An Enhanced Movie Recommender System Using Hybrid Filtering Techniques

Ayush Linghwal*, Neeraj Negi, Nikita Malik*****

Abstract: ***In this research paper, a movie recommender system using machine learning is designed and developed. To offer certain tailored movie suggestions based on user choices and previous interactions, the system uses content-based filtering, collaborative filtering, and a hybrid approach. The Python solution uses Streamlit for its web interface and includes libraries: Scikit-learn, Pandas, and NumPy. According to experimental results, the hybrid model improves user happiness and accuracy in suggestion, while resolving typical issues of data sparsity with the cold-start problem.***

Keywords: Streamlit, Machine Learning, Content-driven Filtering, Collaborative Filtering, Movie Recommender System.

1. INTRODUCTION

Users find it challenging to choose films that suit their tastes as the number of films available on digital platforms grows. Dissatisfaction usually results from the absence of a specific experience afforded via customary search mechanisms. Through making film recommendations dependent upon viewing history as well as user choices, a movie recommender system eases content discovery. This study uses a machine learning-based movie recommender system in order to increase recommendation accuracy. It blends collaborative and content-based filtering strategies.

Digital streaming services providing wide-ranging movie collections intended for various audiences have thoroughly changed the way people obtain entertainment. As the quantity of easily accessible films continues to increase, users battle. Often, they cannot select films that match their interests.

Recommender systems are used extensively by e-commerce, online education, and digital media streaming. Advanced algorithms get used by services, such as Netflix, Amazon Prime, as well as Disney+ [4] for examination of user behavior, plus make content recommendations depending on previous exchanges. Besides helping users locate content, these systems further engagement, retaining audiences plus generating income for the platform.

To get around these restrictions, this study suggests a hybrid movie recommender system of collaborative and also content-based filtering strategies. The system serves for a larger audience; it integrates the advantages of both approaches in order to provide recommendations that are more varied with better accuracy. The system's implementation involves multiple data preprocessing steps, multiple feature extractions, plus similarity calculations. Machine learning-based ranking mechanisms are components of this implementation. The study assesses the efficacy of the suggested system in terms of user satisfaction as well as recall, accuracy, and also precision.

This paper's remaining sections are arranged as follows: Existing recommendation methods and related work are covered in Section 2. The methodology and system design are described in Section 3. Implementation details are presented in Section 4, and performance evaluation and results are presented in Section 5. Section 6 concludes by outlining conclusions and potential avenues for system improvement.

2. LITERATURE REVIEW

The development of movie recommender systems has benefited from several studies:

- Sarwar et al. (2001) [1] developed item-based collaborative filtering, a technique that improves scalability over user-based approaches.
- He et al. (2017) investigated neural collaborative filtering using deep learning models to increase recommendation accuracy in recommender systems.
- Schafer et al. (2007) highlighted the benefits of integrating various filtering methods to improve accuracy and diversity in hybrid recommender models.

The existing literature demonstrates that hybrid recommender systems [3] outperform traditional standalone models in terms of accuracy and adaptability.

3. BACKGROUND STUDY

3.1 Movie Recommendation Systems

By making of personalized film recommendations, movie recommendation systems improve with it the user experience. Based on previous interactions, metadata, as well as behavioral patterns, they broadly forecast user interests via data analytics plus machine learning.

3.2 Content-Based Filtering

Content-based filtering [2] recommends movies through an analysis of their attributes, such as with genres, actors, directors, and also descriptions. This technique uses text-

**Student, Dept. of Computer Applications, Maharaja Surajmal Institute, GGSIP University, New Delhi-58, India*
***Assistant Professor, Dept. of Computer Applications, Maharaja Surajmal Institute, GGSIP University, New Delhi-58, India*
****Corresponding author: *nikitamalik@msijanakpuri.com*

processing methods. Such methods, like TF-IDF (Term Frequency-Inverse Document Frequency) [5] and cosine similarity, compute the relevance of a movie to a user's past preferences.

Collaborative filtering, unlike content-based methods, makes suggestions from similar users' preferences. There are basically two kinds:

1. User-based collaborative filtering: Identifies users that share similar viewing preferences as well as suggests films that they certainly enjoyed.
2. Item-based collaborative filtering: Finds films comparable with ones a user saw and gave high ratings to.

3.3 Hybrid Approach

Hybrid models combine content-based and collaborative filtering methods to improve recommendation accuracy. This approach mitigates the cold-start problem by leveraging both movie metadata and user interactions.

