MovieLens Recommender System

Arunaabh Pant, Vipul Gharde, Tanu Batra
Rutgers University
ap1933,vig4,tb686@scarletmail.rutgers.edu

ABSTRACT

This paper discusses the development of a recommender system using collaborative filtering models to provide personalized recommendations to users based on their preferences and behavior. The study evaluates the performance of different collaborative filtering techniques using the MovieLens dataset to recommend movies to users. The paper presents a comparative analysis of the methodologies used in this study. The results of this research can be useful for improving user experience and engagement on e-commerce, online marketplaces, social media, and entertainment platforms that use recommender systems.

KEYWORDS

recommender systems, collaborative filtering, MovieLens, KNN, SVD

1 INTRODUCTION

The use of recommender systems has become increasingly popular in recent years as they assist users in finding relevant products or items based on their preferences and needs, thereby reducing the time and effort required to search for them. In addition, recommender systems provide organizations with benefits such as increasing customer satisfaction, improving customer retention, and boosting sales and revenue. Notably, various companies such as Netflix, Amazon, and Spotify rely heavily on recommender systems to enhance user experience and engagement.

The prediction of user ratings and recommendations is a key component of building effective recommender systems. However, there are numerous ways to develop recommender systems, and the approach used should be tailored to the specific needs of the user. Additionally, there are several techniques available to achieve the same objective, making it necessary to determine the optimal approach to satisfy the user's requirements.

In this paper, we analyze the MovieLens dataset and employ the collaborative filtering approach to construct a robust recommender system. As multiple methodologies exist for building collaborative filtering recommender systems, we conduct a comparative study to determine the best approach for evaluating a system. The study aims to provide insights into the effectiveness of various methods for building recommender systems and assist in the development of more efficient and effective systems.

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

Recommender systems have been extensively studied in the field of information retrieval and machine learning. Collaborative Filtering (CF) and Content-Based Filtering (CBF) are the two major classes of recommender systems that have been extensively studied in the literature. Collaborative Filtering models such as Matrix Factorization (MF) [4] and k-Nearest Neighbors (KNN) have been shown to be effective in recommendation tasks. Matrix Factorization is a widely used collaborative filtering algorithm that maps users and items into latent factor space and predicts their ratings based on the inner product of their vectors. Similarly, KNN-based approaches predict user-item ratings based on the similarity between users or items.

Content-Based Filtering (CBF) is another class of recommender systems that relies on the features of items to generate recommendations. This approach is based on the idea that users prefer items that are similar to the ones they have liked in the past. CBF has been shown to be effective in cases where user-item interactions are sparse or where the content of items is rich and informative.

Hybrid approaches, which combine CF and CBF methods, have also been proposed in the literature. Hybrid recommender systems are designed to overcome the limitations of individual methods and provide more accurate and personalized recommendations. One popular hybrid approach is the weighted hybrid method, which combines the predictions of multiple recommenders using a linear combination of their scores.

In recent years, deep learning-based approaches have gained popularity in the field of recommender systems. Deep learning models, such as Neural Collaborative Filtering (NCF) [2], have been shown to be effective in generating recommendations based on user-item interactions. NCF uses multi-layer perceptron (MLP) to model the non-linear interactions between users and items, and it has been shown to outperform traditional CF models in terms of accuracy and scalability.

Overall, the literature shows that there is still significant room for improvement in the field of recommender systems. While existing methods have shown promise, there is still a need for more accurate, scalable, and personalized recommendation algorithms that can handle the challenges posed by modern datasets.

3 DATASET

The dataset used in this study is the *ml-latest-small* dataset from MovieLens [1], which includes the *ratings.csv*, *movies.csv*, *links.csv*, and *tags.csv* files. This dataset is a subset of the full MovieLens dataset, which is a popular benchmark dataset for recommender systems. For this study, only the *movies.csv* and *ratings.csv* files were used. The *ml-latest-small* dataset contains 100,000 movie ratings from 610 users on 9,742 movies. The ratings range from 0.5 to 5 stars and were collected between January 09, 1995, and October 16, 2016.

