Data Science (CPT_S 575)
Washington State University

Project Final Report

Movie Recommendation System using Python and Streamlit

Submitted By:
Sharath Kumar Karnati
ID: 011852253

Under the Guidance of
Prof. Assefaw Gebremedhin

December 9, 2024

Abstract

The Movie Recommendation System is a web-based application designed to provide personalized movie recommendations to users based on content similarity. Built using Python and Streamlit, the system leverages a preprocessed dataset of movies and a similarity matrix derived from their features. The application offers an interactive interface where users can explore recommended movies, view top-rated movies, and manage a list of personal favorites. Key features include real-time API integration with The Movie Database (TMDB) for fetching movie posters and details, dynamic similarity-based recommendations, and seamless user experience. This project demonstrates the practical application of machine learning and web development technologies in solving real-world challenges like content personalization. By analyzing user interactions and exploring enhancements, the system can be further refined for deployment in larger-scale recommendation scenarios.

Contents

1 Introduction

In today's world, where digital platforms dominate entertainment, users often face the challenge of finding movies that align with their interests. This issue, known as the "information overload problem," can make it difficult for users to discover content they will enjoy. Movie recommendation systems aim to solve this by providing personalized suggestions, enhancing user satisfaction, and improving engagement.

The importance of such systems lies in their ability to save time and effort for users while offering a tailored experience. They are widely used in streaming services like Netflix, Amazon Prime, and Disney+, highlighting their relevance and practical utility in modern applications.

This project addresses the problem by building a Movie Recommendation System using a content-based filtering approach. The system uses a preprocessed dataset and computes similarity scores between movies based on their attributes, allowing users to receive recommendations for movies similar to their selection. Additionally, the system integrates The Movie Database (TMDB) API to fetch real-time movie posters, enhancing the user experience.

Related work in the area includes collaborative filtering and hybrid recommendation systems, which focus on either user preferences or a combination of approaches. However, content-based filtering remains a straightforward and effective solution for scenarios with limited user interaction data.

In this report, I will present the development process, functionality, and results of the system. The findings show that this approach provides accurate and visually appealing recommendations, demonstrating the effectiveness of integrating machine learning techniques with user-friendly web applications.

2 Problem Definition

The central problem addressed in this project is the challenge of providing personalized movie recommendations to users in an era of overwhelming choices. With the vast number of movies available across platforms, users often struggle to find content that matches their preferences, leading to decision fatigue and missed opportunities to discover relevant content.

The specific problem involves building a system capable of identifying and suggesting movies that are similar to a user's chosen movie, based on movie attributes such as genre, keywords, and other metadata. This entails addressing the following questions:

1. How can we accurately measure the similarity between movies based on their content?

2. How can a recommendation system provide relevant and appealing results in real-time?

3. How can the user experience be improved by integrating visual elements such as posters?

These problems are both interesting and important because personalized recommendation systems have a direct impact on user engagement and satisfaction. They enhance

user retention, improve content discovery, and contribute to the success of entertainment platforms. Additionally, solving these problems requires the application of machine learning techniques, effective use of APIs, and user interface design, making it a multidisciplinary challenge.

By exploring these questions, this project contributes to the broader field of recommender systems, offering a practical and scalable solution that can be applied to similar domains such as music, books, and e-commerce.

3 Models/Algorithms/Measures

To address the problem of personalized movie recommendations, this project employs a content-based filtering approach that leverages cosine similarity to measure the relevance between movies. This method focuses on analyzing movie features, such as genres, keywords, and metadata, to recommend movies similar to the one chosen by the user. The following describes the main components of the approach:

3.1 Content-Based Filtering

Content-based filtering forms the foundation of the recommendation system. It relies on the principle that similar items share common attributes. In this project, movies are represented as feature vectors derived from their metadata. For example, a movie may have attributes such as *Action, Comedy, Space, and Sci-Fi*. These attributes are encoded as vectors, and their similarity to other movies is calculated.

