Movie recommendation app with cosine similarity and flask

P Vedhavyas[1], *V* Srikar[1*], *N* Ashwarda[1] and *Ruqqaiya* Begum[1]

[1]Computer Science & Engineering Department, Vardhaman College of Engineering, Hyderabad, Telangana, India.

Abstract. People are constantly busy with their professions, businesses, and other endeavours in the world in which we currently live. The majority of them find that watching movies is the greatest way to unwind during the limited free time they have between jobs. However, with so many movies available in different languages, choosing which one to watch may be a time-consuming task. The recommendation may be collaborative filtering or content-based. As the name implies, collaborative filtering bases its filtering method on the interactions between relevant user behaviour and that of other users. In order to correlate movies with recommendations, this study describes cosine similarity. To receive the top 5 recommendations, the user must enter the name of a movie they have previously enjoyed. We've also introduced a feature that allows the user to optionally provide the year data that they want the engine to use to propose movies that were published after that specific year. For example, if the user wants to filter recommendations for new movies, they can provide a recent year.

1 Introduction

A recommendation system is a kind of information filtering system that makes recommendations based on its ability to predict a stoner's interests. Recommendation systems have a great range of operations.[2] These have become less and less common over the past few years and are currently utilized on the majority of the websites we visit. Similar platforms' content ranges from images, music, videos, books, and tales on social media platforms to products on e-commerce websites, individuals on dating and professional websites, and Google search results. These systems are often useful for gathering data regarding druggies' decisions, which they can then utilize to improve their recommendations going forward. For example, Facebook may display colourful stories about your transactions in your news feed to determine the kinds of things that you find interesting. Every now and again, recommender systems can improve based on the training of a huge population. For example, if Amazon notices that many customers who purchase the most recent Apple MacBook also purchase a USB-C-to-USB extension, they may suggest the extension to a novice stoner who has recently added a MacBook to his collection. Drug users always expect

* Corresponding author : srikar149@gmail.com

high-quality recommendations because of the advancements in recommender systems. They have a low bar for unfit services, so they can still offer relevant recommendations. The stoner will also soon quit using a music streaming app if it isn't able to predict and play songs they enjoy. As a result, IT businesses are placing a lot of effort on refining their recommendation systems[3]. However, the issue is more complicated than it first appears. Every stoner has unique tastes and inclinations. Furthermore, a stoner's taste can differ according on a variety of things, including as their mood, the time of year, and the kind of exercise they engage in. For recommender systems, there are two primary methods that are widely employed. These are content-based filtering and collaborative filtering [12]. Here, we'll integrate the collaborative filtering approach with the Flask framework in Python. The process of collaborative filtering[8] involves gathering and analysing user activity data. This includes analysing the user's online behaviour and speculating about their preferences by comparing them to other users.

2 Literature Survey

There are many different methods and algorithms used in the literature on movie recommendation systems [1]. Collaborative filtering, matrix factorization, content-based recommendation, hybrid systems, and deep learning are important techniques [11]. Further research has been done on context-aware recommendation [8] and reinforcement learning [10]. These approaches' capacity to offer tailored recommendations is one of their strong points, but other drawbacks include scalability issues and the "cold start" issue for new users or things. Limited study on contextual data, real-time adaptability, and ethical considerations are among the gaps in the literature. By suggesting a content-based recommendation system that takes user context and real-time adaptation into account and uses Cosine Similarity and Flask, this work seeks to close some of these gaps.

One kind of information filtering system that can assist users in identifying interesting items from a vast and frequently overwhelming array of options is the recommender system [8]. Popular recommender systems include those for movies, which have been used to e-commerce websites, social networking platforms, and online streaming services, among other contexts.

Cosine similarity is a statistic used to compare the similarity of two vectors. Comparing two users' or two things' similarity is a common use case for recommender systems. Cosine similarity[14] can be used to compare the similarity between two consumers' movie ratings or two movies' features, like actor, director, or genre, in the context of movie recommender systems.

Python has a lightweight web application framework called Flask[13]. Because it is simple to learn and use and offers many helpful capabilities for developing web applications, like routing, templating, and database connectivity, it is frequently used to design recommender systems. There are still a lot of unanswered questions with cosine similarity and Flask-based movie recommender systems, despite some study in this field. For instance, further study is required to create more individualized and successful movie recommendation algorithms. Furthermore, further investigation is required to assess the effectiveness of movie recommender systems.

3 Machine Learning Methods

A machine learning technique called cosine similarity[12] can be used to calculate how similar two data points are to one another. Comparing two users' or two things' similarity is a common use case for recommender systems. Cosine similarity can be used to compare the likeness of two users' movie ratings or two movies' attributes, including cast, director, or genre, in the context of movie recommender systems.

It is necessary to first represent each data point as a vector in order to compute the cosine similarity between two data points. One possible way to display user movie ratings is as a vector, with each element of the vector denoting a user's rating for a certain film. Comparably, we can use a vector to represent the features of a video, with each element in the vector denoting a distinct feature.

