Movie Recommender System

Team 4

Zain Ahmed, 1959323
Progha Labani Das, 1871821
Bindhu Jangollu Keshavareddy, 1904401
Shivam Suchak, 1958823
Saloni Wade, 1936657

University of Mannheim, Germany

Abstract. Movie Recommender systems are applications that help the users to find movies based on their preferences and behavior. These recommendations can be performed using user's ratings for other movies, other user information such as age, occupation etc., also with the help of movie metadata. This can be done by taking into consideration similarity between users or movies or both. With a great deal of recommendation systems out there that are already performing very well in recommending movies to users, we through this project tried to compare two different approaches to recommend movies. We first tried to recommend movies using some traditional approaches like Collaborative filtering, Content-based recommendation and also Hybrid Recommender. We then tried to use the best model from these approaches and compare it with our second approach, which was to use Graph Neural Networks(GNNs) to recommend movies. We explicitly used the LightGCN model as our core model to build the recommender system. For this project, we used MovieLens 1M dataset that had information about ratings, users and movies, and employed various approaches to understand and explore the data. We then implemented different preprocessing methods to determine one preprocessing method that optimized the results of the recommender. After preprocessing, we tried to build the model using all the aforementioned approaches. We chose Root Mean Squared Error(RMSE) and Normalized Discounted Cummulative Gain(nDCG@k) as our major evaluation metrics.

Keywords: Recommender Systems · Graph Neural Networks · Movies

1 Introduction

1.1 Problem / Task Formulation

The rapid growth of data collection has led to a new era of information. Data is being used to create more efficient systems and this is where Recommendation Systems come into play. Recommendation Systems are a type of information

filtering systems as they improve the quality of search results and provide items that are more relevant to the search item or are related to the search history of the user. They are used to predict the rating or preference that a user would give to an item.

These days, people prefer not to spend their leisure time making choices about which movies to watch. Instead, they rely on others' judgment to make the decisions for them. Given the vast number of available options and the constraints of time, it challenging to make a selection and yearn for personalized movie recommendations. This is where a recommender system comes into play, as we strive to create one that can assist us in suggesting movies tailored specifically to each individual's preferences.

1.2 Research Objectives

Our main goal is to provide movie enthusiasts with a more personalized and enriching experience in discovering new movies that align with their individual preferences. By leveraging the MovieLens 1M dataset, we aim to develop a robust movie recommendation system that by comparing various models that tailors movie suggestions to the unique tastes and interests of each user.

One of the key objectives of our system is to create a sense of unity among movie enthusiasts by incorporating community ratings. By considering the ratings and feedback from a large community of movie viewers, we can generate recommendations that resonate with the collective opinions and interests of the community. This not only promotes engagement and interaction among movie lovers but also allows users to explore popular and highly regarded movies that have garnered widespread acclaim within the community. Our project aims to develop a movie recommendation system that provides personalized recommendations to users, while also fostering a sense of community. By achieving these objectives, we aim to create a more engaging and satisfying movie-watching experience for users, ultimately enhancing the overall movie ecosystem.

2 Methodology

The overall purpose of this project was to predict which movie the user might like based on his ratings for the other movies. As a result, this was a prediction problem. To accomplish this, we used the three datasets available with different approaches and with and without hyperparameter tuning.

To begin, we utilized surprise package, a Python scikit for buildng and analyzing recommender systems that deal with explicit rating data. Without any hyperparameter adjustment, we established pipelines for each of the recommender system models listed below and examined the results on the test sets. Later, hyperparameter tuning was performed on them using grid search and Engine functions from surprise package.

2.1 Collaborative Filtering

Collaborative filtering uses similarities between users and items simultaneously to provide recommendations. This allows for serendipitous recommendations, that is, collaborative filtering models can recommend an item to user A based on the interests of a similar user B. Furthermore, the embeddings can be learned automatically, without relying on hand-engineering of features.

2.1.1 KNN

KNN is a type of collaborative filtering algorithm that uses the k-nearest neighbors technique to find the most similar users or items to a given user or item. It then uses the ratings or preferences of these neighbors to make recommendations.

2.1.2 SVD

SVD is a matrix factorization technique that capture the underlying patterns or preferences of users and items based on latent factors. It decomposes the user-item rating matrix into three smaller matrices: U, S and V, where U represents the user features matrix, S represents the diagonal matrix of singular values (essentially weights), and V represents the item features matrix. The idea is that the user-item rating matrix can be approximated by multiplying these three matrices together.

2.1.3 Slope One

Slope one is a type of item-based collaborative filtering algorithm that uses the average difference between ratings of items to make predictions. The idea is that for each pair of items, we calculate the average difference between the ratings given by the users who rated both items.

