2021

Movie Recommender System

SANJAY JARAS
https://sanjayjaras.github.io/

BELLEVUE UNIVERSITY

Abstract

The amount of content available on platforms like youtube, Netflix, Hulu, Disney is increasing each day. With the increasing number of available content, viewers started spending a lot of time searching for content of their likings. A good recommendation engine can save a lot of time for all the viewers while finding the content of their taste. A great recommender system avoids viewers wasting time watching trailers or movies for a few minutes and then switching to a different one as they did not like it. Without a recommendation system, a user might leave the platform if they cannot find a product or content of their taste. . Recommendation engines or recommender systems filter a large list of products, movies, or things to present only products or things that customers might be interested in. Recommender systems are machine learning systems that help us discover new products, media content, and services depending on their earlier activities.

Intro/background of the problem

On the internet and media platforms like Netflix, youtube it is very easy to get lost while searching for products or content as per our likings. If the platform does not suggest any products or content, we might waste a lot of our time searching for products or content that we are interested in. For example, if we just started using Netflix, it will show us top-rated movies or top charts. Once we watch some movies it will recommend new movies that are similar to movies we watched. Now assume there are no recommendations available from Netflix, we need to search through a big list of 4000 movies to find a movie that interests us. When we search for similar movies on google, in the background google uses a recommendation system to find a similar movies list There are many applications of

recommender systems like movies, series, various products on e-commerce sites, etc. A good recommendation system not only saves users/customers time but also keeps him/her engage with the platform. Without a good recommendation system, a user might leave the platform if they cannot find a product or content of his/her liking. With a good recommendation system, we can keep users/customers happy and engaged, so that they will keep using a platform for a longer duration.

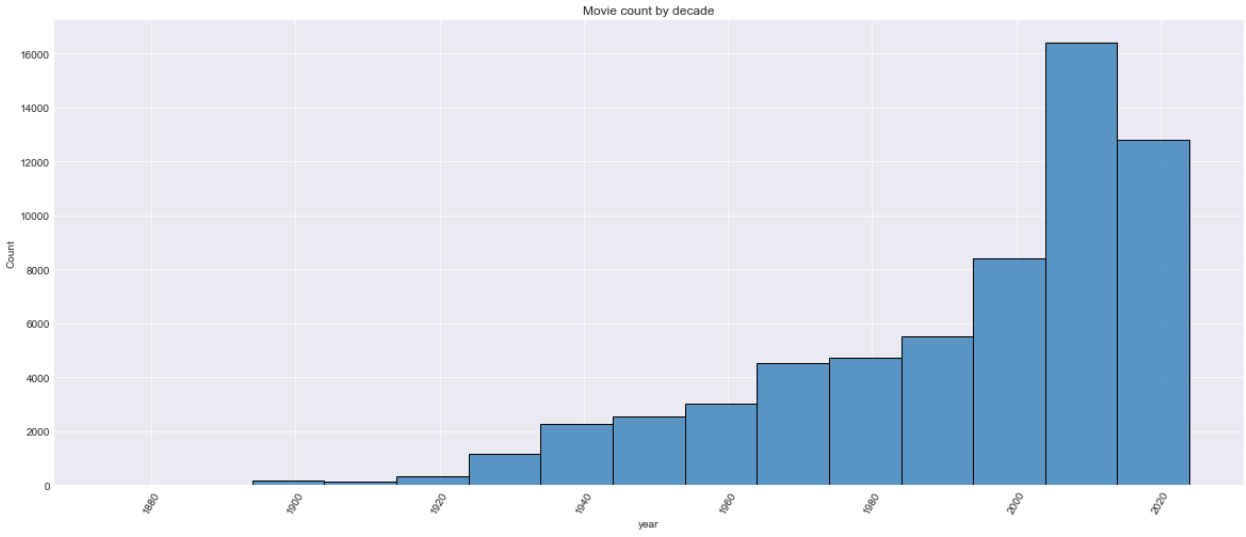
Recommendation engines or recommender systems filter a large list of products or things to present only products or things that customers might be interested in. There are many applications of recommender systems like movies, series, various products on e-commerce sites, etc. When we search the similar movies on google, google uses a recommendation system in the background to find a similar movies list. There are four types of recommendation engines:

