A REPORT ON

"**Movie Magic: Personalized Recommendations through User and Item-Based Filtering**"

Computer Science, College of Engineering and Applied Sciences

Advisor: Prof. Rohit J Kate

Submitted By:

Lourdu Rohith Reddy Gali

Email: - lgali@uwm.edu

ABSTRACT

This project aims to design a personalized movie recommendation system using collaborative filtering algorithms and the Flask framework. The system uses a dataset from the Yahoo Research Webscope database, which contains user ratings and descriptive movie content. Data cleaning was performed to remove unnecessary columns, blank and duplicate values. User-based and item-based filtering models were built using cosine similarity and k-nearest neighbors algorithms, respectively. The Flask web application allows users to view recommended movies based on their ratings and preferences. This project provides an accurate and personalized movie recommendation system that can improve user experience, increase engagement, and ultimately generate more revenue for companies.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and appreciation to my professor Rohit J Kate, for his invaluable guidance, support, and knowledge during my project work. His teachings on machine learning have been immensely insightful and have greatly contributed to my learning and growth in this field. I am grateful for his patience, dedication, and expertise in teaching and making complex concepts easily understandable. I feel happy to have been his student and am confident that the knowledge and skills I have gained under his guidance will prove to be invaluable in my future endeavors.

INTRODUCTION

Recommendation systems have revolutionized the way companies engage with their customers. By analyzing customer behavior and past purchases, these systems can predict user preferences and suggest suitable options. This personalized approach has proven to be an effective tool for increasing customer satisfaction, loyalty, and revenue. However, creating an accurate recommendation system that caters to individual preferences can be challenging. Each user has unique tastes and preferences that may depend on various factors, such as their mood, the occasion, or the reason for their purchase. Therefore, companies need to continually improve their recommendation systems to meet users' evolving needs.

One area where recommendation systems have become increasingly popular is in the entertainment industry. Movies and TV shows have become an integral part of our lives, and many people rely on recommendation systems to help them find new content to watch. However, designing an effective movie recommendation system is not as simple as it seems. The system needs to analyze millions of data points, identify patterns and similarities in user preferences, and make accurate predictions. Moreover, movie recommendations need to be personalized and cater to individual tastes, which can be a challenging task. In this project, we focus on designing a movie recommendation system that uses collaborative filtering algorithms and the Flask framework to provide accurate and personalized recommendations to users.

By implementing this movie recommendation system, we aim to improve the user experience by providing relevant and personalized movie recommendations. The system considers past movie ratings given by various users to suggest suitable options to the user. Collaborative filtering algorithms analyze user behavior and ratings to identify patterns and similarities in

their preferences. The Flask framework is used to develop a website that displays recommendations to the user. This project can help companies increase user engagement, satisfaction, and revenue by providing a more personalized approach to movie recommendations. Additionally, this project can also serve as a framework for designing recommendation systems for other industries, such as music, books, and e-commerce.