File Name	Columns
links.csv	movieId, imdbId, tmdbId
movies.csv	movieId, title, genres
ratings.csv	userId, movieId, rating, timestamp
tags.csv	userId, movieId, tag, timestamp

Table 1: Schema of the MovieLens dataset

The *ratings.csv* file contains the ratings given by users to movies, with each row representing a single rating and including the user ID, movie ID, rating, and timestamp. The *movies.csv* file contains information about each movie, including the movie ID, title and genres. The *links.csv* file provides links to external sources for each movie, such as the movie's IMDB ID and TMDb ID. Lastly, the *tags.csv* file contains tags applied by users to movies, with each row representing a single tag and including the user ID, movie ID, tag, and timestamp.

4 DATA ANALYSIS

The data was cleaned to ensure that it was suitable for analysis and use in the development of a recommender system. To achieve this, we took the following steps:

- Checked for and removed any null values present in the dataset. Fortunately, there were no null values.
- Extracted the year from the movie titles, removing entries where the year could not be obtained.
- Extracted the genres, removing entries that had "(no genres listed)" in the genres field.
- Dropped the timestamp columns, as they were not useful for analysis or recommender systems.

To analyze the dataset, we explored the parameters of year, title, genres, and ratings. It was important to consider not only the relationships between these parameters but also whether the data was sufficient to support our findings as legitimate and not mere outliers.

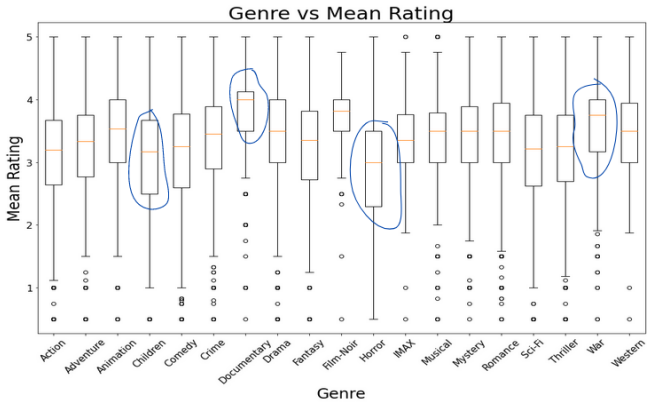

Figure 1: Mean Ratings of Different Movie Genres

For example, Figure 1 shows that genres such as documentaries and war films were well-received by users, while genres such as horror and children's films were not. However, examining the frequency data in Figure 2 shows that the frequencies of children's films and war films were too low to provide conclusive evidence of how well they were received by users. This is because they may simply be outliers, and the trend may change if we have more data.

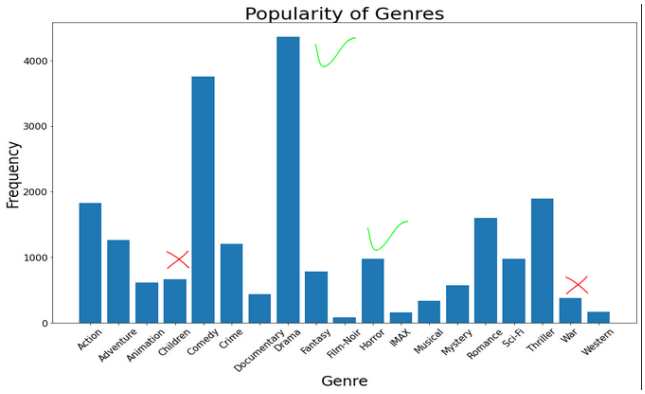

Figure 2: Popularity of Movie Genres by Frequency

Another observation we made was that most of the ratings were above 3 or 3.5, as shown in Figure 3. This indicates that the dataset had been manipulated for use in recommender systems. A dataset with a higher volume of high ratings will be slightly biased towards positive ratings, which is ideal for a recommender system. This means that users will mostly be recommended movies with higher ratings, which is exactly what we want from a recommender system.