The following formula can be used to determine the cosine similarity between data points once they have been represented as vectors:
cosine_similarity = dot(vector_a, vector_b) / (||vector_a|| * ||vector_b||)
where:

dot() is the dot product operator
||vector_a|| is the magnitude of vector_a
||vector_b|| is the magnitude of vector_b
A value between 0 and 1 represents the cosine similarity between two vectors. When two vectors have a cosine similarity of 1, it means they are the same, and when it is zero, it means they are orthogonal, or perpendicular to each other.[14]
The cosine similarity is explained in the graphic below.[]

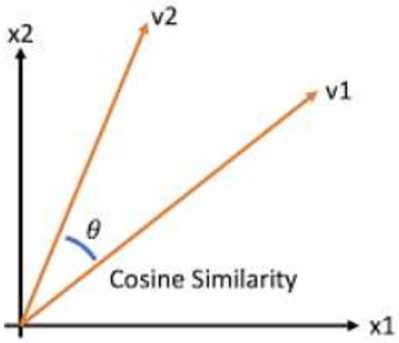

Fig. 1. Cosine Similarity

4 Proposed Methods

Cosine similarity is used by the suggested movie recommendation engine to gauge how similar user preferences or movie elements are. The system finds similar people or movies by computing the cosine similarity between user vectors and movie vectors. This allows for

the creation of individualized recommendations based on weighted ratings from the most similar users or movies

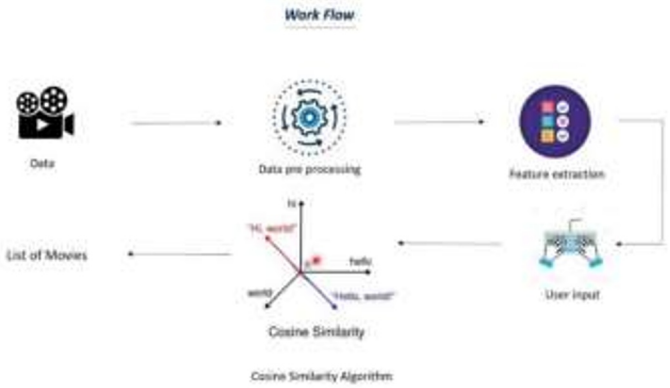

Fig. 2. Class Diagram

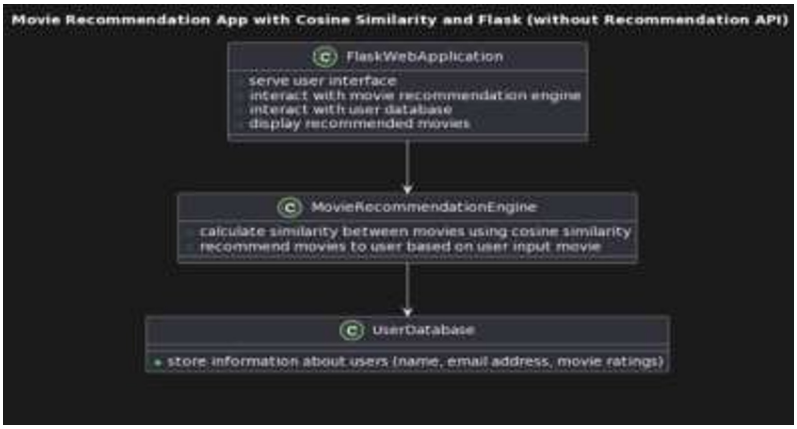

Fig. 3. Flowchart of Cosine Similarity Algorithm

The movie data is loaded by the program from a CSV file. By completing missing values and cleaning up the data, it carries out data preparation. It concatenates the genres, keywords, tagline, cast, and director features to produce a combined feature vector for every film. The concatenated feature vectors are converted into a sparse matrix using a Tfidf Vectorizer. It determines each pair of movies' cosine similarity. The application uses difflib.get_close_matches() to locate a near match in the movies data when a user provides the name of a movie. Next, the movie's index is obtained, and the cosine similarity matrix is used to determine which ten films are the most similar. In the end, the user receives the recommended movie list.

4.1 Here is a detailed description of each step of the app:

4.1.1. Load the movies data from a CSV file:

The app uses the pd.read_csv() function to load the movies data from a CSV file. The CSV file should contain the following columns:

title: The title of the movie
genres: A list of genres for the movie
keywords: A list of keywords for the movie
tagline: The tagline for the movie
cast: A list of cast members for the movie
director: The director of the movie

4.1.2. Perform data preprocessing:

The app performs the following data preprocessing steps:
Cleaning up the data: The app removes any whitespace from the beginning and end of each string value. It also converts all strings to lowercase.
Filling in missing values: The app fills in missing values for all features with an empty string.

4.1.3. Create a combined feature vector for each movie:

The app creates a combined feature vector for each movie by concatenating the genres, keywords, tagline, cast, and director features. The cosine similarity between movies will be computed using this composite feature vector.

4.1.4. Use a TfidfVectorizer to transform the combined feature vectors into a sparse matrix:

The app uses a TfidfVectorizer to transform the combined feature vectors into a sparse matrix. A matrix that has zeros for the majority of its elements is said to be sparse. This is because the combined feature vectors will be very sparse, since most movies will only share a few of the same keywords, genres, etc.

