Movie Recommendation System

Danielė Ačaitė, Evita Petkevičiūtė, Rūta Okulič-Kazarinaitė

Abstract

The World Wide Web continues to grow at a massive rate and all the sites with all the information and their complexity continue to grow along with it. Thus, it becomes time-consuming and burdensome for people to find similar information themselves. To enhance the search for relevant things and make it easier to use them, websites can be personalized. This is where recommendation systems become valuable because they can dynamically add hyperlinks which results in making it easier for users to find interesting things and in the meantime make the interaction between the system and the user better. These personalized recommendation systems are used in various movie sites.

1 Introduction

Movie recommendation systems are increasingly popular and widely used in the entertainment industry. They are a type of information filtering systems that aims to predict the ratings of preferences a user would give to a movie, based on the user's past ratings and the ratings of other users. By recommending movies that are likely to be highly rated by a user, the system can help users discover new films to watch and improve their experience. However, it requires expensive computations in order to find the essential properties as the number of users as well as movies, increases exponentially.

According to M. Sridevi and Dr .R. Rajeswara Rao [1], there are several approaches to building recommendation systems. Therefore, we attempt to include Demographic filtering, Content-based filtering, Collaborative filtering, and Hybrid Recommendation systems. Demographic filtering is based on demographic information about the user. Content-based filtering uses the characteristics of the movies themselves (for example, genre, director, and actors) to recommend similar movies to users. Collaborative filtering is based on the idea of predicting users' ratings for a movie based on the ratings given by other users with similar tastes. Hybrid methods combine both collaborative filtering, and content-based filtering to make recommendations.

In this research paper, we will explore the use of machine learning techniques to build a movie recommendation system based on I Ahmed method [2]. We will discuss the challenges and limitations of building movie recommendation systems as well as the results.

2 Project goals and Outcomes

Our objective is to create a movie recommendation system based on 4 types of recommendation systems that are acknowledged by the literature review.

The principal steps of our study are: 1)Reviewing the types of movie recommendation systems in the literature; 2)Finding the dataset required for the project; 3)Performing the modeling; 4)Reporting results and explaining all the steps of the modeling process thoroughly; 5)Providing conclusions and future work.

Completing this project will require familiarising us with various types of movie recommendations: Collaborative filtering, Content-based filtering, Demographic filtering, and Hybrid Recommendation

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

systems. Gaining practice and expertise in distinguishing these types and learning reproducible experimental design.

3 Literature Research / Methods used

3.1 Simple recommender

The Simple Recommender suggests generalized recommendations for every user according to how popular a movie is and sometimes its genre. THe main idea behind this recommender is that movies with higher popularity and a higher rate of critically acclaimed will have a greater probability of being enjoyed by the average watcher. However, this model does not give personalized suggestions based on user information.

3.2 Demographic filtering

Demographic filtering is a method of making recommendations for a user based on demographic information about the user. The demographic properties may be things like the user's age, gender, location, or occupation. This information can be used to suggest movies that are likely to be of interest to the user based on the preferences or characteristics of people with similar demographics. For example, if a recommendation system knows that a user is a 20-year-old woman, it might recommend romantic comedies or dramas, which are known to be popular among young women. It is more advantageous than collaborative filtering since it addresses the issue of new users with limited data and is easier to implement. If new users have not rated any movies the system can recommend movies based on one's demographic information. On the other hand, it lacks uniqueness. For instance, let's say there is a new young male user and the system recommends action movies as it is the most popular among males. However, it does not capture that this male user might not enjoy action movies since he is more of a romantic comedy fan.

