Movie Recommendation System Using Machine Learning & Web Application

Md. Baker
Department of Electrical and Computer Engineering, North South University
Dhaka, Bangladesh
md.baker@northsouth.edu

Kazi Kaarima Nashia
Department of Electrical and Computer Engineering, North South University
Dhaka, Bangladesh
kazi.nashia@northsouth.edu

Fatema Akter Rimi
Department of Electrical and Computer Engineering, North South University
Dhaka, Bangladesh
fatema.akter02@northsouth.edu

***Abstract*—**A recommender system, a subclass of information filtering systems, aims to predict a user's rating or preference for an item. Based on this prediction, it would suggest relevant items to the users, making their experience more personalized and enjoyable. For example, in the case of Netflix, it would suggest movies; in the case of Kindle, it would be books. In this project, we worked with the MovieLens dataset and then built two recommender systems: content and item-based collaborative systems. The former system recommends relevant movies to the users by using the content of the previously searched items that the user likes. At the same time, the latter deals with finding similar movies using the SVD model. Here, content refers to the movie's attributes and genre. At the end of the work, evaluation parameters were incorporated into the systems, achieving 0.65 in average MAP and 0.865 in average NDCG. At the same time, scores of 0.6410 and 0.4968 have been seen in RMSE and MAE, respectively.

Keywords—recommender system; recommendation system; content-based filtering; collaborative filtering; hybrid system; recommendation algorithm; recommendation models

I. INTRODUCTION

A recommendation system is a data collection and filtering method that helps search engines improve and show users more relevant and related results. In the era of digital entertainment, users are overwhelmed with a vast array of movie options. Finding movies that suit their preferences can take time and effort. This problem is exacerbated by the lack of personalized recommendations that cater to specific tastes.

Collaborative filtering, content-based filtering (also known as the personality-based approach), and other systems like knowledge-based systems are employed in most recommendation systems. In collaborative filtering, a model is created based on a user's prior actions (things previously purchased or selected, as well as the numerical ratings given to those items). Then, this model forecasts the ratings for things the consumer could find interesting.

To recommend other movies with almost the same qualities, content-based filtering approaches use a collection of distinctive, previously occurring features of an element. In today's recommender systems, a hybrid system often combines one or more methodologies.

In the field of movie recommendation systems, several approaches have been explored over the years. These approaches include content-based filtering, collaborative filtering, matrix factorization, deep learning-based and more methods.

Collaborative Filtering: Traditional collaborative filtering techniques like user-based and item-based recommendation systems were the earliest attempts at movie recommendations.

Content-Based Filtering: Content-based filtering systems recommend movies based on user profiles and movie content analysis.

Matrix Factorization: Techniques like matrix factorization were used to improve recommendation accuracy.

Deep Learning: Deep neural networks and models like deep autoencoders were applied to improve recommendation quality.

Hybrid Approaches: Hybrid recommendation systems combine collaborative and content-based filtering for better results.

Reinforcement Learning: Reinforcement learning optimized movie recommendations based on user feedback.

K-means clustering: The user/item-specific information is grouped to form a cluster using K-means clustering.

With an increase in the number of users and movies, the computational time of the system increases. Model building is expensive in the case of Bayesian belief nets. There is a trade-off between prediction performance and scalability. Lose useful information since only user ID and movie ID are used to create the embeddings. Limited diversity in recommendations leads to filter bubbles. Scalability issues when dealing with a large number of users and items.

To solve the limitations, we implement advanced matrix factorization techniques and deep learning models to reduce data sparsity issues. And use the best model from which we got the best accuracy and relevance.

Our unique contributions to this project include:

- Integrating machine learning models for improved accuracy and relevance to develop a robust recommendation system that delivers accurate and relevant movie recommendations.
- The system can recommend movies to new users who have yet to rate any movies.
- The system uses a similarity matrix that is more effective than traditional similarity measures

.