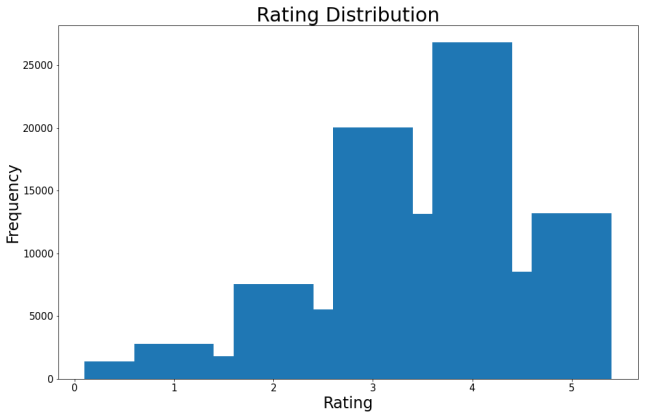

Figure 3: Rating Distribution of Movies

5 MODEL EVALUATION

Several collaborative filtering models can be used to build a recommender system, and selecting the best one for a particular use case can be challenging. In this study, we evaluated and compared various collaborative filtering models to find the most suitable one for our use case. We used the *surprise* [3] library in Python, which provides a set of algorithms for building recommender systems. The models that we considered in this study are:

(1) **BaselineOnly**

It is an algorithm predicting the baseline estimate for a given user and item, formulated as:

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$$

If user u is unknown, then the bias b_u is assumed to be zero. The same applies for item i with b_i.

(2) **KNNBasic**

It is a basic collaborative filtering algorithm, formulated as:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

(3) **KNNWithMeans**

It is a basic collaborative filtering algorithm, taking into account the mean ratings of each user, formulated as:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

(4) **KNNWithZScore**

It is a basic collaborative filtering algorithm, taking into account the z-score normalization of each user, formulated as:

$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - \mu_v)/\sigma_v}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

If σ is 0, then the overall sigma is used.

(5) **KNNBaseline**

It is a basic collaborative filtering algorithm taking into account a baseline rating, formulated as:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

(6) **SVD**

It is the famous SVD algorithm, as popularized by Simon Funk during the Netflix Prize. When baselines are not used, this is equivalent to Probabilistic Matrix Factorization. It is formulated as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

If the user u is unknown, then the bias b_u and the factors p_u are assumed to be zero. The same applies for item i with b_i and q_i.

(7) **SlopeOne**

It is a simple yet accurate collaborative filtering algorithm, formulated as:

$$\hat{r}_{ui} = \mu_u + \frac{1}{|R_i(u)|} \sum_{j \in R_i(u)} dev(i,j),$$

where $R_i(u)$ is the set of relevant items, i.e. the set of items j rated by u that also have at least one common user with i. $\text{dev}(i,j)$ is defined as the average difference between the ratings of i and those of j:

$$dev(i,j) = \frac{1}{|U_{ij}|} \sum_{u \in U_{ij}} r_{ui} - r_{uj}$$

(8) **CoClustering**

It is a collaborative filtering algorithm based on co-clustering. Users and items are assigned some clusters C_u, C_i, and some co-clusters C_{ui}. It is formulated as:

$$\hat{r}_{ui} = \overline{C_{ui}} + (\mu_u - \overline{C_u}) + (\mu_i - \overline{C_i}),$$

where $\overline{C_{ui}}$ is the average rating of co-cluster C_{ui}, $\overline{C_u}$ is the average rating of u's cluster, and $\overline{C_i}$ is the average rating of i's cluster. If the user is unknown, the prediction is $\hat{r}_{ui} = \mu_i$. If the item is unknown, the prediction is $\hat{r}_{ui} = \mu_u$. If both the user and the item are unknown, the prediction is $\hat{r}_{ui} = \mu$.

Clusters are assigned using a straightforward optimization method, much like k-means.