1. Content-based: This recommender system uses attributes of the products or movies to find the ranking and similar products and movies.
2. Collaborative filtering based: This recommender system uses different algorithms to find similar users by some activities they performed like the movie rating, product likes, or movie reviews.
3. Popularity-based: This type of recommender system will suggest products or movies depending on the popularity.
4. Hybrid: These recommender systems are created by using two or more types of the above-mentioned recommender systems.

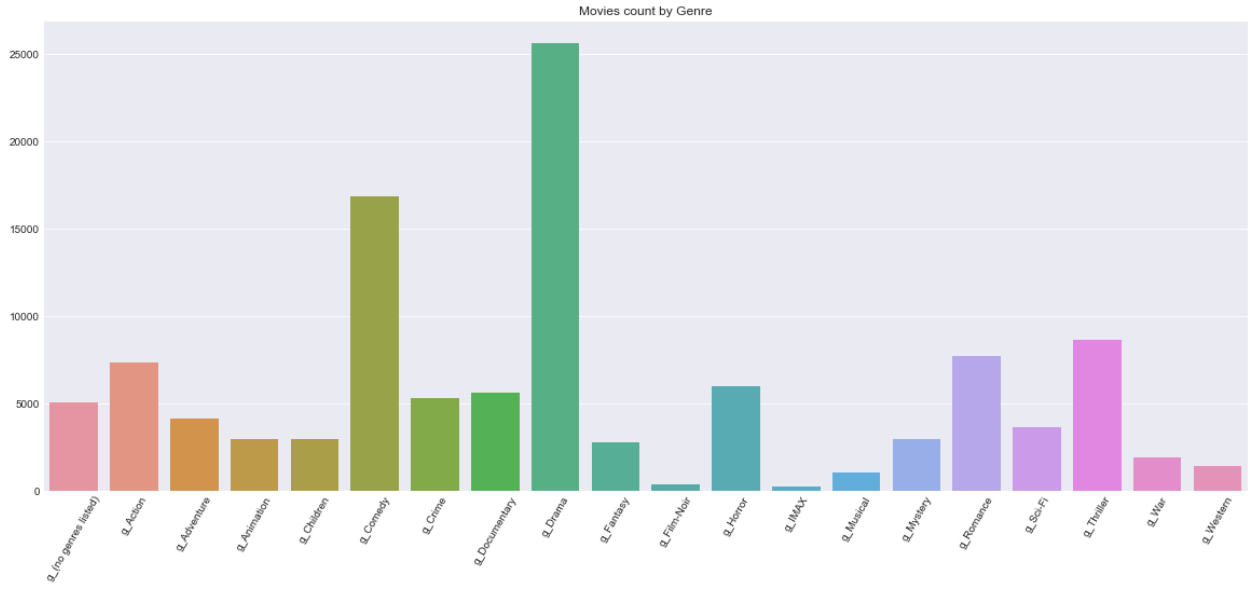
If we use the hybrid type of recommendation engine, it can drastically improve suggested recommendations.

Methods

The dataset is chosen from two sources, rating data is downloaded from grouplens.org (https://grouplens.org/datasets/movielens/) and movies metadata is downloaded from Kaggle.com (https://www.kaggle.com/rounakbanik/movie-recommender-systems/data?select=movies_metadata.csv). The rating dataset contains ratings for 62,000 movies by 162,000 users. It contains 25 million movie ratings. The metadata dataset contains metadata about 45,000 movies.



EDA is performed to understand the data and to do data cleanup. The movie dataset is having 3 columns movieId, title, and genres. The “genres” is a pipe-separated string column. I converted this column to dummy columns to find out different statistics.