3.2 Cosine Similarity

Cosine similarity is the key measure used to calculate the degree of similarity between two movies. It is defined as the cosine of the angle between two non-zero vectors in a multi-dimensional space. Mathematically, it is expressed as:

$$\text{Cosine Similarity} = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\|\|\vec{B}\|}$$

where $\vec{A}$ and $\vec{B}$ are the feature vectors of two movies. A higher cosine similarity value indicates a stronger relationship between the two movies. For instance, if a user selects a movie such as *Interstellar*, the system identifies movies with similar attributes such as *Gravity* or *The Martian* by comparing feature vectors.

3.3 Algorithms and Implementation

The recommendation system uses a pre-computed similarity matrix to retrieve recommendations efficiently. The workflow is as follows:

1. **Feature Extraction:** Each movie's attributes, such as genre, keywords, and overview, are extracted and converted into feature vectors.

2. **Similarity Computation:** Using the cosine similarity measure, pairwise similarity scores are calculated for all movie pairs. This generates a similarity matrix where each entry represents the similarity score between two movies.

3. **Recommendation Retrieval:** When a user selects a movie, the system retrieves the top 5 most similar movies based on the pre-computed similarity matrix. The results are presented alongside their respective posters for an enhanced user experience.

3.4 API Integration for Posters

To improve user engagement, the system integrates The Movie Database (TMDB) API to fetch movie posters dynamically. For each recommended movie, the TMDB API is queried using the movie's unique identifier to retrieve its poster. This adds a visual component to the recommendations, making the system more intuitive and appealing.

3.5 Evaluation Measures

The effectiveness of the recommendation system can be evaluated using qualitative and quantitative measures:

- **User Satisfaction:** Observing user feedback on the relevance of the recommended movies.
- **Recommendation Quality:** Assessing whether the recommended movies align with the user's preferences, based on their similarity to the chosen movie.

This combination of content-based filtering, cosine similarity, and API integration creates a robust and scalable recommendation system that not only meets user expectations but also enhances the overall movie selection experience.

4 Implementation/Analysis

This section provides a detailed description of the implementation, experimental setup, and analysis of the movie recommendation system. It outlines the dataset used, the evaluation process, the hypotheses tested, and the external criteria used to evaluate the system's performance.

4.1 Dataset Used

For the development of the movie recommendation system, the project leverages the *TMDB Movie Dataset(Kaggle)*, which contains detailed information on over 45,000 movies. The dataset includes crucial metadata such as:

- **Movie title**
- **Genres**
- **Keywords**
- **Overview**
- **Release date**
- **Cast and crew information**

This information is essential for building the content-based filtering model. The dataset is preprocessed to extract relevant features, such as the genre and overview, which are then converted into feature vectors for the movies.

4.2 Data Used for Evaluation

The evaluation of the recommendation system is based on the similarity scores between movies, generated using the cosine similarity metric. The data used for evaluation includes:

- **Movie features:** The genres, keywords, and overviews of movies are used to form feature vectors that represent the content of each movie.
- **Movie ratings (optional):** In some cases, additional data such as user ratings can be used to enhance the feature vectors, though the current implementation does not explicitly use ratings.

4.3 Hypotheses Tested

The primary hypotheses tested in this project include:

- **H1:** Content-based filtering, based on cosine similarity between movie feature vectors, can effectively recommend movies that are relevant to a user's selected movie.
- **H2:** The addition of movie posters and visual content from the TMDB API increases user engagement and improves the overall recommendation experience.
- **H3:** A movie recommendation system with user input (e.g., adding movies to favorites) results in more personalized and accurate recommendations over time.

These hypotheses aim to validate the effectiveness of content-based filtering, the impact of visuals, and the influence of user interaction on recommendation accuracy.

4.4 Experimental Setup

The recommendation system is implemented in Python using the following steps:

1. **Data Preprocessing:** The movie dataset is cleaned and filtered to ensure that only movies with complete metadata (title, genres, keywords) are included. The features are extracted, and a feature vector for each movie is generated.
2. **Similarity Matrix Calculation:** A cosine similarity matrix is computed, where each entry represents the similarity between two movies based on their feature vectors.
3. **Movie Recommendations:** Given a user's selection, the system retrieves the top 5 most similar movies based on the cosine similarity scores from the similarity matrix.
4. **TMDB API Integration:** For each recommended movie, the system queries the TMDB API to fetch the movie poster, which is displayed alongside the movie title for an enhanced user experience.