Algorithm	Mean RMSE	Mean MAE
BaselineOnly	0.872201	0.672369
SVD	0.873066	0.670550
KNNBaseline	0.874612	0.668504
KNNWithZScore	0.894428	0.678830
KNNWithMeans	0.897073	0.685302
SlopeOne	0.899393	0.687146
CoClustering	0.944743	0.731583
KNNBasic	0.945871	0.724686

Table 2: Model Evaluations on Rating Prediction RMSE and MAE

From the above,
u and v are users,
$\hat{r}_{ui}$ is the estimated rating of user u for item i,
r_{ui} is the true rating of user u for item i ,
b_{ui} is the baseline rating of user u for item i,
μ is the mean of all ratings,
b_u is the user bias,
b_i is the item bias,
$N_i^k(u)$ are the k nearest neighbors of user u that have rated item i,
μ_u is the mean rating of user u,
σ_u is the standard deviation of the ratings of user u.

All of the above models were built using their default parameters.

To determine the best performing model, we first trained all the models using 5-Fold cross-validation and evaluated their mean RMSE (root mean square error) and MAE (mean absolute error) for rating prediction.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (r_{ui} - \hat{r}_{ui})^2}$$

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |r_{ui} - \hat{r}_{ui}|$$

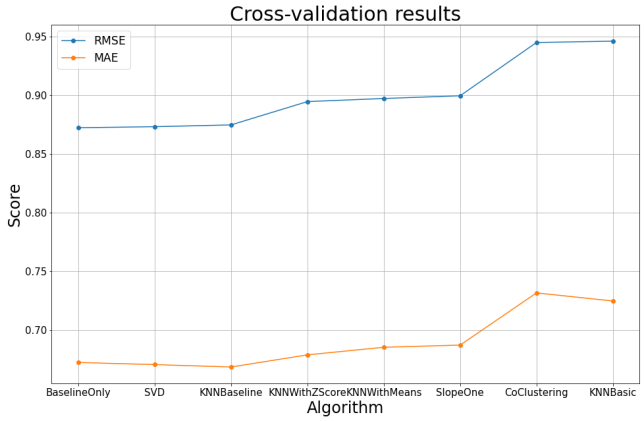

Figure 4: Plot of RMSE and MAE Results of Different Models

Algorithm	Precision@10	Recall@10	F1 Score
BaselineOnly	0.562327	0.256928	0.352679
KNNBasic	0.813299	0.414142	0.548792
KNNWithMeans	0.703773	0.397760	0.508226
KNNWithZScore	0.726752	0.409591	0.523872
KNNBaseline	0.759915	0.392979	0.518015
SVD	0.715530	0.350241	0.470264
SlopeOne	0.746035	0.386291	0.508984
CoClustering	0.618293	0.354801	0.450861

Table 3: Model Evaluations on Rating Prediction RMSE and MAE

As can be seen from Table 2 and Figure 4, among the models evaluated, BaselineOnly, SVD, and KNNBaseline performed the best, with a mean RMSE of around 0.87 and a mean MAE of around 0.67.

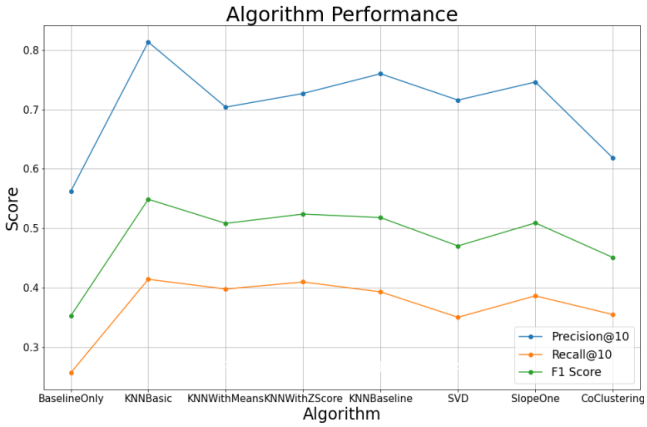

Figure 5: Plot of Precision@k, Recall@k, and F1 Score Results of Different Models, k = 10