2.2 Content Based Filtering

Content-based filtering is a type of recommender system that attempts to guess what a user may like based on that user's activity. Content-based filtering makes recommendations by using keywords and attributes assigned to objects in a database (e.g., items in an online marketplace) and matching them to a user profile. The user profile is created based on data derived from a user's actions, such as purchases, ratings (likes and dislikes), downloads, items searched for on a website and/or placed in a cart, and clicks on product links.

2.2.1 KNNBasic

KNNBasic is a type of content-based filtering algorithm that uses the k-nearest neighbors technique to find the most similar items to a given item based on their features or attributes. The idea is that for each movie, we extract its features or attributes and represent them as a vector. Then, we calculate the similarity between two movies by using a similarity measure, such as cosine similarity, which measures the angle between two vectors.

2.3 Hybrid Based Recommender

A hybrid recommendation system is a special type of recommendation system which can be considered as the combination of the content and collaborative filtering method. Combining collaborative and content-based filtering together may help in overcoming the shortcoming we are facing at using them separately and also can be more effective in some cases. In our project, we used weighted average of the results of content and collaborative approaches to build the hybrid recommender system.

2.4 Light GCN

LightGCN is a variant of Graph Convolutional Network (GCN) designed for recommender systems. It tackles sparsity by leveraging the bipartite graph structure of user-item interactions. It simplifies aggregation using the LightGCN Layer, which involves linear transformations and element-wise summation. LightGCN iteratively updates user and item embeddings, capturing latent preferences. In the MovieLens dataset, the bipartite graph connects users to movies they have interacted with. LightGCN learns embeddings and employs collaborative filtering to make personalized recommendations, even for uninteracted movies.

3 Experimental Setting

3.1 Structure and Statistics of the Data Set

For our problem, we will be using MovieLens 1M Dataset. The data that was originally obtained was in the form of a DAT File. This was converted manually into a CSV file to arrive at an input that could be loaded into a Pandas DataFrame effortlessly. In other words, the dataset we have in our hands is already relatively clean. We will however attempt to learn more about our features and perform appropriate wrangling steps to arrive at a form that is more suitable for analysis. The dataset contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000. We have three csv files - Ratings, Users and Movies.

1. Ratings File Description:
   All ratings are contained in the file "ratings.dat" and are in the following format:
   UserID::MovieID::Rating::Timestamp
   - UserIDs range between 1 and 6040
   - MovieIDs range between 1 and 3952
   - Ratings are made on a 5-star scale (whole-star ratings only)
   - Timestamp is represented in seconds since the epoch as returned by time(2)
   - Each user has at least 20 ratings

2. Users File Description:
   User information is in the file "users.dat" and is in the following format:
   UserID::Gender::Age::Occupation::Zip-code
   All demographic information is provided voluntarily by the users and is not checked for accuracy. Only users who have provided some demographic information is included in this data set.
   - Gender is denoted by a "M" for male and "F" for female
   - Age is chosen from the following ranges:
     - 1: "Under 18"
     - 18: "18-24"
     - 25: "25-34"
     - 35: "35-44"
     - 45: "45-49"
     - 50: "50-55"
     - 56: "56+"
   - Occupation is chosen from the following choices:
     - 0: "other" or not specified
     - 1: "academic/educator"
     - 2: "artist"
     - 3: "clerical/admin"
     - 4: "college/grad student"
     - 5: "customer service"
     - 6: "doctor/health care"
     - 7: "executive/managerial"
     - 8: "farmer"
     - 9: "homemaker"
     - 10: "K-12 student"
     - 11: "lawyer"
     - 12: "programmer"
     - 13: "retired"
     - 14: "sales/marketing"
     - 15: "scientist"
     - 16: "self-employed"
     - 17: "technician/engineer"
     - 18: "tradesman/craftsman"
     - 19: "unemployed"
     - 20: "writer"

3. Movies File Description:
   Movie information is in the file "movies.dat" and is in the following format:
   MovieID::Title::Genres
   - Titles are identical to titles provided by the IMDB (including year of release)
   - Genres are pipe-separated

The number of unique users in the dataset are 6040 and the number of unique movies are 3706. The users have provided ratings on a scale of 1 to 5, with 1 representing the lowest and 5 representing the highest score. The average movie rating was found to be around 3.8 indicating a positive overall sentiment.The distribution of ratings is depicted in the figure below.