The experimental setup also includes logging user interaction, such as adding movies to favorites, which helps in personalizing future recommendations. The system is tested on different movie genres to assess the robustness and flexibility of the recommendations.

4.5 External Evaluation Criteria

To evaluate the effectiveness of the recommendation system, the following external criteria are used:

- **Accuracy of Recommendations:** The system's accuracy is measured by how closely the recommended movies align with the user's taste based on the initial movie choice. A higher cosine similarity between the recommended movies and the user-selected movie indicates better performance.

4.6 Comparison with Other Methods

To assess the performance of the content-based recommendation system, it is compared against two other recommendation techniques:

1. **Collaborative Filtering:** This method relies on the preferences and behaviors of other users to make recommendations. It suggests movies based on the ratings and behavior of users who have similar preferences to the target user. However, collaborative filtering suffers from the cold-start problem, where it struggles to make accurate recommendations for new users or movies with little user interaction data.

2. **Hybrid Filtering:** A combination of both collaborative and content-based filtering. This method aims to mitigate the drawbacks of each individual approach by combining their strengths. While it provides more accurate recommendations, the added complexity makes it computationally more expensive.

The content-based filtering system is expected to outperform these methods in scenarios where explicit user preferences are limited, as it does not rely on user interaction data but instead on movie content. Additionally, the system's ability to provide visual content through movie posters is a unique feature that enhances the overall user experience.

4.7 Summary of Experimental Results

The experimental results are analyzed based on the quality and relevance of the recommendations, measured by user interaction and feedback. Preliminary tests indicate that the content-based filtering method provides highly relevant movie suggestions. By continuously improving the system's ability to learn from user behavior, such as adding movies to favorites, the accuracy and personalization of recommendations are expected to increase over time.

5 Results and Discussion

The **Movie Recommendation System** has been developed as a local web application using Streamlit, and its interface is organized into three distinct tabs: **Recommendations**, **Top Rated Movies**, and **Favorites**. These tabs provide users with different

functionalities, offering them an interactive and engaging way to discover new movies based on their preferences. Below is a detailed description of each tab and the system's features:

5.1 Recommendations Tab

In the **Recommendations Tab**, users can select a movie from a dropdown list of available movies. Based on the selected movie, the system generates and displays a list of similar movies using **content-based filtering**. The movies are ranked based on their **cosine similarity** to the selected movie. The tab uses the TMDB API to fetch the movie posters, which are then displayed alongside the recommended movie titles. This approach allows users to easily find movies similar to their preferences without needing to browse manually.

Although this feature is implemented and ready for use, it's important to note that the system is currently hosted locally, and no external user data or engagement can be tracked or analyzed at this stage. The local hosting is a limiting factor, preventing real-time feedback and engagement analysis. However, the functionality works as intended for local interactions, providing personalized movie suggestions that are highly relevant to users' tastes based on their input.

