Netflix Recommender System

Giorgos Gantsios, Lazaros Keramaris, Maria Moysidou

January 23, 2024

Abstract

This paper explores how we can make movie recommendations more accurate and personalized. With so many movies available, it can be hard for users to find ones they'll enjoy. Our goal is to improve the movie recommendation process to make it more helpful for users, leading to a better overall experience.

We used different methods to achieve this. Singular Value Decomposition (SVD) helps us understand hidden patterns in user preferences, connecting users with similar tastes. Cosine Similarity looks at movie features to find those with similarities, and we also added a neural network to capture more complex relationships in the data.

Our early results show that these methods improve recommendation accuracy. Collaborative and content-based filtering help us understand what users like, while the neural network adds a layer of sophistication to catch even subtle patterns. Combining these methods opens up new possibilities for creating more effective recommendation systems, blending traditional and modern techniques.

1 Introduction

In the realm of streaming platforms, content recommendation systems play a pivotal role in enhancing user experience by offering personalized suggestions tailored to individual preferences. The sheer volume of content available on platforms like Netflix poses a challenge for users to navigate, and an effective recommendation system can significantly alleviate this issue. The importance of such systems is underscored by the ever-growing library of diverse content, ranging from movies and TV shows to documentaries and original productions.

The problem at hand revolves around the need to create an advanced Netflix Recommender System that goes beyond generic suggestions and instead provides users with content that aligns closely with their tastes and preferences. This task is intricate due to the vast and continuously evolving nature of the platform's content library, coupled with the unique and varied viewing habits of each user.

The motivation behind developing an advanced recommender system for Netflix is rooted in the understanding that user satisfaction is intricately linked to the relevance and appeal of the recommended content. A more accurate and personalized recommendation system not only enhances user engagement but also contributes to customer retention and loyalty. In the highly competitive streaming landscape, the ability to offer a curated and enjoyable viewing experience is a strategic advantage that can set a platform apart.

Moreover, the deployment of a sophisticated recommendation algorithm aligns with the broader industry trend of leveraging artificial intelligence and machine learning to enhance various aspects of user interactions. By delving into the complexities of user preferences and viewing patterns, we aim to contribute to the evolution of recommendation systems, pushing the boundaries of what is possible in terms of personalization and accuracy.

In this study, we present an algorithm designed to enhance the accuracy of movie recommendations. The input to our algorithm comprises three distinct datasets: "movie_titles.csv," which includes essential information such as movie titles, release years, and unique identifiers; "combined_data.txt," providing user ratings for each movie; and "netflix_titles.csv," containing movie titles along with their corresponding descriptions. The diverse nature of these datasets allows us to capture crucial facets of user preferences and movie details.

To transform this input into meaningful recommendations, we employ a combination of powerful methodologies, including Support Vector Machines (SVM), Cosine Similarity, and Neural Network.

Each of these techniques plays a unique role in deciphering the complex patterns inherent in user behavior and movie characteristics. The SVM aids in discerning underlying structures, Cosine Similarity measures the similarity between movies, and the Neural Network unravels intricate relationships within the data.

Our objective is to output a refined and personalized movie recommendation based on the amalgamation of these methodologies. The predicted movie recommendation takes into account factors such as user ratings, movie descriptions, and release years, providing users with a tailored selection that aligns closely with their preferences.

2 Literature Review

Netflix's recommendation system plays a crucial role in shaping the user experience, influencing viewership patterns, and ultimately driving revenue. Over the years, numerous research efforts have delved into developing effective recommendation algorithms, employing a diverse range of approaches and methodologies, ranging from traditional techniques like content-based filtering and collaborative filtering to more advanced methods utilizing neural networks.

2.1 Collaborative Filtering

Collaborative filtering uses aggregated behaviour of a large number of users to suggest relevant items to certain users. More specifically, in order to provide recommendations, it uses similarities between users and items simultaneously. Collaborative filtering algorithms recommend items based on the preferences of similar users. These algorithms identify patterns in user behaviour and exploit these patterns to predict items that users might like based on the ratings of other users who share similar tastes.

Based on this kind of filtering, Xin Guan et al [GLG17] proposed a combination of item-oriented and user-oriented high density seeking strategies to create a more effective SVD model which emphasizes on the most popular items. Using matrix factorization on a sub-matrix containing these items, the writers noticed that this is a suitable way to approximate a given matrix with missing values.