3.4 Libraries and Technologies Used

1. **Pandas and NumPy:** These libraries are important for purposes that handle movie datasets. They perform numerical computations, and structure data efficiently.
2. **Scikit-learn [7]:** machine learning algorithms, like similarity measures and matrix factorization techniques for collaborative filtering.
3. **Streamlit [8]:** It's a Python web framework that's used in the development of a user interface for some recommendation system.
4. **SQLite [9]:** A lightweight database for storing user ratings, movie details, and recommendation logs.

Cosine Similarity & Matrix Factorization: Applied in both content-based and collaborative filtering models to compute relationships between movies and users.

3.5 CSV

A popular file format for storing and for sharing of structured data is Comma-Separated Values (CSV). In this movie recommender system, CSV files store user ratings and interaction data. Movie details are also stored. To better ease data management and also enable recommendation retrieval, the system analyzes certain CSV files for extraction of information. For preprocessing, also training, and further testing, the use of CSV guarantees that such datasets remain small and accessible.

Python's wide-ranging libraries are used in the construction of the movie recommender system. These are for web development, machine learning, as well as data processing.

The system's frontend is developed using Streamlit [8], a web framework based in Python, which allows creation of friendly, interactive interfaces. Many users of Streamlit [8] can readily interact with the system, enter preferred movies, and receive recommendations in real-time.

The backend is developed via Python, incorporating scikit-learn [7] for implementing a number of machine learning algorithms like collaborative filtering, content-based filtering, and hybrid models. The system uses pandas and NumPy also in data preprocessing for handling large movie datasets.

The system's data, including user ratings as well as movie details, are stored and thoroughly managed using SQLite [9], a lightweight relational database. Additionally, CSV files can be used in the handling of structured datasets, to allow for easy data retrieval and manipulation.

4. RELATED WORK

Several techniques have been employed in recommender systems, including:

1. **Content-Based Filtering:** Utilizes movie metadata (genre, actors, directors) to suggest similar movies.
2. **Collaborative Filtering:** Recommends movies based on user behavior and preferences.
3. **Hybrid Approach:** Combines both methods to mitigate limitations such as cold-start problems and sparsity.

Prior studies indicate that hybrid models yield better accuracy compared to standalone techniques.

5. SYSTEM DESIGN AND METHODOLOGY

5.1 Data Gathering and Preparation

The dataset, which includes movie details like titles, genres, descriptions, and user ratings, is used to train and test the system. In data preprocessing, missing values are handled and ratings are normalized.

Using methods like TF-IDF vectorization recommendation techniques, text data can be transformed into numerical representations.

Cosine similarity [6] is used in content-based filtering to compare movie descriptions and recommend related content.

Collaborative filtering uses similarity metrics like Pearson correlation to apply both item-based and user-based filtering.

To improve recommendation accuracy and get around individual constraints, the hybrid model combines the two methods.

5.2 System Architecture

The system comprises three main components:

1. **Data Processing Module:** Cleans and structures the dataset.
2. **Recommendation Engine:** Implements content-based, collaborative, and hybrid filtering models.
3. **User Interface:** Built with Streamlit [8] to provide real-time recommendations and user interaction.

6. IMPLEMENTATION

6.1 Data Collection

1. Collecting movie datasets containing movie details, user ratings, and genres.
2. Storing structured data in CSV format for easy retrieval and processing.

6.2 Data Pre-processing

1. Handling missing values by filling in or removing incomplete data.
2. Normalizing numerical features like ratings to improve model performance.
3. Transforming text data using TF-IDF vectorization to analyze movie descriptions.

6.3 Training the Recommendation Models

1. ***Content-Based Filtering:*** Calculating cosine similarity between movie features to generate recommendations.
2. ***Collaborative Filtering:*** Using user-based and item-based similarity matrices to suggest movies.
3. ***Hybrid Approach:*** Combining content-based and collaborative filtering for improved accuracy.

6.4 Model Deployment

1. Implementing a Streamlit [8]-based user interface for interactive recommendations.
2. Connecting the system to a SQLite [9] database for managing user interactions and ratings.
3. Providing real-time recommendations based on user input and feedback.

6.5 Performance Evaluation

1. Evaluating system accuracy using precision, recall, and F1-score.
2. Analyzing computational efficiency and response time for real-time recommendations.

By following these steps, the system ensures scalability, efficiency, and accurate recommendations for users.

6.6 Experimental Setup

To assess the system's performance under various circumstances, it is tested in a variety of settings. Figure 1 shows the user interface of the developed movie recommender system. Several user scenarios are included in the test setup to evaluate efficiency, accuracy, and robustness. The assessment takes into account a number of factors, including:

- User interaction: evaluating suggestions made by different user profiles with varying tastes in films.
- Data Size Variation: To assess the system's scalability, run it on datasets of different sizes.
- Algorithm Comparison: Examining the variations in accuracy among collaborative, content-based, and hybrid filtering models.
- Real-Time Performance: Assessing how quickly the system responds when making dynamic movie recommendations. Evaluating the system's performance for new users with no past interaction history is known as the "cold-start problem."
- Diversity and Novelty Testing: Verifying that the system offers unique and diverse suggestions as opposed to reiterating preexisting ones.

