Comparing Collaborative Filtering-Based Recommender and Hybrid Recommender System

Ajay Dahiya

C0783939@mylambton.ca

Akshita khatri

C0785493@mylambton.ca

Angelica Serrano C0785370@mylambton.ca

Jonatas Aguiar C0790419@mylambton.ca

AML 3204

Instructor: Debashish Roy

Social media analytics

Lambton College – Toronto

Abstract— This paper aims to analyze and compare the performance of the collaborative Filtering and hybrid-based methods in generating movie recommendations according to the user's choice. The collaborative filtering approach uses the rating data, consisting of two main parts: (1) Movie similarity and (2) Movie rating prediction. At the same time, the hybrid-based approach has a content-based add-on to the collaborative filtering-based approach. It uses both the movie and rating data and has four main sections 1) text preprocessing, 2) Term weighting, 3) Movie clustering and 4) Collaborative Filtering based approach. The performance comparisons show that the collaborative filtering approach performs better than the hybrid-based at any Normalized Discounted Cumulative Gain and top-N position in precision.

# 1. INTRODUCTION

A Recommender System (RS) is a machine-learning driven system consisting of techniques and algorithms that can suggest relevant items or products to users. RS becomes one of the most popular marketing strategies to boost sales revenue. They predict future patterns based on past data through a multitude of techniques, including matrix factorization. Seeing good recommendations while browsing a movie provides an immersive movie-watching experience and satisfaction from a customer perspective. It speeds up finding all the preferrable film, gives a rundown of similar interests of movies and genres. Data is essential in building RS. User preferences, Last watched movies and movie ratings are examples of crucial information needed in the Dataset. Recommender systems use three filtering algorithms: (1) Content-based filtering, which utilizes characteristics information; (2) Collaborative filtering, which uses user-item interaction; and (3) Hybrid system, which uses both information. In this paper, Collaborative Filtering and a hybrid-based system are used to compare the best approach among the two. Various information was collected and evaluated against significant factors to produce the best logical output. There are three types of collaborative Filtering: (1) memory-based, which uses user historical data rating to compute the similarity measure between users or items. (2) Model-based, which uses ML algorithms to predict user's rating of unrated items; 3) Neural-based collaborative Filtering. The minimum and maximum ratings present in the data. Embeddings represent each user and each movie in the data. It will be of vector size n that gets fit by the model to capture the interaction of each user/movie.

Diagram

Description automatically generated

*Figure 1: Collaborative Filtering*



*Figure 2: Collaborative filtering using matrix Factorization*

Diagram

Description automatically generated

*Figure 3: Neural-based Collaborative Filtering*

# 2. LITERATURE REVIEW

Recommender Systems (RSs) are software application tools that offer similar items to users [1]. RSs are created to help the users decide on choosing relevant articles that may spark their interest. RSs can give personalized recommendations over general recommendations [2]. They are commonly used for e-commerce platforms to turn browsers into buyers and improve cross-selling among products. There are various recommender systems, as seen in Figure 4. The three essential types are (1) content-based and 2) collaborative-based and Hybrid recommender systems that have two branches: item-based and user-based. Within these two major categories lies multiple techniques or algorithms to provide recommendations.

However, item-based collaborative systems perform better than user-based ones [3]. The most significant difficulty is the cold start problem, where either a new user or item is added to the system, which gets thrown deep down in the list [4]. The knowledge-based approach can be used to tackle the problems above. In today's world, the size of data took an inverse turn. Instead of low amounts, we get large quantities of data, and this sheer volume creates different problems for the recommendation systems [5]. They have several limitations; for example, Content-Based filtering techniques have issues like limited content analysis and sparsity of data [6], collaborative approaches have issues like cold-start, sparsity, and scalability. For these scenarios, the hybrid system can be used, as shown in Figure 5. The hybrid system works by combining two or more filtering techniques in different ways to increase the performance and accuracy of the recommender system has thus been proposed to alleviate these issues [7]. Hybrid systems try to leverage the strength of each method without sacrificing any capability due to inherent weakness [8]. A survey detected that the ratings from the knowledge base improve the efficiency rate of the collaborative Filtering [9]. RS has emerged to work with web applications. It provides customization for customers of e-commerce, promoting one-to-one marketing [10].

Diagram

Description automatically generated

*Figure 4: Types of recommender system*

Diagram

Description automatically generated

*Figure 5: Hybrid-Based recommender system*

3. METHODOLOGY

The ultimate goal of this project is to make a model that performs well for any future data. The training data performs well with the overfitting but performs worse with test data.