In our case, we used the original matrix and the surprise library to create the train and test sets. Comparing the results of these two approaches, we can see that our approach is slightly better since it gives an average RMSE of 0.8 and the other an RMSE¿0.9.

2.2 Content-Based Filtering

Content-based filtering systems analyze preferences given by a particular user and attempt to build a model around this data. The difference between this approach and Collaborative filtering is that the recommendation is based on the information of each item (in our case, the cast, the movie description, the genre etc.) and not only the similarity by rating. In contrast with Collaborative filtering, CB filtering solves the "cold start problem" since the system recommends new movies even though the user has not rated any movie.

This kind of filtering comes with its disadvantages too, since it depends on item metadata, which means that a full description of the movie should be used and in that way the user is going to receive limited recommendations based on the same content.

S. Reddy et al [RNK$^+$19], proposed an approach where the content based filtering is done based on the genre of the movies. More specifically, past behavior of the user is being analysed to achieve the recommendation of movies based on the similarity of genres. To do that, they use two subsets of the same dataset. One for the user ratings and one for the genres of the movies. The dot product is used in order to create the user profile matrix. A similarity measure is calculated by computing the least distance between the user and the other users.

In contrast to this approach, in our project we used Cosine Similarity as a measure because the dot product emphasizes on the frequency at which a movie appears. This means that if the movie is chosen by a small amount of people then there is a chance that it won't be recommended to other users who would actually like it.

2.3 Neural Networks

In this NN approach [Myl18], in order to overcome the issue of cold start, three methods were used: collaborative filtering, ANN to extract the content features from the items and SVM for the process of unknown data prediction. In our case, we used the dataset directly in the NN and didn't use any other method like collaborative filtering.

2.4 Hybrid Recommendation System

Y. Afoudi et al [ALA21] proposed a hybrid recommendation model using CF, CBF, SOM CF and hybrid filtering. SOM is a form of ANN that is trained using unsupervised learning and it was used in order to solve the issue of unsupervised clustering in the dataset. This approach uses SVD, TF-IDF, cosine similarity and SOM for the movie dataset and uses both CBF and SOM to create the users profile. Then, the recommended mpvies come from combining the weighted sum of CF scores and CBF scores with supervised and ranked by SOM-CF sum. In this approach the recommendation speed is a lot slower than the one from all the other approaches, but the results are said to be better.

2.5 State of The Art Approach

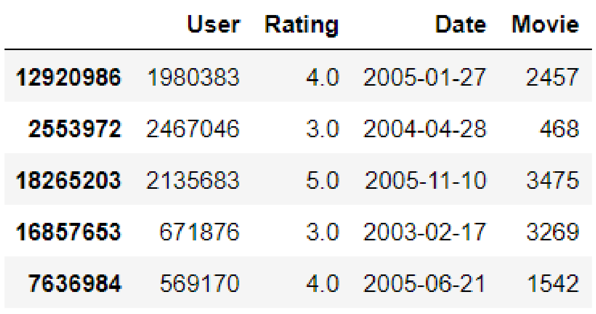
Content-based and collaborative filtering recommendations where the state-of-the-art more than 10 years ago. Apparently, there are many different models and algorithms to improve the prediction performance. Combining the approach of A. Kaushik et al [KGB18] in which they proposed the use of a Neural Network based on the similarity of movie posters. More specifically, it is mentioned that the results of this approach seemed very promising making the inclusion of movie posters in datasets a good asset to create a state-of-the-art recommender system.

3 Dataset and Features

The collaborative filtering and NN recommender systems were based on the Netflix Prize dataset. This dataset contains movie and users information. To make it simpler, we used the combined_data_1.txt file where the combination of all the data in the files are stored. To make the recommender systems more accurate we can use the other 3 combined data text files too.

To preprocess the data, we removed the entries which weren't giving any significant information and would make the accuracy of the models worse.

The dataset 'combined_data_1.txt' we used was in form:

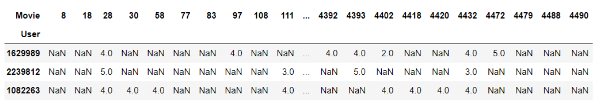


	User	Rating	Date	Movie
12920986	1980383	4.0	2005-01-27	2457
2553972	2467046	3.0	2004-04-28	468
18265203	2135683	5.0	2005-11-10	3475
16857653	671876	3.0	2003-02-17	3269
7636984	569170	4.0	2005-06-21	1542

Figure 1: Initial DataFrame

We can see each row is a rating of a certain user for a certain movie with no missing values as only existing ratings exist in the dataset.