Rating	Count
1	56.174k
2	107.557k
3	261.197k
4	348.971k
5	226.31k

Table 1. Distribution of Ratings

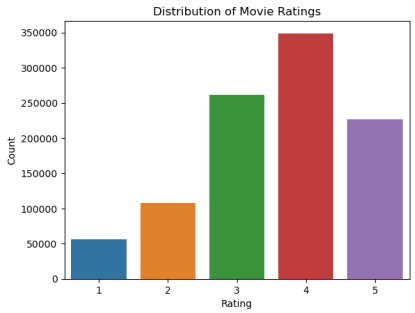

Table 2. Distribution of Movie Ratings

We also tried to depict the distribution of genres in the movies dataset. There are total 18 genres in the dataset namely, Drama, Comedy, Action, Thriller, Romance, Horror, Adventure, Sci-Fi, Children, Crime, War, Documentary, Musical, Mystery, Animation, Western, Fantasy, and Film Noir. The most common movie genres in the dataset was Drama and Comedy.

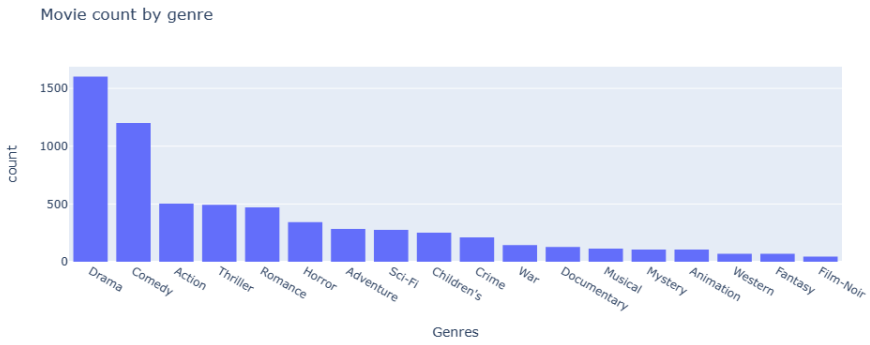

Fig. 1. Movie Genre Distribution

3.2 Data Preprocessing

In general, real-world data is not always complete and error-free. The data set lacked some attribute values of interest, and some attributes were not ready for model preparation. A more appropriate data format generation for the model training involves various treatments.

Missing Value Treatments
All the three datasets had no missing values.

Remove unnecessary features
'Timestamp' feature from ratings dataset was removed as we did not find appropriate use for it.

Preprocess movies dataset
The Title and Genre columns of movies dataset were preprocessed using regex to remove non-aplanumeric characters to be able to apply TfIdf model on it.

Preprocess users dataset
The Gender and Occupation columns of user dataset were also preprocessed in a similar way using regex. In addition, the Occupation column had categorical values with numeric type. We converted them to Categorical values with string type.

Train Test Split
Lastly, the ratings dataset is split into a training set and a testset using an 75:25 split.

3.3 Collaborative Filtering

For the collaborative filtering, we are using 'ratings.csv'. We tried two different models with default parameters. Then we performed hyperparameter tuning to get the best model.Engine, an inbuilt function from surprise package was used to get the best model for Collaborative approach. This function runs all the models with different hyperparameter settings on the entire ratings data and returns the best model, parameters and score.

3.3.1 KNN

We used KNN as one of our baseline model to get the scores to compare it with more complex models. We used a k value of 10 to train the model and 'pearson' as our similarity metric.

3.2.2 SVD

We then tried SVD as our second collaborative approach with random parameters. We instantiated the SVD algorithm from the Surprise library, setting the hyperparameters such as the number of epochs, learning rate, and regularization The best combination of parameters was 10 epochs, learning rate equal to 0.005 and regularization rate of 0.4.

3.2.3 Slope One

Finally, we performed hyperparameter tuning using Engine, an inbulit function from surprise package to get the best model with best parameters and score. This function returned Slope One as the best model for our dataset with no parameters and with the best score out of all the other models that we tried.

3.4 Content Based Filtering

For the content based filtering, we are using all the three datasets - 'ratings.csv', 'movies.csv' and 'users.csv'. For content based filtering, we are using just one model, KNNBasic. We first implemented the model with random parameters(k = 10, user_based = True and similarity metric = cosine). We then performed hyperparameter using Gridsearch from the surprise package. Grid search is a strategy that searches over a specified parameter values with successive halving and starts evaluating all the candidates with a small amount of resources and iteratively selects the best candidates, using more and more resources, also combined with a repeated Stratified K fold, which means that the k-fold cross-validation process is repeated multiple times and report the mean performance across all folds and all repeats and can help reducing the bias in model's estimated performance. For k, we used the values 3, 5, 10, 20. We tried just one similarity metric, cosine and for user_based, we tried the condition with just True. The model returned k=20, similarity metric=cosine and user_based=True as best parameters. We then applied these parameters to our KNNBasic models and the scores improved significantly, but they were not better than collaborative filtering.