To build a collaborative filtering-based recommender system, we have used a matrix factorization algorithm, as shown in Figure 6. To create a hybrid recommender system, Pytorch embedding layers have been used.

Text, letter

Description automatically generated

*Figure 6: Matrix Factorization Algorithm*

Text

Description automatically generated

*Figure 7: Hybrid-Based algorithm*

3.1 MODEL DESIGN

For a collaborative filtering-based recommender system, matrix factorization algorithm is used and SVDs was used to leverage parallel multi-core use. To build a Hybrid-based system Pytorch embedding layer is used.

3.2 HYBRID-BASED SYSTEM ARCHITECTURE

Diagrama

Descrição gerada automaticamente

*Figure 8: Hybrid recommender system design*

Diagram

Description automatically generated

*Figure 9: Data flow Diagram*

3.3 DATASET DESCRIPTION

The Dataset is taken from the Movie Lens dataset collected by the Group Lens Research Project at the University of Minnesota.

This data set consists of:

* 100,000 ratings (1-5) from 943 users on 1682 movies.
* Each user has rated at least 20 movies.

Here are brief descriptions of the data:

* u.data

The entire 'u' data set, 100000 ratings by 943 users on 1682 items. Each user has rated at least 20 movies. Users and items are numbered consecutively from 1. The data is randomly

ordered. This is a tab-separated list of user id | item id | rating | timestamp.

The timestamps are Unix seconds since 1/1/1970 UTC

* u. item

Information about the items (movies) which consist of the column names as:

movie id, movie title, release date, video release date,

IMDb URL, unknown, Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western.

The last 19 fields are the genres. A 1 indicates the movie is of that genre, a 0 means it is not.

Movies can be in several genres at once. The movie ids are the ones used in the u.data data set.

* u. genre dataset

It consists of a list of the genres. The figure 10 shown below shows the Dataset of u. item.

Table

Description automatically generated

*Figure 10: Dataset overview*

3.4 Matrix Factorization

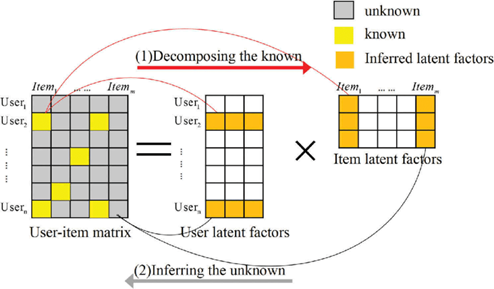
Matrix Factorization is the process of decomposing large matrix into smaller matrices so that they can be more efficiently processed when doing operations on them.

Let’s take the ratingsMatrix as an example it is of size user\*item, we need to break it into two smaller matrices userFeature and itemFeature, so that their dot product produces the original ratingsMatrix. The following equations shows the breakdown, where f is number of latent factors:

ratingsMatrix[user][item] =

sum (userFeature[f][user] \* itemFeature[f][item])

After breaking down we get two matrices, userFeature of size user\*f and itemFeature of size f\*item. More technical procedure is mentioned above (refer figure 11).



*Figure 11: Diagram of Matrix Factorization*

3.5 NEURAL EMBEDDING LAYER

The Neural Embedding Layer is a tensor, a vector of vectors composed of random numbers like a conventional matrix. The number of vectors is the number of rows, and the size of each vector is the number of columns. When instantiating an object with Pytorch, we need to specify the number of vectors (embeddings) and the size of each vector. For example, on the left side of figure 12, we have an object instantiated with five vectors with size 3. This object has five vectors, and each vector has three random values. The right side of the same picture is an example of applying this to the hybrid recommender system. We can create an embedding layer with five vectors corresponding to each user, with three latent features. If we pass a list of users to the embedding layer, this object will return the corresponding vector to each user.

Diagrama

Descrição gerada automaticamente

*Figure 12:*  Latent features generated with embedding layers

3.6 Sentiment score calculation

Vader, a popular sentiment analysis model, was used to calculate the sentiment scores of each tweet extracted from Twitter. Vader's Sentiment Intensity Analyzer library produced a dictionary output that displays sentiment(key) and polarity scores(value). To determine the possible sentiment of each tweet, the weight of the compound can be used as an indicator. Scores between -0.05 to 0.05 can be considered neutral, and anything above 0.05 is positive, otherwise negative.

The average (compound) score for each movie was calculated using the Pandas groupby aggregate mean (agg) function.