II. RELATED WORKS

Taejun Lim et al. presented the Global-Local Kernel-based matrix completion framework, GLocal-K. GLocal-K is a matrix completion framework that utilizes global and local kernels to enhance the performance of recommender systems. It effectively captures global and local user-item interaction patterns, leading to more accurate and personalized recommendations. GLocal-K offers a promising approach to matrix completion and recommendation systems by effectively capturing global and local user-item interaction patterns. Their model outperforms the state-of-the-art baselines on three collaborative filtering benchmarks: ML-100K, ML-1M, and Douban.

D.K. Yadav et al. presented MOVREC, a collaborative filtering-based movie recommendation system. The information provided by the user is used in collaborative filtering. That data is analyzed, and a movie is recommended to users, starting with the movie with the highest rating. The system also allows the user to specify which attributes he wants the movie to be recommended.

Luis M Campos et al. analyzed two types of traditional recommender systems: content-based and collaborative filtering. Given the shortcomings, he proposed a new system that combines the Bayesian network and collaborative filtering. The proposed system solves the problem by optimizing it and providing probability distributions for making valuable inferences.

Harpreet Kaur et al. have presented a hybrid system. The system utilizes a combination of content and a collaborative filtering algorithm. While recommending movies, the film's context is also considered. The user-user relationship, as well as the user-item relationship, play a role in the recommendation.

Utkarsh Gupta et al. use chameleon to combine user-specific or item-specific information to form a cluster. This is an efficient recommender system technique based on Hierarchical clustering. A voting system is used to predict the rating of an item. The proposed system has lower error and better clustering of similar items.

Urszula Kuelewska et al. proposed clustering to deal with recommender systems. Two methods for computing cluster representatives were presented and evaluated. The effectiveness of the proposed two methods was compared using a centroid-based solution and memory-based collaborative filtering methods. As a result, the generated recommendations' accuracy increased significantly compared to the centroid-based method.

Costin-Gabriel Chiru et al. proposed Movie Recommender, a system that provides movie recommendations based on information about the user. This system attempts to address the issue of unique recommendations caused by ignoring the user's specific data. The psychological profile of the user, their viewing history, and data involving movie scores from other websites are all collected. They are based on a calculation of aggregate similarity. The system employs both content-based and collaborative filtering.

Hongli Lin et al. proposed content-boosted collaborative filtering (CBCF) to predict the difficulty level of each case for each trainee. The algorithm is split into two stages: content-based filtering, which improves the existing trainee case rating data, and collaborative filtering, which provides final predictions.

Nicolas Hug proposed a paper on the Surprise library in Python. It is used for building and analyzing rating prediction algorithms. Surprise is mainly written in Python, while the computationally intensive parts are optimized with Cython. Internally, Surprise relies on built-in Python data structures and numpy arrays. It provides a set of rating prediction estimators (or prediction algorithms). Among others, classical algorithms are implemented as the most similarity-based algorithms, and custom prediction algorithms can be created.

Ramni Harbir Singh et al. illustrated the modeling of a movie recommendation system by using content-based filtering within the recommendation system. The KNN algorithm is implemented in this model along with the principle of cosine similarity, which yields more accuracy than other distance metrics, and the running complexity is comparatively low, too.

G. Linden et al. presented the most popular recommender system, Amazon. Amazon is one of the most important sales companies in the United States. As an internet-based retailer, it's a recommendation system integrated into the marketing tool. This technique uses Item-to-Item Collaborative Filtering to personalize the website to suit each customer's interests. A characteristic property of Amazon's recommender system is that it's specially designed to scale to many customers and products to supply high-quality recommendations in real-time.

