Movie Recommendation System: A hybrid approach

Aditya Maheshwari
Rutgers University, DCS
aditya.maheshwari@rutgers.edu

Ezhil Nikhilan Chitrambalam
Rutgers University, DCS
ezhil.nikhilan@rutgers.edu

Ronit De
Rutgers University, DCS
ronit.de@rutgers.edu

ABSTRACT

In contemporary online platforms, recommendation systems have achieved pervasive implementation. Our goal is to construct a movie recommendation system founded on the "MovieLens" dataset, which amalgamates personalized user preferences with movie features like genre and popularity. We have established multiple models, namely popularity, content-based, collaborative filtering, and latent factor-based models, and conducted comprehensive hyper-parameter tuning, testing accuracy, and recommendation evaluation for each model. We have integrated the predictions of latent factor-based and collaborative filtering methods to build a linear combination model, which attains enhanced accuracy in predicted ratings. By fusing all the methods, we have devised a hybrid model that surpasses individual models in terms of recommendation quality and diversity. We have provided an exhaustive analysis of each model in our report.

1 INTRODUCTION AND RELATED WORK

In the current era of exponential data growth and consumer behavior tracking, personalized services have become an essential aspect of technology. Collaborative Filtering and Content-Based Filtering are the two fundamental approaches for recommendation systems as identified by *Ferman* 2002. However, in 2005, Adomavicius G. [n.d.] introduced a hybrid approach that merges both methods, mitigating their individual limitations. This hybrid approach has now become the most widely used method for recommendation systems. Singular Value Decomposition (SVD) is a commonly utilized algorithm for performing Collaborative Filtering. SVD employs Matrix Factorization on the user-movie matrix to produce decomposition vectors that demonstrate similarities between movies.

2 PROBLEM FORMALIZATION

2.1 About the Data

The key dataset employed for this project is the grouplens movie review dataset, specifically the ml-latest-small subset. This dataset comprises 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service, and encompasses 27,753,444 reviews across 58,098 distinct movies by 283,228 users. Each rating within the dataset ranges from 0 to 5. Additionally, the dataset includes several features such as the timestamp of the review, the movie's genre, user-generated comments, and the corresponding IMDB and TMDB identification numbers for each movie.

Conference'17, Washington, DC, USA
© 2016 ACM. 978-x-xxxx-xxxx-x/YY/MM...$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

2.2 Data Preprocessing

2.2.1 Partitioning. The dataset was partitioned into test and train subsets based on UserID, with 80% of a user's reviews being used to train the model and the remaining 20% to validate its accuracy. This was achieved using the train test split method in the scikit-learn library, where the stratify attribute was used to divide the dataset based on a specified feature.

2.2.2 Cleaning . Data cleaning involved removing redundant and irrelevant features from the dataset and converting categorical features such as genres into one-hot vectors. Users with fewer than 5 data points were excluded from the dataset since at least one data point was necessary for testing purposes.

2.2.3 Augmentations. To enhance our understanding of movies for content-based recommendations and the significance of movie features, we integrated the dataset with the publicly available IMDB and TMDB datasets, incorporating features such as release date, writer, director, cast information, tagline, overview, popularity, and ratings and vote counts from their respective websites.