3.7 Activation Function of Hybrid Recommender System

The Rectified Linear Unit (ReLU) activation function was used to build the hybrid recommender system. Many neural networks architectures use the ReLU activation functions. Some of the reasons behind this trend are: 1- ReLU are simply calculated when compared to the tanh and sigmoid activation functions: 2 – ReLU can output actual zero values, leading to a desirable property to accelerate learning called sparse representation: 3 – ReLU makes the model optimize better as it is closer to linear behaviour, unlike the tanh and sigmoid functions.

The rectified linear activation function graph can be seen in figure 13, and its formula is denoted by:

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

*Figure 13: ReLU Function*

4. EXPERIMENTS

The Dataset taken from Movie Lens contains 100k movie ratings. In this section, we will throw some light on the experiment design, Dataset preparation from Twitter tweets(taking account of tag words), Evaluation metrics, results and analyses of hyperparameter selection and recommendation models.

4.1 Experiment Design

To create an experimental design for the model's proper functioning, the following steps are being taken as follows.

STEP 1: Data collection uses movie names and makes a tag word for each movie. Then after searching, that particular tag word, we have downloaded the related tweets for that movie. Step 2: tweets cleaning is done. Step 3: Calculate the sentiment score for each tweet and each movie. Step 4: Calculate the average sentiment score for each movie. The model is designed with the help of the python libraries (.ipynb). Then, from scipy. Sparse. linalg import svds.The singular Value Decomposition (SVD) technique is used to decompose a matrix into several component metrics. Vader is used for measuring the sentiment score. Tweepy is used for accessing Twitter API was utilized to extract tweets related to given movie titles. Twitter developer credentials (Access token and Consumer API keys) were initially defined to obtain Twitter function access. Subsequently, Twitter authentication handler and API objects were created to be able to call the Twitter cursor function.

4.2 Dataset Preparation

1. Sentiment Analysis - The Dataset used in this section are (1) u.item of MovieLens 100k dataset to get the movie list and tag words and; (2) Twitter data to get sentiment scores.

**To prepare the tag words**, a series of steps were performed:

1. Removed the trailing (year) for each movie title
2. Reformat the movie titles with incorrect construction (e.g. "Birdcage, The" and "Time to kill, A")
3. Removed space for each formatted movie title.

**To clean the Twitter data,** the following preprocessing steps were performed:

1. Dropped null and duplicate values
2. Removed alphanumeric characters, including emojis.
3. Removed links
4. Removed stopwords by importing nltk—Corpus stopwords library.
5. Lemmatized each words using nltk.stem.WordNetLemmatizer.

**Average tweets per movie:** A limit of 5 tweets per movie was enforced during extraction.

* **Top 10 movies:** Based on the extracted tweets,"The Big Squeeze" topped the movie with the highest positive review. Followed by "Til There was you," with a 93% average sentiment score.

**Chart

Description automatically generated**

*Figure 14: Top 10 movie* *Graph*

* **Bottom 10 movies:** The movie with the lowest sentiment score turned out to be "Killer: A Journal of Murder (1995) with -84%, followed by "Faster Pussycat! Kill! Kill! with -82%

Chart, funnel chart

Description automatically generated

*Figure 15: Bottom 10 movie Graph*

4.3 Evaluation Metrics

Some of the evaluation methods that make the model fits nicely over the distribution area. Two approaches have been used to compare the two models by applying (1) RMSE (Root Mean Square Error), which measures the error of a model in quantitative predictive data.

𝑅𝑀𝑆𝐸 = √∑𝑛 (𝑦̂𝑖− 𝑦𝑖)2 𝑖=1 /𝑛 (1)

Here, 𝑦̂1, 𝑦̂2, ..., 𝑦̂𝑛 are predicted values. 𝑦, 𝑦 ,... , 𝑦 are observed values.   
𝑛 is the number of observations

The standard deviation σ can be calculated by dividing by n under the square root in RMSE. It is an excellent measure to calculate the standard deviation σ of a typical observed value from our model's prediction, assuming that our observed data can be decomposed as:

𝑜𝑏𝑠𝑒𝑟𝑣𝑒𝑑 𝑣𝑎𝑙𝑢𝑒 (2)  
= 𝑝𝑟𝑒𝑑𝑖𝑐𝑡𝑒𝑑 𝑣𝑎𝑙𝑢𝑒

+𝑝𝑟𝑒𝑑𝑖𝑐𝑡𝑎𝑏𝑙𝑦 𝑑𝑖𝑠𝑡𝑟𝑖𝑏𝑢𝑡𝑒𝑑 𝑟𝑎𝑛𝑑𝑜𝑚 𝑛𝑜𝑖𝑠𝑒 𝑤𝑖𝑡h 𝑚𝑒𝑎𝑛 𝑧𝑒𝑟𝑜

The random noise can be anything (e.g., unknown variables). If RMSE calculates the value of noise as small, it means that the designed model is good at predicting our observed data. And, in the vice-versa case, it means that model is failed to account for the essential features underlying the data. (2) Mean Absolute error is one of the many metrics for summarizing and assessing the quality of a machine learning model. Here, error refers to the subtraction of Predicted value from Actual Value as below.