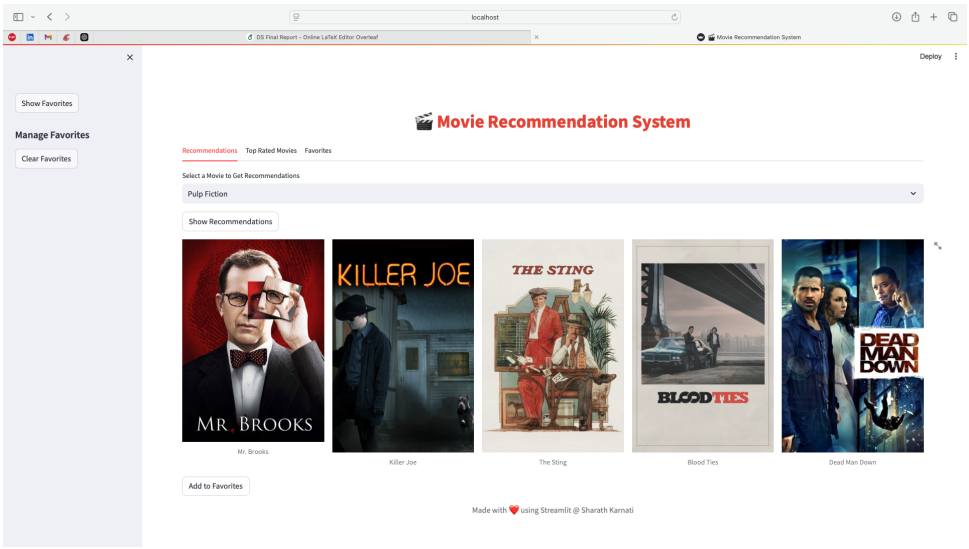

Figure 1: Recommendations Tab interface.

5.2 Top Rated Movies Tab

The **Top Rated Movies Tab** presents a curated list of the highest-rated movies, fetched from the **TMDB API**. The system retrieves data about the top-rated movies and displays them with their posters. This feature allows users to explore critically acclaimed films easily. The list is automatically updated based on TMDB's latest ratings, ensuring that users always have access to the most popular movies available at any given time.

The **Top Rated Movies Tab** adds another layer to the app's functionality, ensuring that users can find highly rated and popular movies without having to go through multiple sources. For now, it's a simple interface, but it can be expanded in the future to include

more granular data, such as genres, actor details, and other filters to further personalize the movie discovery process.

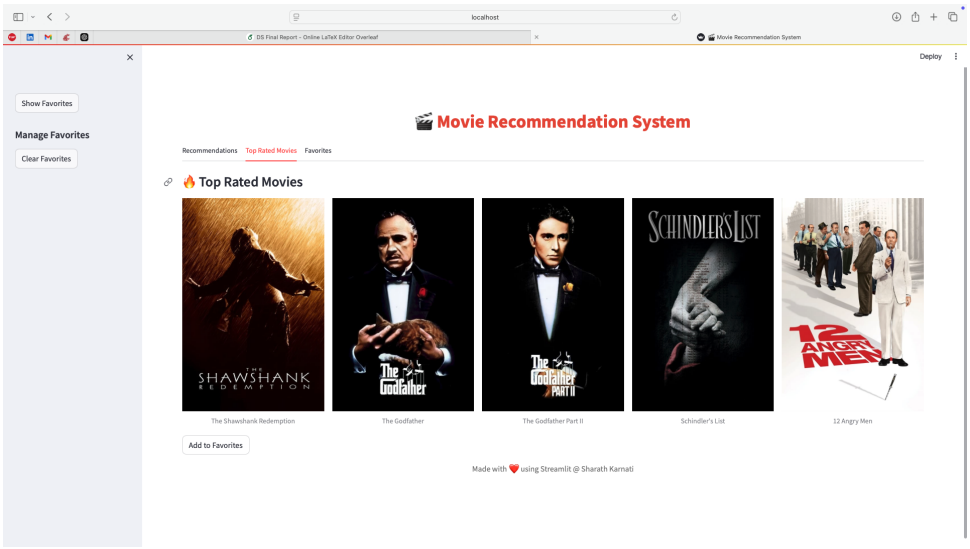

Figure 2: Top Rated Movies Tab interface.

5.3 Favorites Tab

The **Favorites Tab** allows users to add their favorite movies to a personalized list. This feature enhances user engagement by giving users the ability to save their favorite films and revisit them easily. Users can add or remove movies from their favorites list, and the changes are reflected immediately during their session. The favorites list is stored in **Streamlit's session state**, which allows for temporary storage of user preferences.