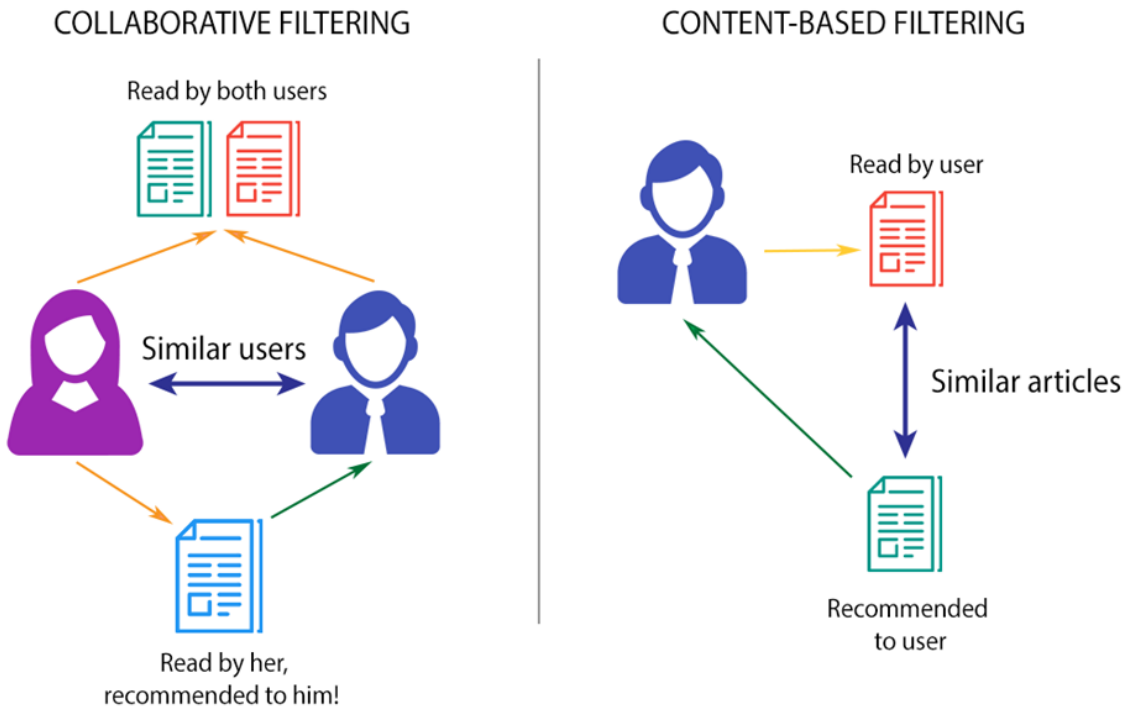

Fig-1: User-Based and Item-Based Filtering.

Overview

A. Recommendation systems

Recommendation systems can be classified into three main types: collaborative filtering, content-based filtering, and hybrid systems. Collaborative filtering systems analyze users' behavior and preferences to predict what they would like based on their similarity with other users. There are two types of collaborative filtering systems: user-based and item-based recommenders. User-based recommenders predict a user's preference based on the preferences of similar users, while item-based recommenders suggest items that are similar to those previously rated highly by the user.

Content-based filtering systems consider the item's description and features, along with the user's preferences, to provide recommendations. These systems suggest items that are similar in content to items that the user has already shown interest in.

Hybrid recommendation systems combine both collaborative and content-based filtering methods. These systems first perform collaborative and content-based predictions separately, then combine the results to provide recommendations. Hybrid systems aim to address the limitations of each method by leveraging their strengths. By using multiple methods, hybrid systems can provide more accurate recommendations and better serve the diverse needs of users.

B. Flask

Flask is a web framework for Python that provides tools, libraries, and technologies for building web applications. Flask is lightweight, flexible, and easy to use, making it an

excellent choice for developers who want to create web applications quickly and efficiently. One of the most powerful features of Flask is its template engine, Jinja2. Jinja2 is a powerful and flexible template engine that allows developers to create dynamic HTML pages with ease. With Jinja2, developers can easily build templates that can be customized based on user input or data from a database. Jinja2 also includes features such as template inheritance, macros, filters, and much more, making it an extremely versatile tool for building web applications. By using Flask and Jinja2, developers can create dynamic and responsive web applications quickly and easily, saving time and effort in the development process.

C. Related Work

The rapid expansion of the internet has led to an abundance of applications, making it challenging for users to easily locate what they need. Recommendation systems have been widely utilized in numerous domains to provide customers with suggestions based on their predicted preferences. These systems are commonly implemented in various areas such as music, movies, news, grocery shopping, travel, online dating, books, restaurants, and e-commerce websites, among others.

[1] The paper describes the implementation of a movie recommendation system using collaborative filtering. The system was developed using Apache Mahout, which takes into account the movie ratings given by users to provide movie suggestions. Collaborative filtering is a technique that analyzes user behavior and preferences to identify patterns and similarities in their preferences, and then recommends items based on those patterns. By using this approach, the system can provide personalized recommendations to users based on their past movie ratings.

IMPLEMENTATION

Dataset:

The movie dataset used in my experiment is obtained from the Yahoo Research Web scope database. The database provides two files, namely Yahoo! Movies User Ratings and Yahoo! Descriptive Content Information, v.l.O. The Yahoo! Movies Users Rating file contains 211231 records and consists of User ID, Movie ID and Ratings. The Yahoo! Movies Descriptive Content Information file contains 106959 records and consists of Movie ID, Title, Genre, Directors, Actors and so on.

Data Cleaning:

The Movies Descriptive Content Information file had around 40 columns, but only a few were needed for my experiments, so I removed the unnecessary ones. There were also many blank and duplicate values in the dataset, which I addressed. Moreover, I found some entries in the Movies Users Ratings file that didn't match any movie in the Movies Descriptive Content Information file, and I eliminated those entries to simplify the processing.

Model Building:

For User-based filtering, I used the cosine similarity to determine the similarity between users' ratings, hence the preference. The item-based filtering is implemented using the k-nearest neighbors model using the cosine similarity metric and the brute-force algorithm.