3.5 Hybrid Recommender

Since the Collaborative and Content based systems were not performing well as separate models, we combined the two models to get best performance. We used the weighted average approach and the scores were much better than the other two models, and also the recommendations looked relevant with better rankings.

3.6 LightGCN

3.6.1. Data Loading and Preprocessing: Importing the necessary libraries and reading the movie ratings data from the 1m MovieLens dataset. We filter

out low-rated movies (ratings below 3) to ensure accurate predictions of user preferences for future movie choices, we selectively consider only the interactions with high ratings, disregarding lower-rated interactions. Our goal with this approach would also contribute to faster model performance, particularly in the case of LightGCN. By reducing the number of interactions and considering only high-rated movies, the computational load can be lightened, potentially resulting in quicker execution of the model.

3.6.2. Data Encoding: We encode the user and movie IDs using label encoding to convert them into numerical indices by assigning integer indices to the unique values of the variables using the fit_transform method and storing them in new columns. This step is important for constructing the edge indices for the training graph using the encoded data, forming a symmetric adjacency matrix representing connections between users and movies.

3.6.3. Model Definition: The LightGCN model is a simplified version of the Graph Convolutional Network (GCN) layer designed for collaborative filtering. For our purposes, we defined the latent dimension of 64 which consists of 3 layers. The model performs message propagation through the graph structure, updating user and item embeddings in each layer.

3.6.4. Evaluation Metrics: We first calculated the relevance scores for user-item pairs by multiplying user and item embeddings. Then, created a dense tensor represents all user-item interactions and masked out the training interactions from the relevance scores. Next, the top K-scoring items for each user were determined. Additionally, we measured the overlap between recommended items and held-out interactions to compute metrics like recall, precision, nDCG@k, and MAP.

3.6.5. Training and Evaluation: We implemented a training and evaluation loop for our model. Over a total of 10 epochs, we sampled mini-batches of user-movie interactions with a batch size of 1024. The BPR (Bayesian Personalized Ranking) loss and regularization loss were computed during training. We used the Adam optimizer with a learning rate of 0.05 and a decay rate of 0.01. After training, we evaluated the model's performance by calculating recall and precision metrics for the top-K recommended items (K = 20).

4 Evaluation and Discussion of Results

4.1 Evaluation Framework

As previously stated, the dataset was split into train and test sets using a fixed train test split of 75:25. We evaluated all our models using different evaluation metrics like Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, Precision, Recall, F1, Accuracy, Average Precision, Precision@k

and nDCG@K. We will be considering RMSE and nDCG@k to compare the performance of the models in traditional approaches whereas, precision@k and nDCG@k to compare the traditional approach with the GNN approach.

4.2 Results

4.2.1 Traditional Approach Comparison

We are going to implement 2 different approaches to evaluate the results. For our first approach, we are going to compare the three traditional approaches to see which model is performing best. Among the three models KNNBasic, SVD and Slope One that we used for Collaborative filtering, we can see from Table 3 that Slope One is outperforming the other two models, with RMSE and nDCG@k of 0.908 and 0.889 respectively. Slope One was the best model that we got after hyperparameter tuning. For content based, we tried KNNBasic with random parameters and with hyperparameter tuning. Training the model again with the obtained hyperparameters gave better results with RMSE and nDCG@k scores 1.066 and 0.850, which is evident from Table 4 and also better recommendations compared to the one with random parameters. But, ultimately, our best results for this dataset were obtained by using Hybrid model which used the weighted average of collaborative filtering and content based model that were obtained after tuning the parameters with an RMSE of 0.958 and an nDCG@k of 0.909.

Recommender	RMSE	nDCG@k
KNN Basic	0.9649	0.888
SVD	0.9302	0.888
Slope One	0.908	0.889

Table 3. Collaborative Filtering

Recommender	RMSE	nDCG@k
KNN Basic Without Hyperparameter Tuning	1.131	0.783
KNN Basic With Hyperparameter Tuning	1.066	0.850

Table 4. Content Based Filtering

Recommender	RMSE	nDCG@k	P@k
Hybrid	0.958	0.909	0.708

Table 5. Hybrid Recommender

4.2.2 Hybrid Vs LightGCN

For our second approach, we are going to compare the hybrid model with the LightGCN as we concluded that it was the best performing model among the traditional models. The LightGCN model yielded moderate performance with precision, recall, nDCG@k, and MAP values of 0.1668, 0.1498, 0.2213, and 0.0605

respectively. The train data loss on the last epoch was 0.6164, indicating a relatively high error rate during training. The precision@k and nDCG@k for hybrid model was 0.708 and 0.909 respectively, which was way better than the score of LightGCN, whose best scores were 0.1668 and 0.2213 over 10 epochs, respectively. Also, leveraging the two models and available user and movie datasets, personalized movie recommendations were generated for user 196. The recommendations from hybrid model was more relevant to the movies that the user previously rated. It as clearly evident that hybrid model from traditional approach outperformed the LightGCN in terms of both the scores and recommendations for our data.