III. METHODOLOGY

A. Flowchart

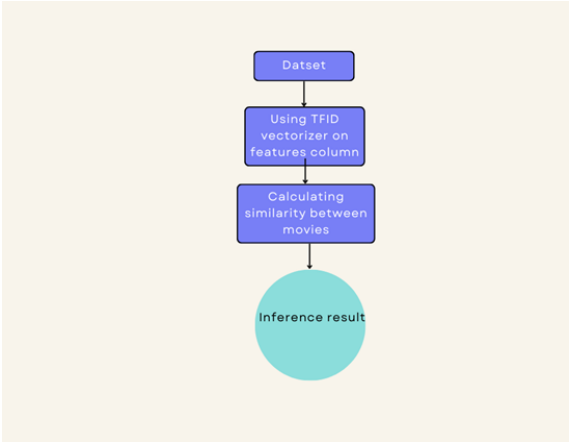


Figure 1. Content-Based filtering diagram

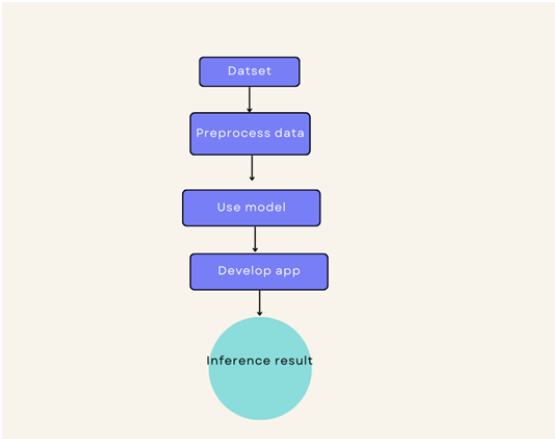
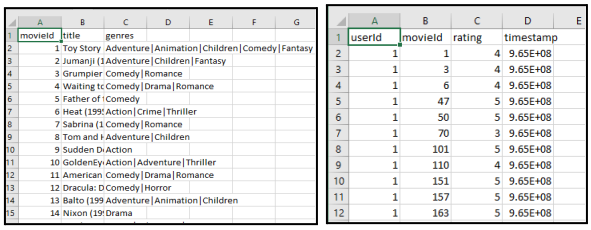


Figure 2. Collaborative filtering diagram

B. Dataset

We used MovieLens dataset(ml latest small). It has 4 tables: movies, ratings, links and tags. For our project, we used only 2 tables: movies and ratings. The movies csv file has movie id, title and genre while the ratings file contains user id, movie id, ratings and genre. All of this information is fundamental when we will apply filtering. For tags csv file, it contains the tags that best describes the movie. As for the links csv file, it contains movie id, imdb id and tmdb id.



movieId	title	genres
1	Toy Story	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji	(1Adventure|Children|Fantasy
3	Grumpier	Comedy|Romance
4	Waiting tc	Comedy|Drama|Romance
5	Father of	Comedy
6	Heat (199	Action|Crime|Thriller
7	Sabrina (1	Comedy|Romance
8	Tom and H	Adventure|Children
9	Sudden De	Action
10	GoldenEy	Action|Adventure|Thriller
11	American	Comedy|Drama|Romance
12	Dracula: D	Comedy|Horror
13	Balto (199	Adventure|Animation|Children
14	Nixon (19	Drama

userId	movieId	rating	timestamp
1	1	4	9.65E+08
1	3	4	9.65E+08
1	6	4	9.65E+08
1	47	5	9.65E+08
1	50	5	9.65E+08
1	70	3	9.65E+08
1	101	5	9.65E+08
1	110	4	9.65E+08
1	151	5	9.65E+08
1	157	5	9.65E+08
1	163	5	9.65E+08

Figure 3. MovieLens Dataset

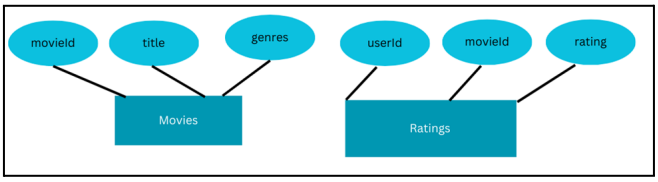
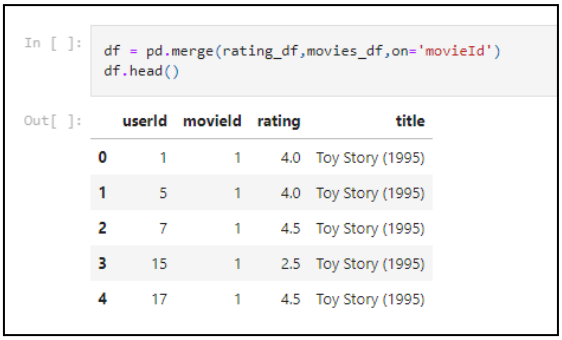