3.3 Content-based filtering

Content-based filtering systems choose items based on the correlation between items' content and the user's preferences as opposed to the collaborative filtering system meaning that they suggest items based on a particular topic [3]. These recommendation systems have some things in common like means for describing the items that may be recommended, for creating an account of the user that describes the types of items the user prefers, and in more complex scenarios a means of comparing items to the user account to dictate what to recommend [4]. Recommendations are made by comparing a profile with the matter of each file in the correlation where there are a lot of wordings. This system has item metadata where there is information about the director, genre, description, actors, etc for films to create these recommendations. Some words in the plot description like tags and stop words are removed because they occur pretty often and cannot be used as discriminators. The remaining usually are minimized by removing prefixes and suffixes. In the end, content-based recommendation systems look for items that a person has liked and then search for similar ones. Indicating that the movies that were the most liked will likely be similar and recommended.

Term Frequency-Inverse Document Frequency method will be used. It is a numerical statistic that is used to reflect the importance of a word in a document or a collection of documents (called a corpus). It is commonly used in information retrieval and natural language processing.

TF-IDF is calculated by multiplying the term frequency (TF) of a word by the inverse document frequency (IDF) of the same word. The term frequency is the number of times a word appears in a document, while the inverse document frequency is a measure of how rare the word is across the entire corpus. By reducing the importance of words that occur frequently in plot overviews (or any other type of document), TF-IDF can help to focus on the words that are more meaningful and informative for the specific task at hand. This can be useful, for example, when comparing documents or determining the relevance of a document to a particular query.

Steps that were taken:

- Overview columns indicating the plot's description meaning that conversion into the matrix by computing TF-IDF vectors was needed;

- Using the cosine similarity to denote the similarity between two different movies;
- Defining recommendation system results and limitations.

3.4 Collaborative filtering

Collaborative filtering is a method of making recommendations for a user by collecting preferences or taste information from many other users. This information is then used to identify patterns and suggest items that are likely to be of interest to the user. In the case of a movie recommendation system, collaborative filtering involves collecting ratings or reviews from many users and using those ratings to suggest movies to a particular user. The assumption behind collaborative filtering is that if user A has similar preferences to user B, and B likes a particular movie, then A is more likely to like that movie as well. M. Sridevi and Dr. R. Rajeswara Rao address 3 main issues of this type of movie recommendation system [5]:

- Most users do not rate the movies;
- The recommendation fails to recommend a movie when new users come to the environment since they do not rate movies and the system lacks data;
- As the number of users increases exponentially it becomes expensive and computationally hard to compute an accurate system.

3.5 Hybrid filtering

Hybrid filtering is a technique used in movie recommendation systems that combines the properties of collaborative filtering and content-based filtering. Collaborative filtering relies on the preferences of similar users to make recommendations, while content-based filtering uses the characteristics of the movie itself to make recommendations. By combining these two approaches, hybrid filtering is able to make more accurate and personalized recommendations. For example, a hybrid system might consider a user's past ratings of comedies, as well as the genre and actors of the movie being recommended, to provide a more tailored suggestion. Hybrid filtering can help to improve the effectiveness of movie recommendation systems by combining the two methods.

4 Methodology and Experiments

4.1 Datasets

All provided datasets were extracted from kaggle [6]. The features of the datasets were displayed in the coding environment.

4.2 Simple filtering

The model's implementation is extremely trivial because it requires sorting the movies based on ratings and popularity. The top movies on the list are later displayed. As an additional step, a genre argument was implemented to present the top movies of a particular genre.

4.3 Demographic filtering modification

This paper examines the demographic filtering methodology based on the TMDb (The Movie Database) website. This is one of the major databases about movies and television shows. TMDb allows users to access metadata for various TV shows, movies. These users are able to find the status of the movies, get a list of trending films and get the features. One of the simplest metrics that could be used is movie ratings. However, this method is not as advantageous since, firstly, it does not take into account the popularity of the movie. If there is a movie with a rating of 9 by 50 000 users will have a lower place than a movie which was rated 9.5 by only three users, hence, the list would not be reliable. Therefore, one of the main metrics that are used when building a movie recommendation is the weighted rate used in the TMDb [7].