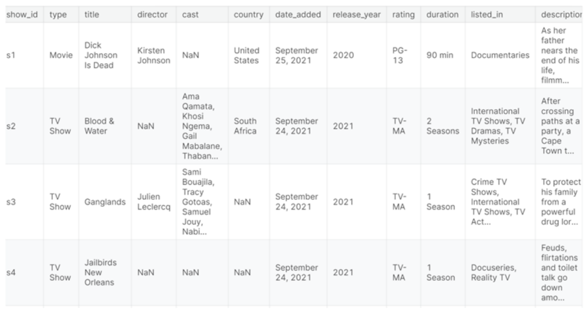
We converted it to a user-row and movie-column dataframe where an element i, j is the rating of user i for movie j. Obviously, this introduces a lot of missing values as most of the users do not rate or even watch most of the movies.This prompted us to filter the dataframe so as to ignore unpopular movies and idle users that do not provide much information to the system anyway but overload the calculations. We ended up with a dataframe of this form:



Movie	8	18	28	30	58	77	83	97	108	111	...	4392	4393	4402	4418	4420	4432	4472	4479	4488	4490
User																					
1629989	NaN	NaN	4.0	NaN	NaN	NaN	NaN	4.0	NaN	NaN	...	4.0	4.0	2.0	NaN	NaN	4.0	5.0	NaN	NaN	NaN
2239812	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	3.0	...	NaN	5.0	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN
1082263	NaN	NaN	4.0	4.0	4.0	NaN	NaN	NaN	NaN	4.0	...	NaN	NaN	4.0	4.0	4.0	4.0	NaN	NaN	NaN	NaN

Figure 2: Filtered Dataframe

For the content-based filtering, the netflix_titles.csv dataset is used. This dataset contains basic information about movies and series like titles, descriptions, cast etc.



show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	descriptior
s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...

Figure 3: netflix_titles.csv

4 Methods

Regarding the methods we used, we turned to SVD for collaborative filtering between users and cosine similarity for content-based filtering because we preferred to explore methods based on both similar users and movies that share features. Additionally, we incorporated a neural network implementation to further enhance our recommendation system, leveraging its ability to capture intricate patterns and relationships within the dataset.

4.1 SVD for collaborative filtering

The advantages of the use of collaborative filtering for a recommender system are as follows: first, it is independent of the contents of recommended items; second, it can be closely integrated with social networks; third, it has good accuracy in terms of recommendations. SVD is a matrix factorization technique commonly used for producing low-rank approximations. Given a matrix $X \in R^{m \times n}$ with rank$(X) = r$, the Singular Value Decomposition of X is defined as the following:

$$X = USV^T \tag{1}$$

where $U \in R^{m \times m}$, $V \in R^{n \times n}$ and $S \in R^{m \times n}$. The matrices U, V are orthogonal, with their columns being the eigenvectors of XX^T and X^TX, respectively. The middle matrix S is a diagonal matrix with r nonzero elements, which are the singular values of X. In our case, U denotes the user features, S is the diagonal matrix of singular values and V transpose are the move features. An important property of SVD, which is particularly useful in recommender system, is that it can provide the optimal approximation to the original matrix X using three smaller matrices multiplication.

4.1.1 Handling missing values

For the SVD method we first had to deal with the missing values of the user x movie dataframe. So we experimented with 3 methods:

1. We filled the missing values (NaN) with zeros so that the sparsity of the dataframe is preserved. However, this gives the model the low rating assumption where the user has not seen the movie.

2. We filled the missing values (NaN) with the average rating of the column (movie) in order to have more relevant imputation. However, this means that the sparsity of the dataframe is lost and may make the computation more complicated.

3. We used the SVD() method of the surprise library which treats missing values as unknowns and ignores them.

4.2 Cosine Similarity for content-based filtering

Cosine similarity is a method to measure the difference between two non zero vectors of an inner product space. Mathematically, it measures the co- sine of the angle between two vectors projected in a multi-dimensional space. The smaller the angle , the higher the cosine similarity. So in this case, what it measures is how similar the preferences between two users are. In our case we use it both for user-user recommendations and for finding items (movies) that are similar to a certain item.

$$cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{2}$$

The dataset we used in Cosine Similarity for content-based filtering is 'netflix_titles.csv'. This dataset contains information on movie's director, actors, category and description which we use to find the similarity between movies. From this information we create a soup which we import into a vectorizer of the sklearn library. By calculating the angle cosine we have a measurement of how similar 2 bands are. So, given a movie we can recommend the movies that are most similar (have the highest cosine) to the movie.