We then evaluated the models for recommender Precision@k, Recall@k, and F1 Score, using a rating threshold of 4 and a value of k set to 10.

$$Precision@k = \frac{|Recommended\ items\ that\ are\ relevant|}{|Recommended\ items|}$$

$$Recall@k = \frac{|Recommended\ items\ that\ are\ relevant|}{|Relevant\ items|}$$

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

An item is considered relevant if its true rating r_{ui} is greater than a given threshold. An item is considered recommended if its estimated rating $\hat{r}_{ui}$ is greater than the threshold, and if it is among the k highest estimated ratings.

In the edge cases where division by zero occurs, Precision@k and Recall@k values are undefined. As a convention, their values are set to 0 in such cases.

The models were retrained using 5-Fold cross-validation. It was observed that while the BaselineOnly model had the best RMSE and MAE for rating prediction, it performed poorly in terms of Precision@k, Recall@k, and F1 score for recommendation, as shown in Table 3 and Figure 5.

6 RESULTS

In our evaluation of various models for the recommender system, we observed that each model has a different trade-off between evaluation metrics such as RMSE, MAE, Precision@k, Recall@k, and F1 score. In order to balance these metrics and obtain optimal results, we chose the KNNBaseline model as our final recommendation model.

```
# Get top 10 recommendations for user_id 1
user_id = 1
```

```
top_n_df = get_top_n(user_id, predictions, movies_df, n=10)
top_n_df
```

	movieId	title	genres
0	318	Shawshank Redemption, The (1994)	Crime|Drama
1	131724	The Jinx: The Life and Deaths of Robert Durst ...	Documentary
2	720	Wallace & Gromit: The Best of Aardman Animatio...	Adventure|Animation|Comedy
3	1272	Patton (1970)	Drama|War
4	5746	Galaxy of Terror (Quest) (1981)	Action|Horror|Mystery|Sci-Fi
5	5764	Looker (1981)	Drama|Horror|Sci-Fi|Thriller
6	6835	Alien Contamination (1980)	Action|Horror|Sci-Fi
7	7899	Master of the Flying Guillotine (Du bi quan wa...	Action
8	898	Philadelphia Story, The (1940)	Comedy|Drama|Romance
9	1719	Sweet Hereafter, The (1997)	Drama

Figure 6: Top 10 Movie Recommendations for User ID 1

To generate movie recommendations, the KNNBaseline model was retrained on the entire dataset, and predictions were made

for unrated movies of each user. The predicted ratings were then sorted in descending order and the top 10 highest predictions were selected as the recommended movies. Figure 6 displays an example of these movie recommendations.

7 CONCLUSIONS AND FUTURE WORK

In conclusion, recommender systems are a valuable tool for both users and companies, offering personalized recommendations and increased profits. Through extensive hyperparameter tuning, existing models can be optimized for scalability, performance, and efficiency.

Future work in the field of recommender systems can focus on exploring novel hybrid techniques and deep learning models to address existing issues with individual methods and to improve the accuracy and efficiency of these systems. Additionally, further research can be conducted on the development of more scalable and efficient algorithms to handle increasingly large datasets. Finally, more attention can be given to the ethical implications of these systems, such as ensuring fairness and avoiding bias in recommendations.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Prof. Yongfeng Zhang for his valuable guidance and support throughout the duration of this project. His expertise and insights were instrumental in helping us navigate the complexities of recommender systems and machine learning, and we are truly grateful for the time and effort he invested in us.

REFERENCES

[1] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (dec 2015), 19 pages. DOI: http://dx.doi.org/10.1145/2827872

[2] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. DOI: http://dx.doi.org/10.1145/3038912.3052569

[3] Nicolas Hug. 2020. Surprise: A Python library for recommender systems. *Journal of Open Source Software* 5, 52 (2020), 2174. DOI: http://dx.doi.org/10.21105/joss.02174

[4] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37. DOI: http://dx.doi.org/10.1109/MC.2009.263