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

- v - number of votes for the movie -> already given
- m - minimum number of votes that keep movie in the list (we take 80
- R - mean rating of the movie –> already given
- C - mean rating of all the movies in the list (calculated in the code, mean is 6.092)

Steps:

- filtering of movies that qualify for the list → 96
- define function of tmdb weighted rating and create new feature score and calculate the value by applying a function to our DataFrame of qualified movies
- sorting movies based on score:

	Title	Vote_count	Score
1881	The Shawshank Redemption	8205	8.248353
662	Fight Club	9413	8.096134
3337	The Godfather	5893	8.007404
3232	Pulp Fiction	8428	8.074738
65	The Dark Knight	12002	8.044250
809	Forrest Gump	7927	7.972814
96	Inception	13752	7.969290
95	Interstellar	10867	7.937399
1990	The Empire Strikes Back	5879	7.904757
1818	Schindler's List	4329	7.900080

Figure 1: List representing movies based on weighted rating

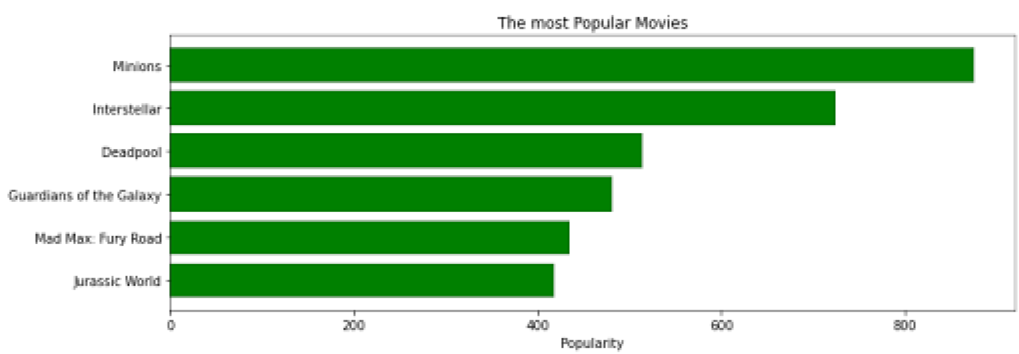

Figure 2: List representing movies based on popularity

These are the results of basic Demographic Filtering Movie Recommender. The table of 10 movies displays the list of the movies which can be treated as “Trending now” movies. That proves that demographic filtering does not take into account the individual preferences as tastes of a particular user.

4.4 Content-based filtering modification

This recommendation system is based on the pairwise similarity index. The index is used according to the plot descriptions and movies are recommended based on the similarity score.

At first, text reprocessing was performed in order to convert the word vector of each overview. Then vectors were computed using TF-IDF. Each word vector needs to be converted by computing TF-IDF vectors. This provides a matrix whose each column is a word in the overview vocabulary and each row shows the original movie name.

After using the scikit-learn function it showed that there are more than 20 000 different words that describe more than 4800 films in the given dataset. Cosine similarity was used to calculate a numeric quantity that denotes the likeness between two different movies. This index is more independent of magnitude and is in comparison not difficult and faster to calculate. In the end, Sklearn's linear.kernel() was used instead of cosine.similarities() since it is faster.

So the system would recommend, the output is set to a movie's name and the input is set to the movie's name and their description of other liked 10 similar movies. This was done by identifying the index of a movie in the metadata, prior given its title, meaning that reverse mapping was needed.

4420	Black Rock
1816	The Best of Me
1057	Coach Carter
3382	The Mighty Macs
3478	College
4541	The Slaughter Rule
2562	When the Game Stands Tall
507	Independence Day
228	An American Carol
3900	Air Bud

Figure 3: Recommendations made using TF-IDF

Nonetheless, the given output quality can be improved because the recommendation system does not take into account the cast, crew, keywords and genres. After converting the list into a safe and usable structure, the functions are written to help to extract the needed information from each feature. Later all the words are put into lowercases and the spaces are moved so that the system would calculate the name and surnames as one word. The string "metadata soup" is created where all the metadata is because it is needed to give to the vectorizer.