4.3 Neural Networks

Neural networks are computational models inspired by the structure and functioning of the human brain. They consist of interconnected nodes, or neurons, organized into layers. A neural network typically comprises an input layer, one or more hidden layers, and an output layer. Each connection between neurons has an associated weight, and the network learns through a process of training, adjusting these weights to minimize the difference between predicted and actual outcomes.

Neural networks are powerful tools for pattern recognition, feature extraction, and complex function approximation. They excel in capturing intricate relationships within data and can be applied to various tasks, including classification, regression, and recommendation systems.

In the context of our recommendation system, the neural network implementation serves as a sophisticated model to discern intricate patterns in user preferences and movie characteristics. By learning from the underlying structures of the dataset, the neural network contributes to the accuracy and personalization of our movie recommendations.

Mathematically, the output of a neural network can be represented as:

$$\text{output} = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3}$$

where w_i represents the weights, x_i represents the inputs, b is the bias term, and is the activation function. Training the neural network involves adjusting these weights and biases to minimize the difference between predicted and actual outputs.

5 Experiments/Results/Discussion

5.1 Introduction to experiments

Our experiments aimed to evaluate the performance of a recommender system employing three aforementioned distinct algorithms:

- Singular Value Decomposition (SVD) for collaborative filtering

- Cosine similarity for content-based filtering
- Neural Network for collaborative filtering

Let's break each experiment down and analyse the evaluation metrics of each experiment.

5.2 SVD experiment and error evaluation

After entering the data and preprocessing it, we will proceed to the implementation of the first method which is SVD. We approached the implementation of the SVD method in 2 different ways. The surprise library and the svds of scipy.sparse.

5.2.1 SVD using surprise

First, we implemented SVD using surprise library. It is important to underline that this method handles missing values of user-movie matrix as unknown and unobserved. Here we even use the build_full_trainset() method to separate the set into training and testing. The results we got are as follows:

- RMSE: 0.8176
- MSE: 0.6685

Below we can see the suggested movies based on a wanted user's similarity with other users. It is important to say this method was the most time-consuming one used in the project as it may needed more than a minute to run.

5.2.2 SVD using sparse matrix

Here the svds() method requires the user-movie table to have no empty cells. We understand that in such a sparse table, the way we approach empty evaluations (cells) plays a very important role in the performance of the algorithm. Already from the previous milestone we approached the problem with 2 practices:

- Filling the empty ratings with 0 which significantly drops the average of the ratings.
- Filling the empty cells with the average of the individual column significantly reducing the variance by simulating each non-existent rating as the average score of the individual movie.

This method runs a lot faster than surprise's svd() and using the same evaluation metrics we take the following results:

1. **missing values = 0:**
   - RMSE = 1.8311
   - MSE = 3.3529
2. **missing values = movie mean:**
   - RMSE = 0.8885
   - MSE = 0.7894

Easily, we decide that filling empty ratings with each column's (movie's) mean rating is preferable than filling with zero.

Finally, when deciding between surprise and sparse matrix for SVD, it happens to be a trade-off between accuracy and speed, where surprise's svds() is the most accurate but slow and sparse matrix's svd() is the faster but a little less accurate.

5.3 Cosine Similarity experiment

For this experiment we used 'netflix_titles.csv' dataset that contains movies and information about each movie (director, cast, duration, category, description etc.). The input here is a single movie and the output is multiple movies that match input's features. This part of the recommender system could be used to suggest similar movies to the one a user is currently wathcing, as it is based only on movie (item) similarity.

The cosine similarity experiment is implemented with based on the vectorization (with the help of sklearn.feature_extraction.text's TfidfVectorizer()) of the description, actors, directors and category of the movie. So, a 'soup' is created using sklearn library's CountVectorizer() and fit_transform(). We have roughly evaluated the model by giving the input titles of known movies and observing the output we saw that relevant movies appear in which the same actors or directors or movies of similar content appear.

For example, in the input we put Pokémon Master Journeys: The Serieson and the output (which is the Recommender System suggestions) is Pokémon Journeys: The Series, Pokémon the Series, Glitter Force Doki Doki, etc. which we know empirically are movies of similar content and with the same actors. With Breaking Bad coming in we get Better Call Saul, The Assassination of Gianni Versace etc. which we also know are of a similar category (crime, action).