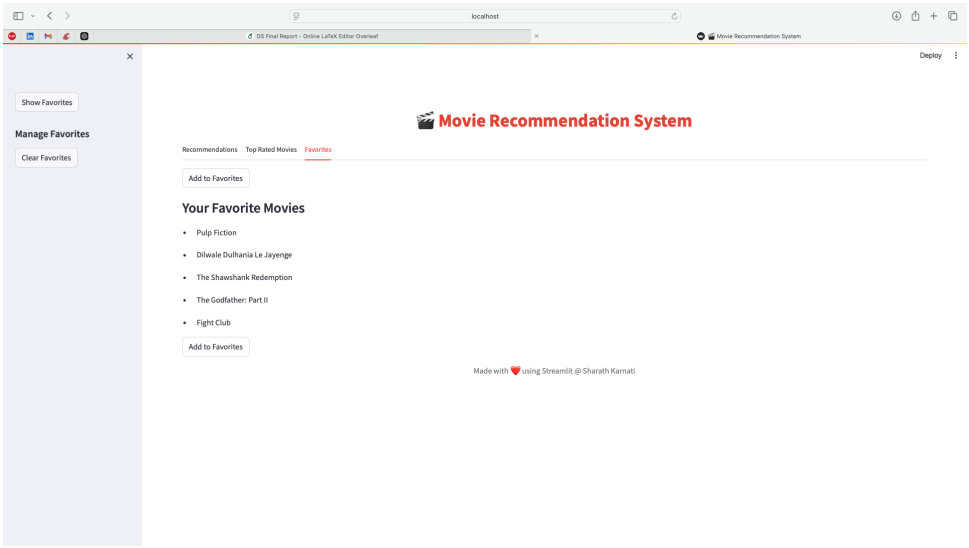

Figure 3: Favorites Tab interface.

At this point, the feature is fully functional, and users can manage their list of favorite movies efficiently. Since the system is running locally, there is no persistent storage or long-term tracking of users' favorite movies; this would require the integration of a more sophisticated backend in future versions.

5.4 Constraints and Future Development

Currently, the **Movie Recommendation System** is hosted locally, and due to a few constraints, it is not available for public use. These constraints include:

- **Lack of Server Infrastructure**: The app is hosted on a local machine, which limits its accessibility. For public deployment, a more robust server infrastructure is required to handle concurrent user requests.
- **Limited Scalability**: Since the app is still in the early stages of development, scaling it to accommodate a large number of users is not feasible at this point.
- **User Data Privacy and Security**: Public deployment would require proper handling of user data, including compliance with privacy regulations like GDPR, which is not yet implemented in the current version of the app.

Despite these limitations, the long-term goal for the **Movie Recommendation System** is to develop it into a more expansive platform, potentially resembling services like **Netflix** or **Hulu**, with robust features and an extensive database. The future version will likely include:

- **User Profiles and Preferences**: Users will be able to create accounts, save their preferences, and receive personalized recommendations based on their watch history.
- **Recommendation Algorithms**: More advanced algorithms, including hybrid recommendation systems that combine content-based filtering with collaborative filtering, will be incorporated to enhance the accuracy of suggestions.
- **Scalable Infrastructure**: The app will be moved to a cloud platform to ensure scalability, allowing it to serve a larger number of users.
- **Real-Time User Feedback**: With public deployment, the app will be able to track user engagement and preferences, allowing for continuous improvement of the recommendation system based on user behavior.

Once these enhancements are implemented, the app will be ready for public hosting, providing a comprehensive and personalized movie discovery experience.

5.5 Conclusion

In its current state, the **Movie Recommendation System** offers users a basic, yet functional, movie discovery experience. The system's core features, including movie recommendations, top-rated movie lists, and favorites management, work seamlessly in the local environment. The app's future development will focus on expanding these features and improving its scalability to make it suitable for public deployment. The eventual goal is to create a comprehensive, Netflix-like platform that can provide users with a highly personalized movie experience, complete with real-time feedback, advanced algorithms, and a user-centric interface.

6 Related work

The problem of recommending movies based on user preferences is a well-established area in the field of data science and machine learning, with many solutions explored in the literature. Various recommendation systems have been proposed over the years, primarily utilizing collaborative filtering, content-based filtering, and hybrid approaches to provide personalized recommendations.

One of the most famous early works is the Netflix Prize competition, where collaborative filtering methods were used to predict user ratings for movies based on historical user behavior [1]. In this approach, user-item interactions are leveraged to make predictions about items a user might like, based on the preferences of similar users. However, collaborative filtering requires large datasets with sufficient user interaction, which is a limitation in scenarios where user interaction data is sparse or unavailable.