Now the countvectorizer() is used instead of TF-IDF because there is no need to down-weight the presence of actors or directors. Finally, the recommendation system outputs similar movies.

436	Grow Ups 2
499	Jack and Jill
2989	Happy Gilmore
361	You Don't Mess with the Zohan
445	Just Go with It
1539	Big Daddy
796	The Ridiculous 6
802	That's My Boy
1126	Here Comes the Boom
1392	The Benchwarmers

Figure 4: Recommendations made using countvectorizer()

4.5 Collaborative filtering modification

To provide an even more personalized recommendation system, it is worthy to implement collaborative filtering. In this case, we are looking at specific persons (userID) ratings for certain movies and trying to predict another person's rating for the same movie. That can be used to recommend movies to people in the same interest groups.

To implement this model, we used Surprise Python Scikit due to its specific recommendation functions. Then we implemented a 5-fold cross-validation - to predict new data, together with RMSE and MAE - to predict the quality of the predictions. Overall, we aim for a low RMSE for more precise results. We get the RSME equal to 0.89684716, which is decent. Then we train the model:

```
trainset = data.build_full_trainset()
svd.fit(trainset)
```

Figure 5: Training the data set

Now to check the results we pick a random user based on their id, and predict the rating for the movie (also based on the id), in this case:

```
[108] # predicting users rating for a movie
      svd.predict(10, 20, 3)

      Prediction(uid=10, iid=20, r_ui=3, est=3.0636579409030085, details={'was_impossible': False})
```

Figure 6: Predicting the ratings

Here, we predict what user (10) would rate the movie (20). SVD - Singular Value Decomposition algorithm is used to model the user and item biases from users and items, which uses stochastic gradient descent to optimize the parameters.

5 Limitations

Digging into recommendation system issues and limitations, there are a few main ones to be named:

- Slow start for new users or also known as the cold start problem, that is related to the sparsity of information. As proposed by Lika et al., can be split into three categories [8]:
  - Recommendations for new users;
  - Recommendations for new items;
  - Recommendations for new users on new items.
- Users experience might not be pleasant at first because they will not receive personalized recommendations that would be accurate until they have watched enough movies.
- Another issue is that some recommendation systems are vulnerable to feedback loops because recommendations made by the system influence the data that is used to generate future recommendations. Along with that, the feedback loops also may compromise recommendation quality and homogenize user behavior, raising ethical and performance concerns [9].
- Additionally, if the data that the models were trained on were biased, the model will be negatively affected by those multiple biases. As found in some studies, fairness and other social bias could be amplified by the use of graph structures. To name some bias that make this issue more important: data bias, algorithmic bias (model bias), result and response bias, and feedback loop bias, as proposed in a paper by Chizari et al. [10].
- Collaborative Filtering algorithms are limited due to high computational cost and in terms of space and time, when dealing with large-scale data, another issue could be the mentioned earlier cold start problem [11].

These and many other issues do not have a quick or easy fix - it requires attention and ethical decisions. For example:

- The data set could be including metadata, diverse (including different genders, age groups, races etc.), and updated often [12].

While these limitations impose a serious problem for recommender systems, some of them can be helped through regulation, management of data and domain experts of ethics and other.

6 Results

We have created recommendations using content-based, collaborative filtering, and demographic filtering. Demographic filtering is important in movie recommendation systems because people's interests and preferences can often be correlated with their demographics (e.g. age, location, gender). E. g. taking into account a user's demographics, our recommendations can be potentially more accurate and relevant. To make more personalized suggestions, we used collaborative filtering, in our case we tried to predict a users rating for a movie purely based on others' ratings. Collaborative filtering can be used in various ways, that can improve since it is easy to implement, scale, and personalize. Content-based filtering was used to create recommendations based on the metadata about the movies, which can be useful when we do not have enough personalized data. The project models are a base of a recommendation system, that is responsible for identifying items that are likely to be of interest to a user, ranking those items in order of predicted relevance or preference, modeled off of given data. Besides some limitations, not only do movie recommendation systems can minimize users' already spent time on the platform by discouraging them from leaving the website but recommendation systems also can help them find good-fitting movies faster, thus increasing spent time staying on the website. In the end, it raises business' total profits.