𝑃𝑟𝑒𝑑𝑖𝑐𝑡𝑖𝑜𝑛 𝐸𝑟𝑟𝑜𝑟 = 𝐴𝑐𝑡𝑢𝑎𝑙 𝑉𝑎𝑙𝑢𝑒 − 𝑃𝑟𝑒𝑑𝑖𝑐𝑡𝑒𝑑 𝑉𝑎𝑙𝑢𝑒 (3) This prediction error is taking for each record after which we convert all error to positive. This is achieved by taking the Absolute value for each error as below:  
𝐴𝑏𝑠𝑜𝑙𝑢𝑡𝑒 𝐸𝑟𝑟𝑜𝑟 → |𝑃𝑟𝑒𝑑𝑖𝑐𝑡𝑖𝑜𝑛 𝐸𝑟𝑟𝑜𝑟| (4)

Finally, we calculate the mean for all recorded absolute errors (Average sum of all fundamental errors).

∑𝑛|𝑦−𝑥| (5)  
𝑀𝐴𝐸=𝑖=1𝑖 𝑖 / 𝑛

Here, 𝑦𝑖 is the predicted value, 𝑥𝑖 is the actual value, and 𝑛 is the number of observations. (3) Precision:  precision is calculated as the number of true positives divided by the total number of true positives and false positives.

* Precision = True Positives / (True Positives + False Positives) . (6)

(4) Recall: This is calculated as the number of true positives divided by the total number of true positives and false negatives.

* Recall = True Positives / (True Positives + False Negatives) . (7)

(5) F-score: It combines precision and recall into a single measure that captures both properties.

* F-Measure = (2 \* Precision \* Recall) / (Precision + Recall). (8)

The figure shown below shows the evaluation metrics using Pytorch and the collaborative filtering recommender system.

Graphical user interface, text, application

Description automatically generated

*Figure 16: Using PyTorch evaluation score*

Table

Description automatically generated

*Figure 17: Collaborative filtering evaluation score*

4.4Results and Analyses

**1) Quantitative Analysis**

We first look at the comparison of RMSE and MAE errors between a Collaborative Filtering based and Hybrid system. Here we choose the top 10-recommended movies by both methods for ten users and calculate RMSE errors for each strategy for comparison. The data shown in figure 15 and figure 16 indicates that the hybrid system has comparatively lower RMSE overall. We then do the same evaluation for MAE, and from figure 15 and figure 16 see that the hybrid recommendation system has comparatively lower MAE, i.e., better accuracy.

2) **Qualitative Analysis**

We see that Collaborative Filtering can tell us the movies that a user is likely to rate higher. But it has no way of recommending similar movies to a particular one tailored for the specific user.Whereas, a hybrid system gives us the similar recommendation Therefore, we can conclude that from both qualitative and quantitative perspective, a hybrid recommendation system performs comparatively better than standalone Collaborative Filtering or Content-Based Filtering recommendation system.

4.4.1. Hyperparameter Selection

In this section, two different experiments were run. For both models, Matrix Factorization and Hybrid Recommender, it was calculated the RMSE returned when running the model with latent vectors of sizes 20, 30, 40, and 50. It was plotted a line diagram RMSE x Latent Feature sizes for each model, see figures 18 and 19. The Matrix Factorization model demonstrates a linear relation between the number of latent features and RMSE where RMSE decreases as the number of latent features increases. The lowest RMSE value was around 1.85. The Neural Network experiment demonstrated that RMSE decreases as the embedding layers size increase until it reaches the size of 40, when RMSE starts to remain constant, with lower fluctuation, at value 0.91.

Comparing both results, it is worth noting the Hybrid Recommender gave lower RMSE value, 0.91, than the first model, Matrix Factorization, which gave 1.85.