While our evaluation of cosine similarity did not involve a formal mathematical approach, we conducted empirical trials and observed, based on experiential evidence, that the model performs effectively.

5.4 Implementation with Neural Network using Keras

In the last experiment for our project we use Keras to create a Neural Network Recommender System. This will be a collaborative filtering model. Now let's dive into the implementation.

The dataset used here is the df_filtered we produced from 'combined_data1.txt'. We use mapping separately for user and movie to make data form more dense and we divide the dataset to train and test, again separately. After we create the model and the embedding layers for users and movies we reshape the vectors to prepare for the next layer. Next is a dense layer to increase model's ability to learn and generalize from training data. Now, we can compile the data and specify the loss function we want (here we choose mse) and the optimizer algorithm (here we choose adam). Choosing between optimizer algorithms is not easy because they all have strengths and weaknesses but we chose adam because it is a familiar one and it produced the best results from the ones we have tried.

After model.fit(), finally, we are ready to test on testing dataset. We calculate RMSE from model's mse and the result is:

- RMSE = 0.9069

We can say that the result is quite close with the SVD method. It is the best result we got by tweaking the parameters. The embedding size and batch size parameters of the neural were however approximated randomly so it is possible for one to find a better approximation than the final one.

6 Conclusion/Future Work

In conclusion, our study showcases the effectiveness of collaborative filtering using Singular Value Decomposition (SVD), content-based filtering with cosine similarity, and a neural network for collaborative filtering. Notably, the neural network excelled in capturing intricate user-item interactions, while SVD minimized Mean Squared Error (MSE). The nuanced performance trade-offs highlight the necessity of selecting algorithms based on specific system goals.

For future work, deeper exploration of hyperparameter tuning and neural network architecture refinement is warranted. Collaboration enhancements may facilitate ensemble methods for improved accuracy. Furthermore, we recognize the potential in pursuing a hybrid approach that seamlessly integrates collaborative and content-based filtering. Such an approach holds promise for creating a more robust and personalized recommendation system, marking a path for future advancements in recommender system research.

7 Contributions

Throughout the project, each team member contributed collaboratively and synergistically, fostering an environment of shared responsibility and equal involvement. Our team embraced a collective approach, with all members actively participating in every phase of the project, including problem formulation, literature review, algorithm selection, implementation, experimentation, and analysis.

To ensure equal contribution, team members engaged in regular collaborative meetings where decisions were made collectively. Coding tasks were often approached through pair programming, enabling the simultaneous involvement of all team members in the implementation of algorithms and system components. Additionally, documentation and report writing were collaborative efforts, with team members jointly contributing to the creation of this comprehensive report.

By adopting this inclusive and cooperative approach, we aimed to eliminate any disparities in individual contributions, ensuring that each team member played an integral role in all aspects of the project. The commitment to equal involvement and shared responsibility enhanced the overall quality of our work and contributed to the successful completion of the recommender system project.

References

[ALA21] Yassine Afoudi, Mohamed Lazaar, and Mohammed Al Achhab. Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network. *Simulation Modelling Practice and Theory*, 113:102375, 2021.

[GLG17] Xin Guan, Chang-Tsun Li, and Yu Guan. Matrix factorization with rating completion: An enhanced svd model for collaborative filtering recommender systems. *IEEE Access*, 5:27668–27678, 2017.

[KGB18] Ajay Kaushik, Shubham Gupta, and Manan Bhatia. A movie recommendation. system using neural networks. *International Journal of Advance Research, Ideas and Innovations in Technology*, 4(2):425–430, 2018.

[Myl18] Bharadwaja Krishnadev Mylavarapu. Collaborative filtering and artificial neural network based recommendation system for advanced applications. *Journal of Computer and Communications*, 6(12):1–14, 2018.

[RNK$^+$19] SRS Reddy, Sravani Nalluri, Subramanyam Kunisetti, S. Ashok, and B. Venkatesh. Content-based movie recommendation system using genre correlation. In Suresh Chandra Satapathy, Vikrant Bhateja, and Swagatam Das, editors, *Smart Intelligent Computing and Applications*, pages 391–397, Singapore, 2019. Springer Singapore.

Links

- Recommendation Systems using Different Methods.ipynb - Colaboratory.
- Recommender System Using Machine Learning
- Netflix Movies and TV Shows, for the 'netflix_titles.csv' dataset
- Netflix Prize data, for the 'combined_data.txt' dataset