7 Discussion and conclusion

Current trends of movie recommender systems show that there is huge potential for them to improve to even more precise level. Our project summarizes and explains how the recommend systems work with certain data and models. To conclude, the advantage of our research and simple movie recommending algorithm is that it is easy to implement in potential real-world situations, it can be personalized and it provides a good base to be worked with future ideas or other improvements.

8 Future work

The methods we used mostly focus on predictive accuracy. The future of recommendation systems should look at a broader range of metrics, for example, serendipity - in order to recommend unexpected or novel items to a user that may be outside of the user's usual interests, but still end up being relevant. Another metric is coverage, improving that would allow the system to provide recommendations for a larger number of items, and increase the chances that a user will find something that they are interested in [13]. Another thing that is being improved in the recommendation algorithm research is employing on graph neural networks (GNNs) on recommendation in which collaborative filtering are exhibited as high-order connectivity, and are used in context-aware recommender systems [14]. To take an even further step forward, additional ways of improving recommender algorithms are using users "sentiments and emotions" in order to create an specific recommendation, where fine-tuned BERT was used to extract sentiment and emotion information [15].

9 Contributions

All authors contributed equally throughout all steps of the project, including literature review and research, and model analysis.

10 Code

The project code is located at github: https://github.com/okukaz/movie

References

[1] M. Sridevi. Table 1 from decors : A simple and efficient demographic collaborative recommender system for movie recommendation: Semantic scholar, Jan 1970.

[2] Ibtesama. Getting started with a movie recommendation system, May 2020.

[3] Robin Van Meteren and Maarten Van Someren. Using content-based filtering for recommendation. In *Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop*, volume 30, pages 47–56, 2000.

[4] Peter Brusilovski, Alfred Kobsa, and Wolfgang Nejdl. *The adaptive web: methods and strategies of web personalization*, volume 4321. Springer Science & Business Media, 2007.

[5] Iateilang Ryngksai and L Chameikho. Recommender systems: types of filtering techniques. *International Journal of Engineering Researck & Technology, Gujarat*, 3(2278-0181):251–254, 2014.

[6] Rounak Banik. The movies dataset, Nov 2017.

[7] Jiang Zhang, Yufeng Wang, Zhiyuan Yuan, and Qun Jin. Personalized real-time movie recommendation system: Practical prototype and evaluation. *Tsinghua Science and Technology*, 25(2):180–191, 2019.

[8] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert systems with applications*, 41(4):2065–2073, 2014.

[9] Karl Krauth, Yixin Wang, and Michael I Jordan. Breaking feedback loops in recommender systems with causal inference. *arXiv preprint arXiv:2207.01616*, 2022.

[10] Nikzad Chizari, Niloufar Shoeibi, and María N Moreno-García. A comparative analysis of bias amplification in graph neural network approaches for recommender systems. *Electronics*, 11(20):3301, 2022.

[11] Harris Papadakis, Antonis Papagrigoriou, Costas Panagiotakis, Eleftherios Kosmas, and Paraskevi Fragopoulou. Collaborative filtering recommender systems taxonomy. *Knowledge and Information Systems*, pages 1–40, 2022.

[12] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *arXiv preprint arXiv:2010.03240*, 2020.

[13] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 257–260, 2010.

[14] Jiancan Wu, Xiangnan He, Xiang Wang, Qifan Wang, Weijian Chen, Jianxun Lian, and Xing Xie. Graph convolution machine for context-aware recommender system. *Frontiers of Computer Science*, 16(6):1–12, 2022.

[15] CheonSol Lee, DongHee Han, Keejun Han, and Mun Yi. Improving graph-based movie recommender system using cinematic experience. *Applied Sciences*, 12(3):1493, 2022.