2.3 Data Analysis

(1) **Number of Ratings per Genre**

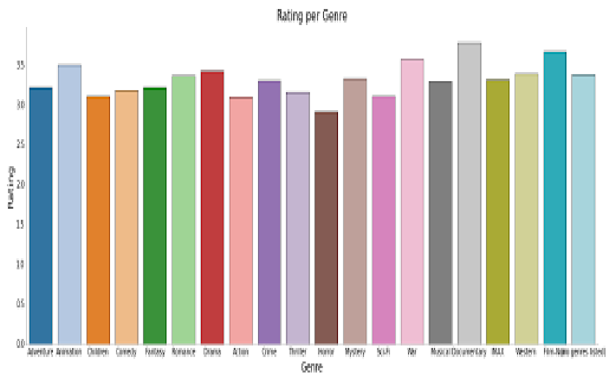


Figure 1: Number of Ratings per Genre

(2) **Number of Movies per Genre**

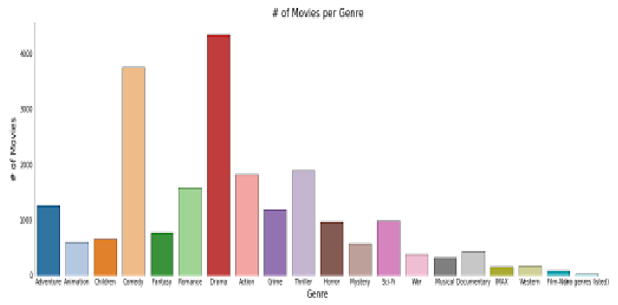


Figure 2: Number of Movies per Genre

(3) **Number of Users for each Rating**

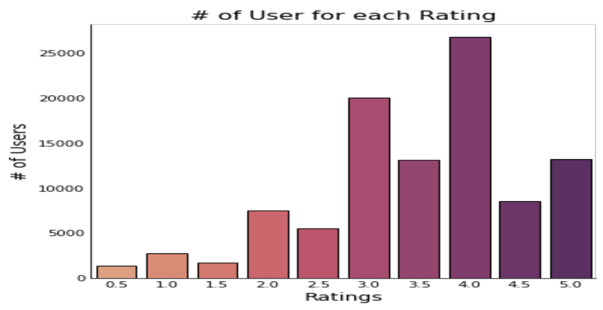


Figure 3: Number of Users for each Rating

(4) **Number of Ratings per User**

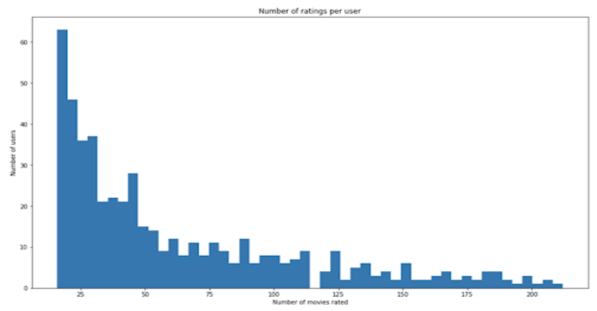
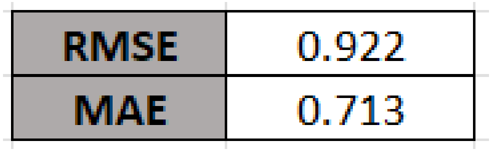


Figure 4: Number of Ratings per User

3 THE PROPOSED MODEL

We propose a model that attempts to combine the benefits of both content-based and collaborative filtering methods. In the following subsections, we explore these models that we integrate within our final model.



RMSE	0.922
MAE	0.713

Figure 5: Relatively higher error for Content-based filtering

3.1 Content-Based Recommendation Component

Content based recommendation system works on the idea of finding similar items based on the content of items. In case of movies, we tried to include the main features of movies like genres, release year, overview and tagline. The following two approaches were tried:

3.1.1 Generating user vector and item vector: Movies can be well described by the genre it falls under. And this property is again used to prepare a user vector. Movie vector = Vector of zeros and ones with a one for a relevant genre User vector = Average rating of the user for that particular genre

3.1.2 Movie-Movie (Item-Item) Similarity: The process of combining MovieLens data with TMDB data involved extracting movie overview and tagline information, followed by applying the TF-IDF vectorizer to generate a vector for each movie. Then, cosine similarity was calculated for each movie-movie pair. However, upon analyzing the results, it was observed that the content-based approach, which relied solely on movie descriptions, was not taking the genre of the movies into account. Instead, it was only considering the frequency of common words in the description. For instance, movies containing Christmas and Santa appeared in a similar list.
Thus we also added the genre information while constructing the movie description. Again, adding genre just once did not change the results. Hence, we added this information twice.

