'n\_epochs': 100

## 5. Conclusion

---

Based on our analysis and testing with different models, I want to recommend Model-based Singular Value Decomposition (SVD) with user-based focus as the final model for the recommendation system given the MovieLens dataset. The

parameters used in this model is 'lr\_all' = 0.005, 'n\_factors' = 250, 'reg\_all' = 0.05, and 'n\_epochs' = 100. Based on our test results, this model can achieve 0.8519 root mean squared error.

Another reason for using Model-based SVD is its scalability for large scale applications. This scalability ensures that SVD-based systems can handle millions of users and items effectively. Moreover, SVD can uncover latent relationships between users and items, such as implicit preferences or themes, which are not immediately visible in the raw interaction matrix. These latent features improve the system's ability to predict user preferences, even for items that the user has not explicitly interacted with.

## 6. Future Works

---

To better improve the quality of this report, I will extend this project by investigate deep learning-based methods like Neural Collaborative Filtering (NCF) or autoencoders for better capture of complex user-item interactions. Another possibility to improve this project is to combine collaborative filtering with content-based filtering or context-aware techniques to address limitations like cold-start problems.