Libraries

sklearn.neighbors: This library in scikit-learn provides functionality for performing various neighbor-based learning tasks, such as k-nearest neighbors classification and regression. It includes the NearestNeighbors class, which allows you to find the nearest neighbors of a given data point based on a specified distance metric.

flask: Flask is a lightweight web framework for Python that allows you to build web applications easily and quickly. It provides functionalities for routing, request handling, and rendering templates. Flask is commonly used for building RESTful APIs and web applications.

pandas: pandas is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrame and Series, which allow you to efficiently handle and manipulate structured data. pandas offers a wide range of functionalities for data cleaning, transformation, and analysis, making it a popular choice for data scientists and analysts.

*sklearn.metrics.pairwise.cosine_similarity***:** This function in scikit-learn calculates the cosine similarity between two or more vectors. It measures the similarity between vectors based on the cosine of the angle between them, ranging from -1 (completely dissimilar) to 1 (completely similar). Cosine similarity is commonly used in information retrieval, recommendation systems, and text mining tasks to compare the similarity between documents or vectors.

Flask Web Application:

The Flask application is designed to display movie recommendations based on filtering methods. A Pandas DataFrame stores filtered movie information that is then rendered through a Jinja template. The user can view the generated recommendations for the selected filtering method. The Flask application offers two filtering options, namely user-based and item-based filtering. If the user selects the user-based option, they must input their user ID, which is then used to generate recommendations based on the behavior and preferences of similar users. Collaborative filtering methods are employed to generate such recommendations. Alternatively, if the user chooses item-based filtering, they are required to enter the movie item. The system will then generate recommendations based on similarity between the given movie and other movies in the database. Content-based filtering is used to generate recommendations for item-based filtering. Overall, the Flask application is a useful tool for generating movie recommendations based on different filtering methods.

Results

Figure 2 shows an example of the Flask application in action. The user has selected "user-based" from the dropdown menu and input their user ID as "1". The system then generates recommendations based on user-based filtering and displays them on the screen. This allows the user to quickly and easily find movies that match their preferences and interests.

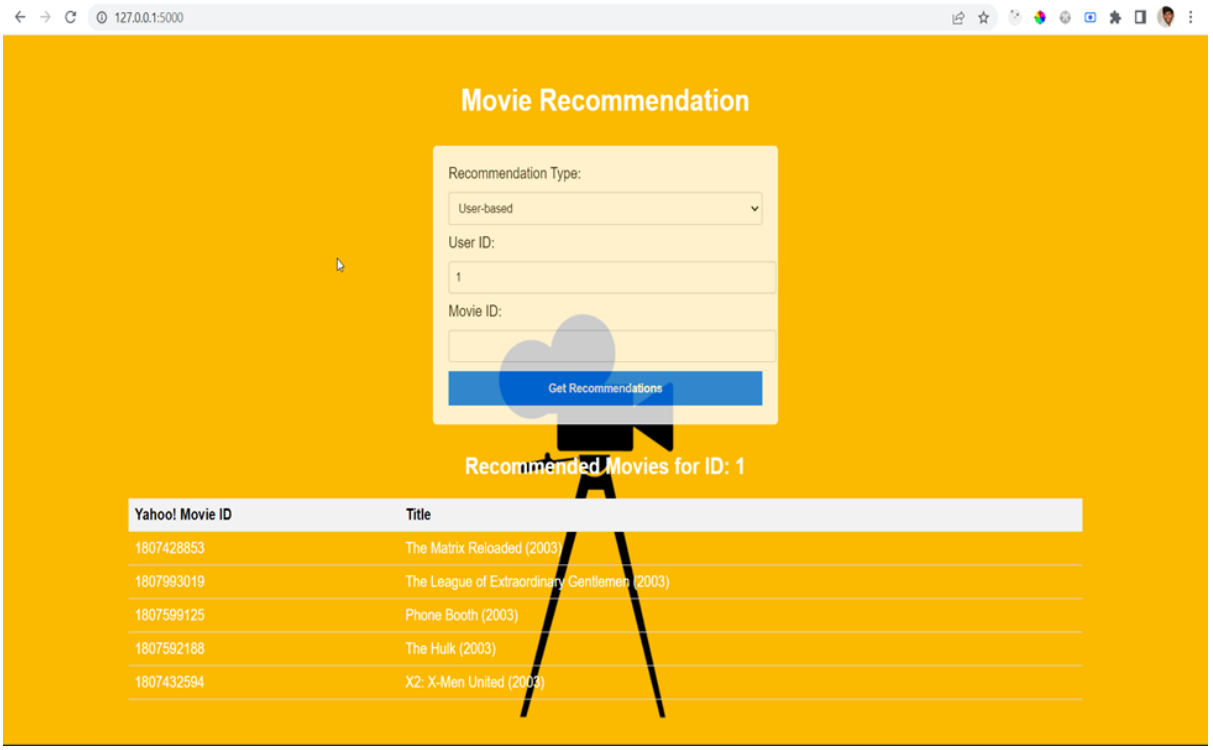

Fig-2: User-based filtering

In Figure 3, the user selects item-based from the dropdown menu and enters the movie ID as 1800361191. The system then generates recommendations based on item-based filtering, which takes into account the features of the selected movie and suggests similar movies to the user. The resulting recommendations are displayed to the user, allowing them to select which movie they would like to watch next based on their interests and preferences.

