Movie Recommender System

Yasmin Karimi

June 2023

There are a variety of problems related to recommendation engines which are used across numerous companies and industries. Using recommender systems in many businesses can help with increasing: revenue, Customer satisfaction, personalization, etc. It's said that Amazon's AI-powered recommendation engine is responsible for over one-third of its sales, making it one of the most valuable AI systems on the planet.

The dataset that I've used for my project is originally from '[Movie Lens'] and it is publicly available both through Movie Lens website and [Kaggle]. My target tables are Movies, Which include the information about movies. And rating, which includes the user ratings for various movies. Since the data was huge to process and due to the computation limitations of personal laptop, I took a sample out of both files.

Cleaning and Preprocessing of Data

- Removing the nulls from both files.
- Removing duplications form Movies table.
- Dropping the unnecessary columns
- Converting non-numeric columns to numeric in the Movies table
- Sampling 5% of the ratings
- Out of the samples, taking out the ratings from the user who have rated more than 10 movies.
- From the Movies table keeping the movies which have more than 100 reviewers voted for them.

Insights, Modeling, Results

1. Content-based Filtering

In this method of learning, the recommendation is based on other items similar to what the user likes, and the relationship between that item and its features. As a statement, it can be said, "Because you liked this, you may also like those". The calculations are happening based on item-item relationship. In this project I've used 'Genre' and 'Overview' for my content features. Based on the similarity of genres or overviews the model will recommend you a movie.

2. Collaborative Filtering

This filtering method is implemented through the User-Item Matrix. In other words, how a user behaves to a movie or the rating that a user gives to that movie (each movie is considered to be an "item" in this project). This filtering consists of two subcategories which are user-user filtering and item-item filtering. However, as it only considers the previous interactions to make recommendations, collaborative filtering suffer from the "cold start problem" which means, it is impossible to recommend anything to new users or to recommend a new item to any users and many users or items have too few interactions to be efficiently handled.

3. User-based Filtering

In order to make a new recommendation to a user, user-user method roughly tries to identify users with the most similar “interactions profile” (nearest neighbors) in order to suggest items that are the most popular among these neighbors (and that are “new” to our user.)

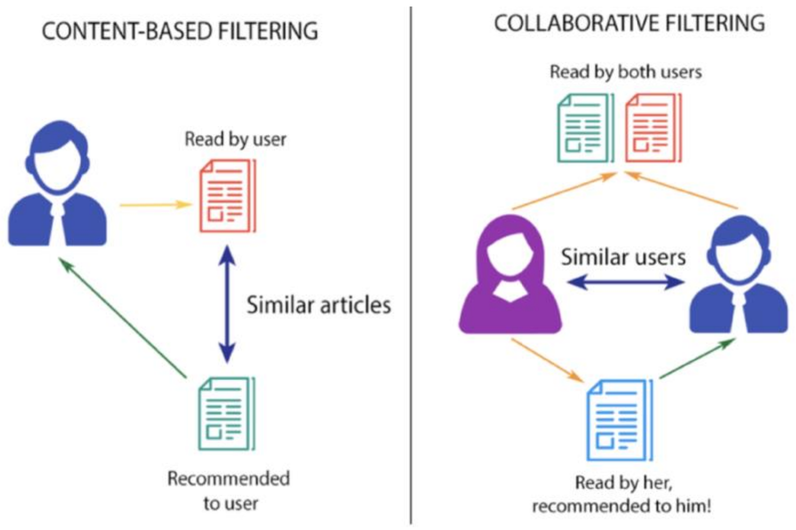


Figure 1. Overview of filtering methods

The project mostly uses TF-IDF, which is a very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine learning algorithms for prediction. Another Machine learning algorithm that has been user for our collaborative filtering is Funk SVD (which stands for Funk Singular Value Decomposition). SVD can be useful because it allows us to use regression-based metrics like MSE or MAE to assess performance. In this way understand the metric before deploying our recommendations to customers.

Prior to fitting the ‘Movie title’ into the functions, the title gets checked in the movies table, if there is no such movie, there might be a spelling error, so the function will send the title to ‘find_similar_movies’ and returns a list of movies that are suspected to have a similar name. Since the method uses TF-TDF method on words by vectorizing word by word, if the title of the movie has one letter and matches one of the letters in the title searched that would be returned and to fix this problem we have to vectorize by letters, which takes a lot of storage and that is why it is not included in this project. (e.g., title searched: ‘Barman’, result includes: ‘m’)

When we are designing our recommender system, we need to consider the service we’re providing as well as the data we have. if the user is a new user, we have to wait and learn by the small interactions they do but if we have data on the user we can user collaborative filtering or content based or a hybrid system which uses information from both models.

Findings and conclusions

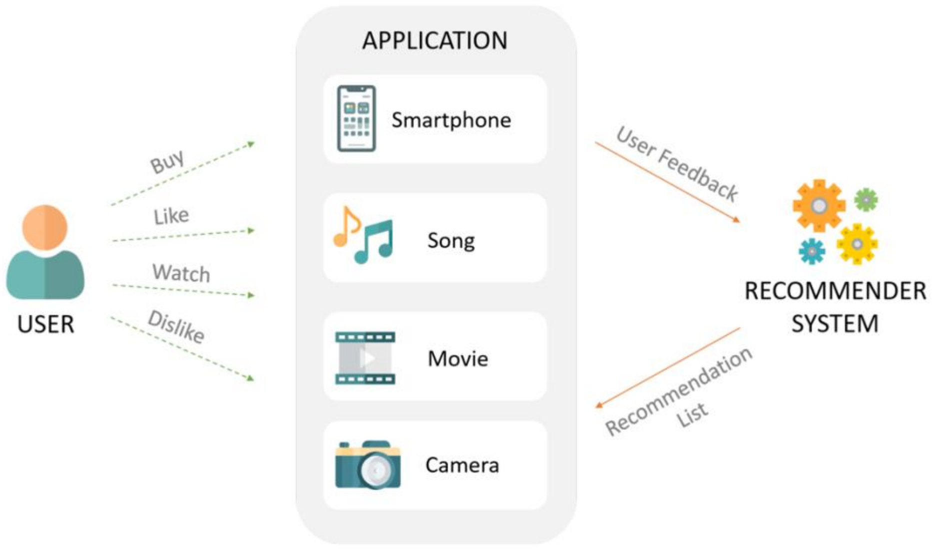
The Collaborative-filtering method which uses Funk SVD, by having a ‘UserID’ and a ‘MovieID’ will return an ‘expected rating’ for the movie, that can indicate whether the user likes the movie or not.

Content-based filtering with input of one favorite ‘movie title’ will output 10 recommendations based on ‘Genre’ and ‘Overview’ features. In total 20 movies will be recommended.

User-based filtering method with input of a ‘movie title’ will return up to 10 movie titles with some information about the movies.

Hyperparameter optimization can help us a lot in collaborative filtering to use to get realistic and more accurate results. In this type of filtering We can fit machine learning models and try to predict how many ratings will a user give a product. Including: Deep learning methods, clustering algorithms or matrix factorization, which I used a version of it with dimensionality reduction method of Funk SVD.

The application of Recommender systems are broad. they can be used in: Media, retail, banking, etc. For this case it can potentially be applied in a website for recommending movies or tv-shows to people.


APPLICATION
Smartphone
Song
Movie
Camera
USER
Buy
Like
Watch
Dislike
User Feedback
RECOMMENDER SYSTEM
Recommendation List