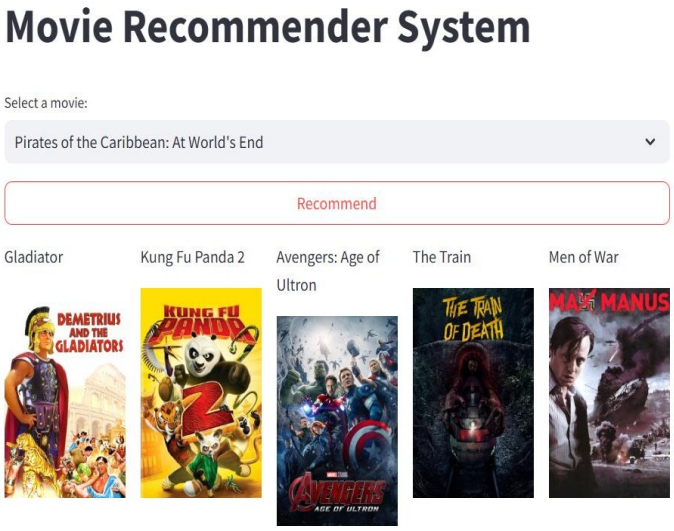

Figure1: Movie Recommender System user interface

6.7 Testing Scenarios

- Baseline Testing: Running the recommender system with a small dataset to ensure core functionality and correctness.
- Large-Scale Testing: Using a dataset with thousands of movies and user ratings to evaluate performance under real-world conditions.
- Cold-Start User Simulation: Creating new user profiles and analyzing the system's ability to generate meaningful

recommendations.

- User Feedback Integration: Collecting user responses to assess satisfaction with the recommendations and make necessary adjustments.
- Latency Measurement: Recording the time taken to generate recommendations under different loads to ensure a responsive system.

7. RESULTS AND DISCUSSION

The system's performance is evaluated based on standard recommendation system metrics, including precision, recall, F1-score, and mean absolute error (MAE).

From the results, the hybrid model outperforms both individual models, providing more accurate and diverse recommendations. The content-based model struggles with new users due to a lack of prior data, while collaborative filtering performs well when sufficient user interactions are available. The hybrid approach mitigates both these issues, making it the most effective model for real-world deployment.

8. CONCLUSION

This paper presents a machine learning-based Movie Recommender System that integrates content-based and collaborative filtering approaches to enhance recommendation accuracy. The experimental results demonstrate that the hybrid model provides better precision, recall, and overall performance [10] compared to standalone models.

9. FUTURE SCOPE

Future enhancements to the system could include:

1. **Deep Learning Integration:** Using neural networks (autoencoders, for example) to increase the accuracy of recommendations.
2. **Real-Time User Feedback:** Using dynamic feedback systems to continuously improve suggestions.
3. **Scalability Improvements:** Making the system more user-friendly for massive datasets.
4. **Cross-Platform Recommendations:** extending the model to suggest content from various categories, including music, TV series, and books.

By implementing these improvements, the recommender system can become more robust, scalable, and user-friendly, further enhancing personalized content discovery in digital platforms.

REFERENCES

[1] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, May). *Item-based collaborative filtering recommendation algorithms*. In Proceedings of the 10th International Conference on World Wide Web (pp. 285–295).

[2] Pazzani, M., & Billsus, D. (2007). *Content-based recommendation systems*. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), The Adaptive Web (pp. 325–341). Springer, Berlin, Heidelberg.

[3] Burke, R. (2002). *Hybrid recommender systems: Survey and experiments*. User Modeling and User-Adapted Interaction, 12(4), 331–370.

[4] Gómez-Uribe, C. A., & Hunt, N. (2015). *The Netflix recommender system: Algorithms, business value, and innovation*. ACM Transactions on Management Information Systems, 6(4), Article 13.

[5] Salton, G., & Buckley, C. (1988). *Term-weighting approaches in automatic text retrieval*. Information Processing & Management, 24(5), 513–523.

[6] Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[7] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.

[8] Streamlit Inc. (2024). *Streamlit Documentation*. Available online: https://docs.streamlit.io

[9] Hipp, D. R. (2024). *SQLite Documentation*. Available online: https://sqlite.org

[10] Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). *Collaborative filtering recommender systems*. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), The Adaptive Web (pp. 291–324). Springer, Berlin, Heidelberg.