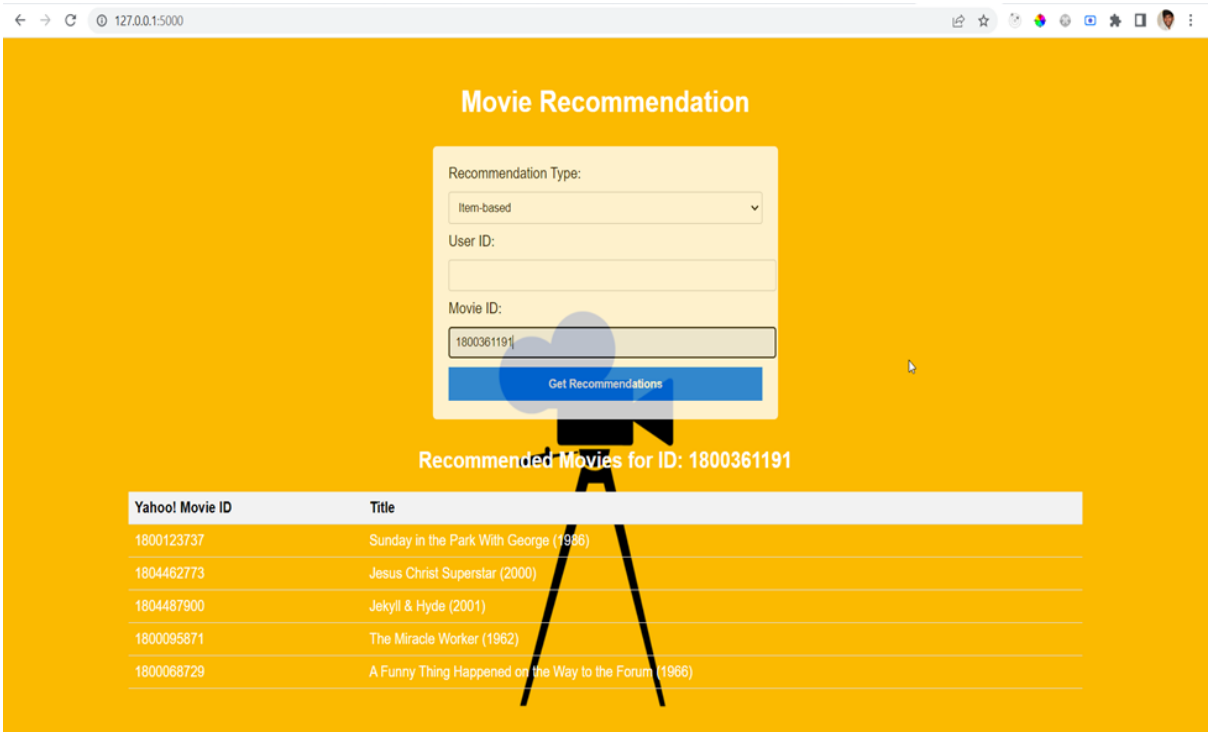

Fig-3: Item-based filtering

In figure 4, the user selected user-based filtering from the dropdown but entered an incorrect user ID, 100000000. As a result, the system was unable to find any data related to that user and could not generate any recommendations. The user was presented with a message informing them that no recommendations could be generated based on the incorrect input. This highlights the importance of accurate input in the recommendation system to receive relevant and personalized recommendations.