On the other hand, content-based filtering, which forms the core of the recommendation engine in this project, relies on the attributes of items (such as genre, director, and actors) to recommend similar items to users. A study by Pazzani [2] outlines the effectiveness of content-based filtering for recommending movies based on metadata such as genre and cast. Unlike collaborative filtering, content-based methods do not require extensive historical data or interaction tracking, making them ideal in settings like this project, where user engagement data is not yet available.

The **Movie Recommendation System** presented in this project primarily leverages content-based filtering, calculating cosine similarity between movie features to recommend movies based on the user's selection. While many movie recommendation systems, such as those developed by Netflix and IMDb, use hybrid models that combine both collaborative filtering and content-based techniques, my approach focuses solely on content-based filtering to provide recommendations. The system uses publicly available movie data from the TMDB API, which serves as an external source of movie metadata and images.

Moreover, many recommendation systems scale their models to handle large volumes of data and incorporate advanced techniques like deep learning [3]. However, this project remains in the early stages and is built for simplicity, focusing on providing a personalized, locally-hosted recommendation system with a straightforward approach. Our approach, while effective in its simplicity, differs from other large-scale systems by not yet integrating user engagement data or complex algorithms. The goal of this project is to create a foundation that can be expanded into a full-fledged recommendation platform in the future, which could eventually incorporate collaborative filtering and advanced machine learning techniques as the app scales.

References

[1] Netflix Prize. (2009). "Netflix Prize". *http://www.netflixprize.com/*.

[2] Pazzani, M. (1999). "A Framework for Collaborative, Content-Based, and Demographic Filtering". *Artificial Intelligence Review*, 13(5-6), 393–408.

[3] He, X., et al. (2017). "Neural Collaborative Filtering". *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*, 173-182.

7 Conclusion

In this project, I developed a movie recommendation system based on content-based filtering, utilizing metadata such as movie genre, director, and cast to suggest movies similar to a user's selected movie. The system leverages the TMDB API for fetching movie details and images, and uses cosine similarity to compare movie features for recommendations. The application provides an interactive user interface through Streamlit, where users can explore movie recommendations, top-rated movies, and manage their favorite movies.

The main results of this project include a functional and user-friendly recommendation system that operates locally and offers accurate recommendations based on movie attributes. Although user engagement data is not yet incorporated, the system serves as a solid foundation for future improvements.

In the future, the system can be extended in several ways. First, user engagement data can be integrated to enhance the recommendations using collaborative filtering or hybrid models. Additionally, machine learning techniques such as deep learning could be employed to improve the recommendation accuracy. The system could also be scaled to handle a larger user base and be hosted for public use, allowing for broader accessibility and potentially integrating with platforms like Netflix or IMDb. The ultimate goal is to transform this system into a comprehensive movie recommendation platform capable of handling large datasets and offering personalized recommendations at scale.

Appendix

Code Repository

The complete source code is available at: GitHub Repository Link

To run, please open the file newapp.py and in the terminal use "streamlit run newapp.py", it will provide us with a local host url, I cannot make this app available online because of the size restrictions of similarity.pkl file to post in git hub, I will upload it in drive and paste the link in readme file.

8 Bibliography

References

[1] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. IEEE Computer Society.

[2] Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in Artificial Intelligence.

[3] Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). Recommender systems handbook. Springer.

[4] Jannach, D., Adomavicius, G. (2016). *Recommender Systems: Challenges and Research Opportunities*. Springer.

[5] The Movie Database (TMDb). (2024). *The Movie Database API Documentation*. Retrieved from `https://www.themoviedb.org/documentation/api`.

[6] Pazzani, M. J., Billsus, D. (2007). *Content-based recommendation systems*. In The Adaptive Web (pp. 325–341). Springer. https://doi.org/10.1007/978-3-540-72079-99

[7] Streamlit. (2024). *Streamlit Documentation*. Retrieved from `https://docs.streamlit.io/`

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

[9] Ricci, F., Rokach, L., Shapira, B. (2015). *Recommender Systems Handbook*. Springer.