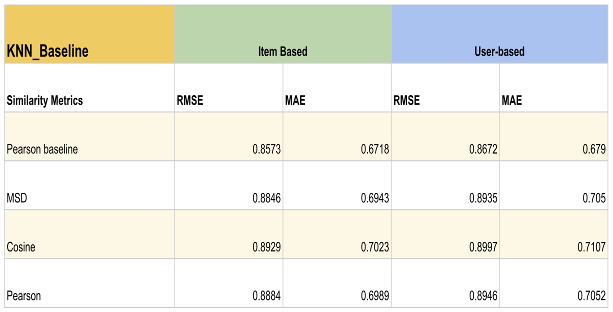
Now, by the inherent property of TF-IDF vectorizer, it reduces the weight of terms frequently occurring over the entire document. Thus, as the names of genre will appear in all the movies, it will be given lesser importance. Thus, we changed the TF-IDF vectorizer to count vectorizer and repeated the experiments.

3.2 Collaborative Filtering Component

Collaborative filtering is a technique that uses similarities between users and items to make recommendations. It can be divided into two main categories: user-based and item-based. In user-based collaborative filtering, we find users who have similar preferences to the target user and recommend items that those similar users have liked. In item-based collaborative filtering, we find items that are similar to the items that the target user has liked and recommend those similar items.

Surprise is a Python library for building and analyzing recommender systems that provides a variety of algorithms for collaborative filtering, including KNN-based models. KNNBasic, KNNwithMeans, KNNwithZScore, and KNNBaseline are variants of the K-Nearest Neighbors algorithm, which is a type of collaborative filtering algorithm that looks for the k most similar users (or items) to the target user (or item) and makes predictions based on their ratings.

The choice of similarity metric is important in collaborative filtering because it affects how similar users (or items) are determined. Cosine similarity is a common metric used in collaborative filtering because it measures the angle between two vectors and is unaffected by the magnitude of the vectors. Mean squared difference and Pearson correlation are also commonly used metrics. Pearson Baseline is a variant of Pearson correlation that takes into account a user's baseline rating (i.e., their average rating) when computing the correlation.



KNN_Baseline	Item Based		User-based	
Similarity Metrics	RMSE	MAE	RMSE	MAE
Pearson baseline	0.8573	0.6718	0.8672	0.679
MSD	0.8846	0.6943	0.8935	0.705
Cosine	0.8929	0.7023	0.8997	0.7107
Pearson	0.8884	0.6989	0.8946	0.7052

Figure 6: Item-Item vs User-User CF for KNN Baseline

3.3 Popularity based recommendation component

By integrating the MovieLens dataset with TMDB data, we were able to derive a new feature called "popularity" that is based on various user activities such as average rating, number of views, likes, favorites, watchlist additions, and release date. With this new feature, we created a list of popular movies for each genre, which can be utilized during the recommendation process to cater to the user's preferred genres.

Apart from popularity, we have also considered the weighted ratings as used by the legacy IMDB system. The weighted ratings can be computed using the following equation:

$$W = R * (v/(v + m)) + C * (m/(v + m))$$

where,
W = Weighted Rating
R = Average movie rating on a scale of 1-10
v = No. of votes for the movie
m = Minimum votes required to be listed in top 10 percentile
C = Mean "vote-average"

Table 1: Error Metrics for the Popularity-Based Model

RMSE	0.93
MAE	0.718

4 EXPERIMENTS

4.1 Individual method analysis

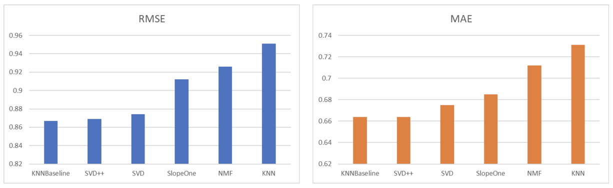


Figure 7: RMSE and MAE Values:
The KNN Baseline model delivers the lowest error values. This also translates to the other accuracy metrics.

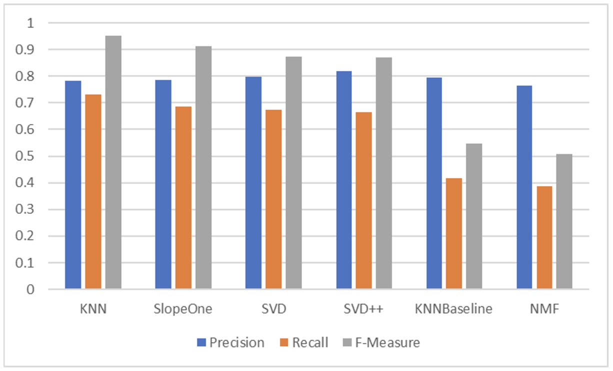


Figure 8: Precision, Recall, F-measure
The KNN model with Pearson Baseline produces the best results here despite higher errors.