The “link” dataset is having 3 columns movieId, imdbId, and tmdbId. The “imdbId” can be used to connect the movielense dataset to IMDb datasets and similarly, tmdbId can be used to connect the movielense dataset to the TMDB movie dataset. For this project, I have used the “tmdb” dataset to get metadata of movies and use that for the content-based recommendation. The rating dataset has information about user-wise ratings. The rating dataset has 4 columns userId, movieId, rating, and timestamp. From these columns, I am not using timestamp columns. The genome_scores and genome_tags datasets are not used for this project. The metadata dataset has 25 columns, out of those 25 columns we have used id, title, genres, overview, tagline, vote_average, and vote_count columns. The Id column is the relational key to tmdbId from the movies dataset. The “genres” column is in JSON format, so I converted it to the list. The overview column has a description of movies. The tagline contains a short tagline indicating the content of movies.

If we want to recommend movies, we can simply go by the high-rated movies(the average rating for a movie). However, with this approach, we will treat a movie with 5 users the

same as 5000 users. To address this issue, I started with a simple weighted-rating recommendation. This approach considers the numbers of users that rated the movies as well along with the average rating. The model can recommend movies by genre or overall top-charts.

The above-mentioned approach simply goes by the average rating and the number of users who rated each movie, it doesn't care about the content of a movie. To address this, I added a content-based recommendation to further improve the recommendations. I added cosine similarity by using overview and tagline columns from the metadata dataset. By cosine similarity, we can find movies similar to any movie user already watched by comparing the tagline and overview of those movies This approach combines content-based recommendations with weighted-rating recommendations. In this approach, first, we found similar movies by using the cosine similarity and then pass this list to find movies by weighted rating approach.

In the above approach, we are not considering what is user's taste is. In the next approach, I used a surprise library to build a model. This model uses a collaborative filtering approach. This approach finds similar users by their ratings given on earlier watched movies and then tries to predict the rating a user can give to a movie. This approach is good enough to predict movie ratings and recommend movies. This model shows an accuracy of 93% on test data.

I tried further to improve the recommendations by using a hybrid model. With the hybrid model, I combined the above-mentioned weighted-rating, content-based recommendation, and collaborative filtering. By combining these three models we can find similar movies,

then find estimated ratings for those movies, and also filter the movies with low weighted ratings.

Results

Under this project, I created four types of recommendation models. The first model is purely based on average ratings and the number of users rated for each movie. This model finds the weighted ratings and returns top-n movies with greater weighted ratings.

In the second model, we used a content-based recommendation approach by using overview and tagline columns from the metadata dataset. This model calculates similarity scores by using cosine similarity and then returns more similar movies by sorting movies by similarity score.

In the third model, I combined the above two models content-based recommendations and weighted-rating recommendations. In this approach, first, we found similar movies by using the cosine similarity and then pass this list to find movies by weighted rating approach.

In the fourth model, we used the collaborative filtering model. For this model, I used a singular value decomposition algorithm. With this approach, we tried to find similar users and tried to predict the estimated rating for a user for a movie. This model can predict movie ratings for a user with a 93% accuracy on test data.

In the final hybrid model, I combined the above-mentioned models' weighted-rating, content-based recommendation, and collaborative filtering recommendation. With this model, we get movies with high ratings, similar to the movie watched by the user and movies for which the user might give a higher rating.

Discussion/conclusion – Next steps

Hybrid Recommendation systems usually perform better than recommendation systems based on any of the one types. These models combine the advantages of different recommendation models into one. For the current content-based recommendation model I used genre, tagline, spoken language, and overview as the features. The content-based model I created can be further improved by including more features under content-based filtering. The improvement in one model under the hybrid model can increase the accuracy of the hybrid model.

Acknowledgments

The authors of this project have referred to Gouplense and Kaggle for the data set. We referred to some notebooks available on the Kaggle. There is a significant number of solutions available on the Internet for the problem. We have referred to some of those available solutions and have designed our solution. We have also referred to multiple websites(not limited to the below references) including data science articles from various authors, machine learning websites, mediam.com, and many others for the basic machine learning and statistics concepts and practical examples.