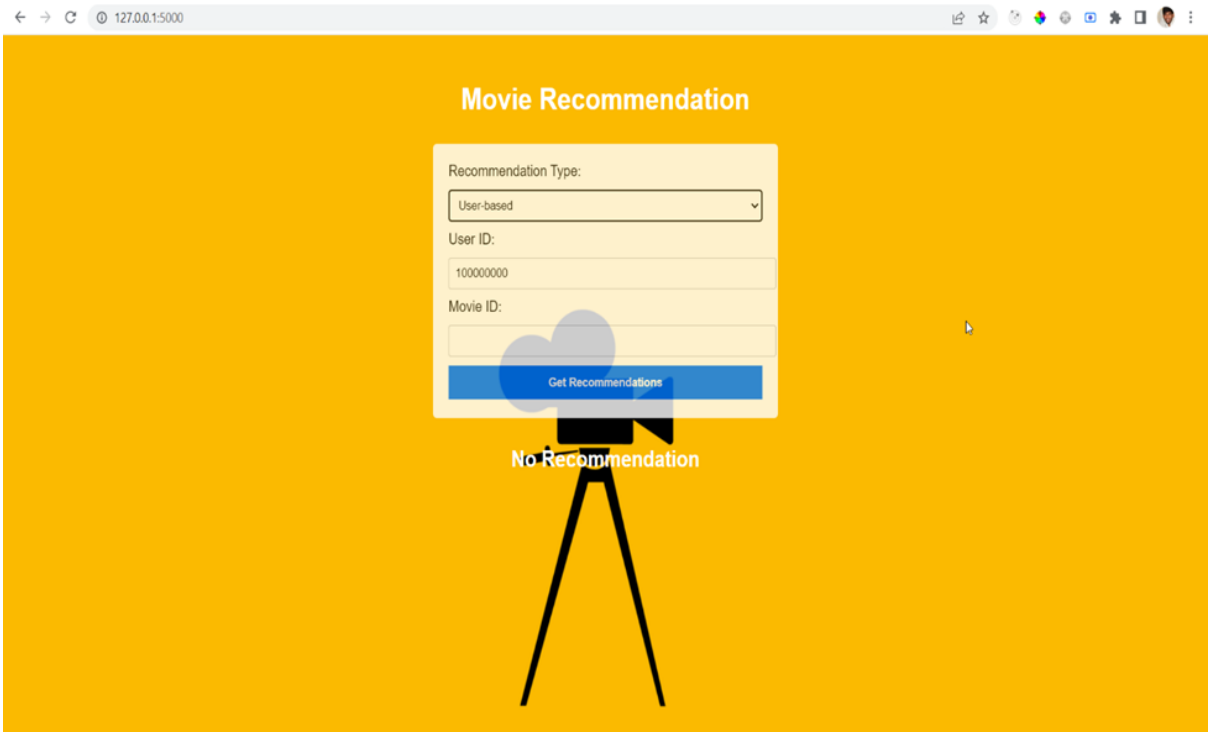

Fig-4: No recommendations Found

Major Challenges

With recommender systems, the following issues are always raised. I evaluate my system based on these issues and propose an implementation design to resolve them. Firstly, the New User Problem is concerned with the scenario when a new user is added to the recommender system. There are no ratings provided by him/her for any movie in the system yet. This is also called a User Cold Start. One simple possible resolution can be to recommend top-rated movies or the most recently added movies to this new user. Secondly, there is a concern that no new item is recommended. It can be understood as an Item Cold Start problem. When a new movie is added to the system, it does not have any ratings associated with it. How can it be discovered and recommended? One resolution can be to recommend movies similar to the genre of top-rated movies. If the new movie falls into that genre, it will get discovered. However, in this resolution, I will need to build a system based on the genre of the movies. Keeping these known issues apart, either of the two approaches can be used based on the infrastructure. For item-based, the similarity computation is large of the order MxM (M: total movies) but since it is static, we can do it offline and re-compute it only after some threshold time. On the other hand, expensive to do at runtime as for each user, the neighborhood is dynamic with new ratings from different users. Therefore, the former requires cache data storage, latter requires a dedicated processing server.

Conclusions

In this project, I have implemented a movie recommendation system using collaborative filtering. This system is developed using Python and Flask framework and considers the ratings given to movies to provide movie suggestions.

REFERENCES

[1] C. -S. M. Wu, D. Garg and U. Bhandary, "Movie Recommendation System Using Collaborative Filtering," *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, 2018, pp. 11-15, doi: 10.1109/ICSESS.2018.8663822.

[2] PythonBasics. (n.d.). What is Flask? – Python Tutorial. Retrieved from https://pythonbasics.org/what-is-flask-python/

[3] M. Gupta, A. Thakkar, Aashish, V. Gupta and D. P. S. Rathore, "Movie Recommender System Using Collaborative Filtering," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 415-420, doi: 10.1109/ICESC48915.2020.9155879.

[4] B. Sarwar, G. Karypis, 1. Konstan , 1. Riedl, "Item-based collaborative filtering recommendation algorithms", Proceedings of the 10th international conference on World Wide Web, pp. 285-295, 2001