Figure 4. MovieLens Dataset diagram

At first, the movies and ratings file were merged using their common attribute, that is movie id. This is done to get the overall information of the movies. After this, EDA was performed on the dataset. We checked the presence of Nan values in any column and if found, the rows were dropped.

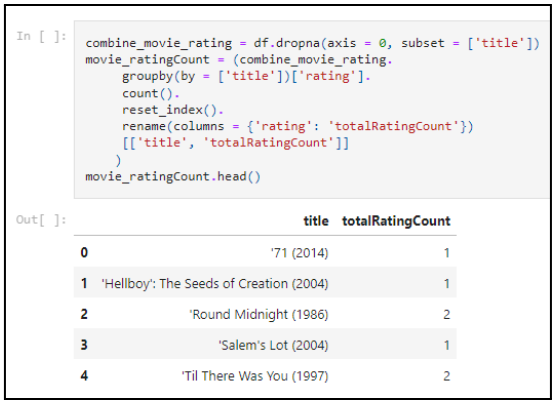


```
In [ ]: df = pd.merge(rating_df,movies_df,on='movieId')
        df.head()
```

Out[ ]:

	userId	movieId	rating	title
0	1	1	4.0	Toy Story (1995)
1	5	1	4.0	Toy Story (1995)
2	7	1	4.5	Toy Story (1995)
3	15	1	2.5	Toy Story (1995)
4	17	1	4.5	Toy Story (1995)

Figure 5. Merging data



```
In [ ]: combine_movie_rating = df.dropna(axis = 0, subset = ['title'])
        movie_ratingCount = (combine_movie_rating.
             groupby(by = ['title'])['rating'].
             count().
             reset_index().
             rename(columns = {'rating': 'totalRatingCount'})
             [['title', 'totalRatingCount']]
            )
        movie_ratingCount.head()
```

Out[ ]:

	title	totalRatingCount
0	'71 (2014)	1
1	'Hellboy': The Seeds of Creation (2004)	1
2	'Round Midnight (1986)	2
3	'Salem's Lot (2004)	1
4	'Til There Was You (1997)	2

Figure 6. Drop null value from dataset

C. Models and Programming languages

Initially, we worked with various models such as Decision Tree, Random Forest, Global K, followed by content filtering and collaborative filtering. In the end, we selected collaborative filtering as its accuracy was best compared with the other models.

For web implementation, Python, being the backbone of machine learning, was used along with Flask for developing the web application, accompanied by HTML and CSS for front-end and back-end.

D. Web Application

```python
@app.route('/')
def index():
    return render_template('index.html', movie_names=movies['title'])

@app.route('/recommend', methods=['POST'])
def recommend():
    movie_name = request.form['movie_name']
    movie_id = movies[movies['title'] == movie_name]['movieId'].values[0]

    # Get movie recommendations based on item similarity
    recommendations = get_movie_recommendations(movie_id)

    return render_template('recommendations.html',movie_name=movie_name, recommendations=recommendations)

def get_movie_recommendations(movie_id, num_recommendations=10):
    # Find movies most similar to the selected movie
    similar_movies = list(item_similarity_df[movie_id].sort_values(ascending=False).index[1:num_recommendations+1])
    recommended_movies = movies[movies['movieId'].isin(similar_movies)]['title'].tolist()
    return recommended_movies

if __name__ == '__main__':
    app.run(debug=True)
```