References

1. [1] Beginner Tutorial: Recommender Systems in Python – Aditya Sharma – May 2020 - https://www.datacamp.com/community/tutorials/recommender-systems-python

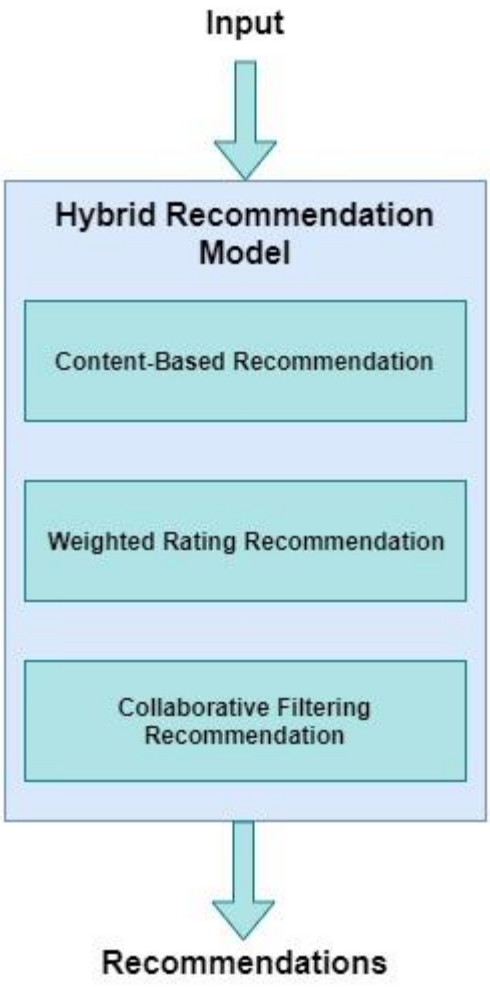
2. [2] Movies Recommendation - Şevval Yurtekin - https://www.kaggle.com/sevvalyurtekin/movies-recommendation
3. [3] Recommender system - https://en.wikipedia.org/wiki/Recommender_system#:~:text=A%20recommender%20system%2C%20or%20a,would%20give%20to%20an%20item.
4. [4] Introduction to recommender systems – Baptiste Rocca – Jun 2019 - https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada
5. [5] Recommendation systems: Principles, methods, and evaluation - F.O.IsinkayeaY.O.FolajimibB.A.Ojokoh – Nov 2015 - https://www.sciencedirect.com/science/article/pii/S1110866515000341
6. [6] Building a Movie Recommendation Engine in Python using Scikit-Learn – Suman Adhikari – Feb 2019 - https://medium.com/@sumanadhikari/building-a-movie-recommendation-engine-using-scikit-learn-8dbb11c5aa4b
7. [7] Build A Movie Recommendation Engine Using Python - Aug 2019 - https://medium.com/analytics-vidhya/build-a-movie-recommendation-engine-using-python-scikit-learn-machine-learning-e68ba297e163
8. [8] Movies Recommender System - Rounak Banik - https://www.kaggle.com/rounakbanik/movie-recommender-systems
9. [9] recommendation through movie_lens_rating - Karan Choudhary - https://www.kaggle.com/karanchoudhary103/recommendation-through-movie-lens-rating?select=tag.csv

10. [10] An In-Depth Guide to How Recommender Systems Work - Badreesh Shetty – Jul 2019 - https://builtin.com/data-science/recommender-systems

Appendix

For this project, I have used the Hybrid recommendation model. This Hybrid recommendation model works in the following way.

1. Find the movies similar to the movie user already watched by sorting on similarity score by using content-based filtering.
2. Remove movies with low weighted ratings depending on the parameter
3. Predict the estimated rating for these movies by using collaborative filtering and get the top movies with higher estimated ratings.


Input
Hybrid Recommendation Model
Content-Based Recommendation
Weighted Rating Recommendation
Collaborative Filtering Recommendation
Recommendations