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

*Figure 18: Matrix Factorization - RMSE x Number of latent features*

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

*Figure 19: Hybrid Recommender - RMSE x Number of latent features*

4.4.2 Results **of** Recommendation Models

From the previous section, the models with the lowest RMSE were selected to conduct experiments in this section. For each model, the following experiments were run:

* Create a top-20 recommendation list for a user’s.
* Check if the movies in the recommended list were watched by the user
* Based on these suggestions, calculate the RMSE, RMAE, Precision, Recall, and F1-Score
* Repeat the process for all users
* Calculate the mean of all metrics

Table 1 shows the metrics of the top-10 recommendation for each model. The Hybrid Recommender had lower RMSE and MAE than Matrix Factorization. Matrix Factorization Recall was 14% higher than the one output by the Hybrid Recommender model. However, precision and F1-score of the Neural Network Recommender were 53% and 37%, respectively, higher.

|  |  |  |
| --- | --- | --- |
| Metrics of Top-10 recommendation | | |
|  | Matrix Factorization | Hybrid Recommender |
| RMSE | 1.289 | 0.879 |
| MAE | 1.099 | 0.826 |
| Precision | 0.143 | 0.220 |
| Recall | 0.862 | 0.755 |
| F1-score | 0.229 | 0.315 |

*Table 1: Metrics of Top-10 recommendation*

The table 2, shows the metrics of the top-20 recommendation for each model. All the metrics of the Hybrid Recommender model were better the Matrix Factorization ones. Precision, Recall and F1-score of Neural Collaborative Filtering were 24%, 57%, and 24% higher than the ones provided by the traditional method.

|  |  |  |
| --- | --- | --- |
| Metrics of Top-20 recommendation | | |
|  | Matrix Factorization | Hybrid Recommender |
| RMSE | 1.435 | 0.925 |
| MAE | 1.238 | 0.834 |
| Precision | 0.209 | 0.259 |
| Recall | 0.739 | 0.903 |
| F1-score | 0.295 | 0.367 |

*Table 2 . Metrics of Top-20 recommendation*

**5. CONCLUSION**

The present work compares two different recommendation systems, the traditional Matrix Factorization and the Hybrid Recommender driven by the newly Neural Collaborative Filtering.

Three experiments were conducted to evaluate the performance of each model. First, it was demonstrated that RMSE decreases as the number of latent features increases from 20 to 50. Secondly, two top-10 recommendation lists generated from each model were compared. All metrics, except recall, were better in the list generated by the Hybrid Recommender. Lastly, two top-20 recommendation lists generated from each model were compared. All metrics of the Hybrid Recommender were better than the ones of Matrix Factorization.

In conclusion, the Hybrid Recommender performed better demonstrated to be a potential method to improve recommender systems that drive industry today.

6. REFERENCES

1. Ricci, F., Rokach, L., Shapira, B. (2015) Introduction and Challenges. In: Ricci F., Rokach L., Shapira (Eds). *Recommender Systems Handbook*. (1-34). <https://doi.org/10.1007/978-1-4899-7637-6_1>

2. Ricci, F., Rokach, L., Shapira, B. (2011) Introduction to Recommender Systems Handbook. In: Ricci F., Rokach L., Shapira B., Kantor P. (Eds). *Recommender Systems Handbook*. (1-35). <https://doi.org/10.1007/978-0-387-85820-3_1>

3. Joel Azzopardi, Department of Artificial Intelligence Faculty of ICT University of Malta, [Item-based vs User-based Collaborative Recommendation Predictions](http://www.keystone-cost.eu/ikc2017/slides/presentation_17.pdf)

4. Melville, P., Sindhwani, V. (2010). Recommender System. Encyclopedia of Machine Learning. <https://doi.org/10.1007/978-0-387-30164-8_705>

5. Resnick, P. J., Varian, H. R., (1997). Recommender Systems. *Communications of ACM, 519(3),* 56-58. <https://doi.org/10.1145/245108.245121>

6. G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," in IEEE Transactions on Knowledge & Data Engineering, Apr. 25, 2005, pp. 734-749.

7. B. Mobasher, “Recommender Systems,” Kunstliche Intelligenz, Special Issue on Web Mining, BottcherIT Verlag, Bremen, Germany, 2007, pp. 41-43.

8. M. Y. H. Al-Shamri and K. K. Bharadwaj, "Fuzzy-genetic approach to recommender systems based on a novel hybrid user model," Expert systems with applications, vol. 35, no. 3, pp.1386-1399., Oct. 2008.

9. Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction 12*, 331–370.

<https://doi.org/10.1023/A:1021240730564>

**10.** Jiawei Han, ... Jian Pei, in [Data Mining (Third Edition)](https://www.sciencedirect.com/book/9780123814791/data-mining-concepts-and-techniques), 2012