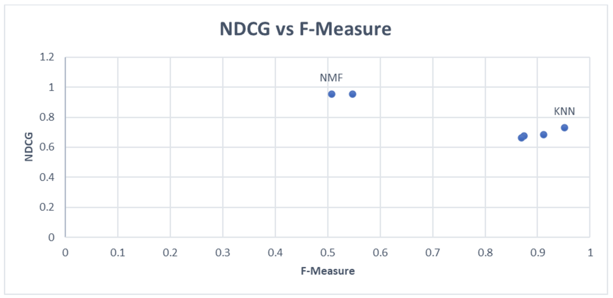


Figure 9: NDCG vs F-measure
Some models perform better on the NDCG scale vs the KNN scale. This means that there is merit in considering a combination of the different models

4.2 Combined Model Analysis

According to the results in Table 2, we can observe that the combination of various models helps us to get better overall recommendations. The values in the table indicate the performance of each

Table 2: Evaluation Metrics of Combined Models

KNN	NMF	SVD++	RMSE	MAE	Precision	Recall	F-Measure	NDCG
0.5	0.5	0	0.906	0.696	0.998	0.368	0.538	0.982
0.5	0.25	0.25	0.885	0.681	0.998	0.374	0.54	0.989
0.7	0	0.3	0.865	0.668	0.783	0.354	0.488	0.98
0.5	0	0.5	0.864	0.666	0.795	0.232	0.36	0.99

model with different hyperparameters. The best performing model according to the metrics used is the combination of KNN (Pearson Baseline) and SVD++ with a weight of 0.5 each, giving us a NDCG score of 0.99

5 CONCLUSIONS AND FUTURE WORK

When it comes to recommending movies to users, different algorithms have different strengths and limitations. Content-based methods work well when a user has limited ratings and can provide decent recommendations based on the features of the movies they have already rated. However, collaborative filtering and latent factor methods are better at capturing user-related features and can generate more accurate recommendations compared to other algorithms.

To achieve better overall performance, we combined different models and balance the errors and accuracy measures. For example, a combination of KNN (Pearson baseline) and SVD gives a better NDCG score but lower F-measure, while a combination of KNN and NMF gives higher F-measure but lower NDCG score. Depending on the specific use case, we can choose to use either of these models or a more balanced model.

To further enhance our recommendations, we also include popularity-based recommendations. These recommendations take into account the collective activity of users, such as average rating, number of views, likes, and watchlist additions, and provide a crowd-sourced perspective that is not dependent on any particular user's rating. This approach helps address the cold start problem for new users who have not yet rated any movies.

REFERENCES

(1) *Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014.* Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In Proceedings of the 37th international ACM SIGIR conference on Research development in information retrieval (SIGIR '14). *Association for Computing Machinery, New York, NY, USA, 83–92.*

(2) *Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang.* 2021. Neural Collaborative Reasoning. In Proceedings of the Web Conference 2021 (WWW '21). *Association for Computing Machinery, New York, NY, USA, 1516–1527.* https://doi.org/10.1145/3442381.3449973

(3) *Kumar, Manoj Yadav, Dharmendra Singh, Ankur Kr, Vijay.(2015).* A Movie Recommender System: MOVREC. *International Journal of Computer Applications. 124. 7-11. 10.5120/ijca2015904111.*

(4) C. M. Wu, D. Garg and U. Bhandary, "Movie Recommendation System Using Collaborative Filtering," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), 2018, pp. 11-15, doi: 10.1109/ICSESS.2018.8663822.

(5) A. Adomavicius G., Tuzhilin. [n.d.]. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions.. In Regional Conference Series in Mathematics.

(6) Errico J. H. Beek P. V. Sezan M. I. Ferman, A. M. 2002. Content-based filtering and personalization using structured metadata. In Regional Conference Series in Mathematics.

(7) Nicolas Hug. 2017. Surprise, a Python library for recommender systems. http://surpriselib.com.