Recommender	nDCG@k	P@k
LightGCN	0.2213	0.1668

Table 6. LightGCN

4.2.2.1 Movie Recommendations

Table 7 depicts highly(5) rated movies my user 196. Tables 8 and 9 are the recommendations from LightGCN and Hybrid model respectively. It is clearly evident that the Hybrid model is recommending more relevant movies to the movies that the user rated highly previously. We can see that the user 196 highly rated movies with genres comedy and thriller. Hybrid model mostly recommends movies with the same genre while the movies recommended by LightGCN are not relevant in terms of genres. This gives us a conclusion that Hybrid model is performing better in terms of recommendations.

Title	**Genre**
Usual Suspects, The (1995)	Crime—Thriller
L.A. Confidential (1997)	Crime—Film-Noir—Mystery—Thriller
Sixth Sense, The (1999)	Thriller
What Ever Happened to Baby Jane? (1962)	Drama—Thriller
Almost Famous (2000)	Comedy—Drama
Meet the Parents (2000)	Comedy
Godfather: Part II, The (1974)	Action—Crime—Drama
Four Weddings and a Funeral (1994)	Comedy—Romance
American Beauty (1999)	Comedy—Drama
High Fidelity (2000)	Comedy
Rear Window (1954)	Mystery—Thriller

Table 7. Movie Rated Highly(5) by User 196

Title	Genre
House II: The Second Story (1987)	Comedy—Horror
Arsenic and Old Lace (1944)	Comedy—Mystery—Thriller
Stand by Me (1986)	Adventure—Comedy—Drama
This Is Spinal Tap (1984)	Comedy—Drama—Musical
Eyes Without a Face (1959)	Horror

Table 8. Recommended Movies for User 196 by LightGCN

Title	Genre
Usual Suspects, The (1995)	Crime—Thriller
American Beauty (1999)	Comedy—Drama
L.A. Confidential (1997)	Crime—Film-Noir—Mystery—Thriller
Almost Famous (2000)	Comedy—Drama
Sixth Sense, The (1999)	Thriller

Table 9. Top 5 Recommendations for User 196 by Hybrid Model

5 Conclusion

Streaming movies online has especially gained a lot of prominence since the Covid-19 pandemic. It is estimated that there were around 40.6 million new Netflix users since the beginning of 2020, when the coronavirus outbreak started to spread globally. With this increasing popularity of online streaming platforms and the competition between different platforms like Netflix, Amazon Prime, Disney Plus, HBO Max and others, there is a necessity for exceptional recommender systems. Even though there are a lot of traditional models that are performing well, we need other approaches that would outperform the traditional approaches. Through this project, we tried to compare the traditional approach of recommender systems to a deep learning approach, LightGCN. But from the results, we can see that the hybrid model from the traditional models is performing better than LightGCN in our case. There might be several reasons for this.

Firstly, the hybrid recommendation system may be able to leverage both user and item features, as well as user-item interactions, to make better predictions, while LightGCN only uses user-item interactions. Secondly, we were not able to perform the hyperparameter tuning for LightGCN model because of insufficient computational resources. So, using default parameters might not have worked well for our model. So, we conclude that traditional approach works well for our dataset. But there is a scope to improve recommendations with LightGCN if we could find a way to use user and movie features and also with better computational resources to do hyperparameter tuning. We confirmed this by applying the same LightGCN model on a smaller dataset with 100k ratings. Performing hyperparameter tuning on this dataset gave better results than using random parameters.

References

1. F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015)
http://dx.doi.org/10.1145/2827872
2. https://www.kaggle.com/code/rounakbanik/the-story-of-film/notebook
3. Derrick Li (2022, Jan 11), Recommender systems with GNNs in PyG
https://medium.com/stanford-cs224w/recommender-systems-with-gnns-in-pyg-d8301178e377
4. Elad Rapaport (2022, Jan 8), MovieLens-1M Deep Dive — Part I, A hands-on recommendation systems tour using the popular benchmark dataset
https://towardsdatascience.com/movielens-1m-deep-dive-part-i-8acfeda1ad4