Figure 7. Code for checking movie recommender system

Figure 8. Home page for movie recommendation

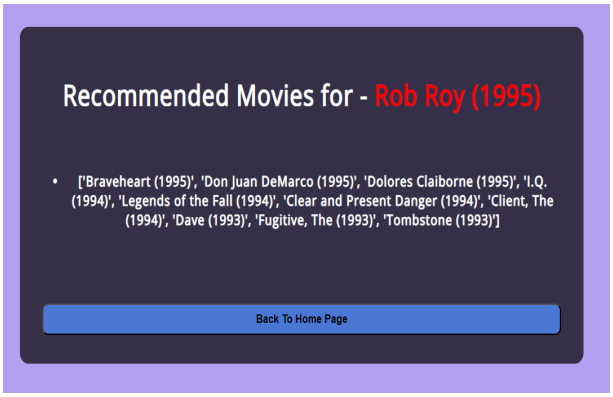


Figure 9. Recommended Movies

IV. RESULTS

V. CONCLUSION

Recommendation systems have become an important part of everyone's lives. With an enormous number of films released worldwide each year, people frequently miss out on some amazing works of art due to a lack of appropriate suggestions. Putting machine learning-based recommendation systems to work is thus critical for getting the right recommendations. We saw content-based recommendation systems that, while ineffective, can solve the difficult problems that collaborative filtering methods face when run independently. Similarly, neural network embedding can improve the quality of recommendations and make them more user-personalized. As a result, we conclude that researching various approaches to recommendation engines is critical for developing a collaborative engine that overcomes the shortcomings of these independent approaches while multiplying their benefits. Whereas independent approaches to a movie recommendation system may have flaws, they will assist users in receiving accurate movie recommendations when combined correctly.

In the future, we aim to improve the suggestion process by adding a user registration feature to save each user's unique preferences and recommend movies accordingly. In recommender systems, user data is always useful. In the future, we can collect more user data and add a list of movies users dislike. We will also enter the list of disliked movies into the recommender system and generate scores that will be added to the previous result. We can improve the recommender system's results in this manner. Create an internal service for the recommender system. We can make it an internal API for developers to use in the future. Some of the website's movie lists will be sorted by recommendation.

VI. REFERENCE

[1] R. Singh, S. Maurya, T. Tripathi, T. Narula, and G. Srivastav, “Movie recommendation system using cosine similarity and knn,” pp. 2249–8958, 06 2020.

[2] M. Kumar, D. Yadav, A. Singh, and V. K. Gupta, “Article: A movie recommender system: Movrec,” Inter-national Journal of Computer Applications, vol. 124, pp. 7–11, August 2015.

[3] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales, “Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks,” International Journal of Approximate Reasoning, vol. 51, no. 7, pp. 785 – 799, 2010.

[4] E. A. S. Harpreet Kaur Virk, Er.Maninder Singh, “Analysis and design of hybrid online movie recommender system,” International Journal of Innovations in Engineering and technology, 2015.

[5] U. Gupta and N. Patil, “Recommender system based on hierarchical clustering algorithm chameleon,” Inter-national Advance Computing Conference (IACC), 2015.

[6] U. Kuzelewska, “Clustering algorithms in hybrid recommender system on movielens data,” Studies in Logic,Grammar and Rhetoric, vol. 37, 01 2014.

[7] C. Chiru, C. Preda, V. Dinu, and M. Macri, “Movie recommender system using the user’s psychological profile,” in 2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 93–99, 2015.

[8] N. Hug, “Surprise: A python library for recommender systems,” Journal of Open Source Software, vol. 5,p. 2174, 08 2020.

[9] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: item-to-item collaborative filtering,”IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, 2003

[10] S. C. Han, T. Lim, S. Long, B. Burgstaller, and J. Poon, “Glocal-K,” Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021. doi:10.1145/3459637.3482112

[11] G. Malve, L. . Lohar, T. . Malviya, and S. . Sabnis, “Movie Recommendation System ”, IJRITCC, vol. 9, no. 4, pp. 13–16, Apr. 2021.

[12] H. Lin, X. Yang, and W. Wang, “A content-boosted collaborative filtering algorithm for personalized training in interpretation of radiological imaging,” Journal of digital imaging.