4.1.5. Determine the cosine similarity between every pair of videos:

The application computes the similarity between every pair of movies using the cosine similarity metric. By computing the angle between two vectors, the cosine similarity metric determines how similar they are to one another. The similarity between the two vectors is shown by a greater cosine similarity score.

When a user enters a movie name, the app finds a close match in the movies data using difflib.get_close_matches(). When a user enters a movie name, the app uses the difflib.get_close_matches() function to find a close match in the movies data. This function returns a list of strings that are most similar to the given string. The app then uses the first string in the list as the close match.

To get the ten most similar films, obtain the movie's index and using the cosine similarity matrix. The software retrieves the movie's index from the movies data when it has discovered a close match for the film. The ten most comparable movies are then determined using the cosine similarity matrix.

Lastly, the user receives a list of suggested movies. At last, the user receives a list of suggested movies from the app. After that, the viewer can click on any suggested movie to find out more information about it.

5 Experiment Result

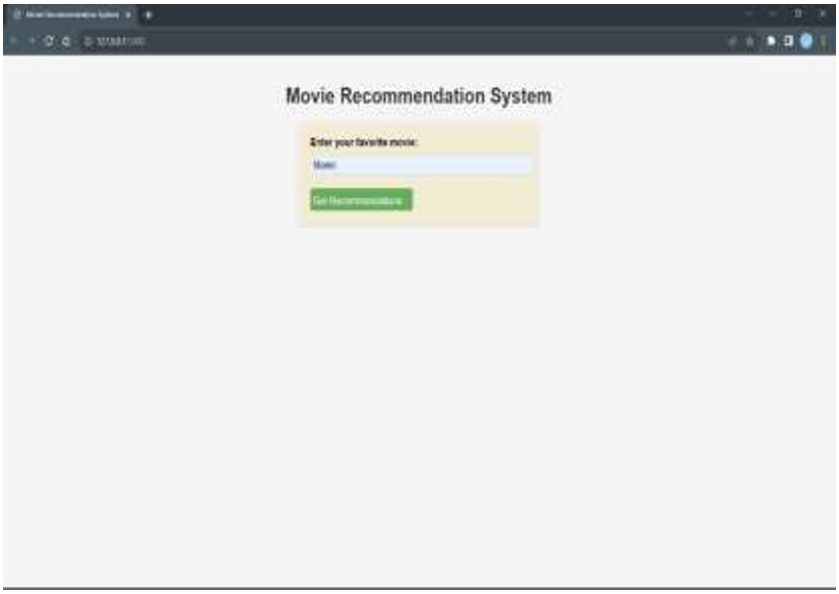

Fig. 4. Input

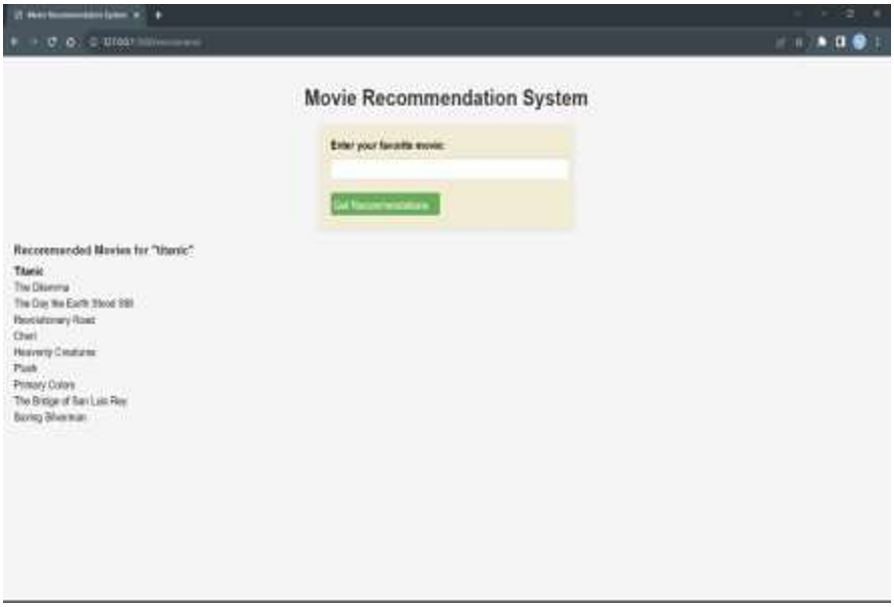

Fig. 5. Output

6 Conclusion

To sum up, the collaborative filtering-based movie recommendation app has shown to be a useful resource for tailored entertainment recommendations. Through the utilization of a wide range of user preferences and viewing behaviors, the collaborative filtering algorithm

is highly effective in providing personalized movie suggestions that correspond with individual preferences. This method's strength is its capacity to spot complex user patterns and similarities, resulting in a responsive and dynamic system. The collaborative filtering model continuously improves its recommendations as users engage with the app, offering ratings and feedback. This allows the model to adjust to changing user preferences and provide a consistently relevant experience.

References

1. Shalabh Aggarwal, Flask Framework Cookbook: Enhance your Flask skills with advanced techniques and build dynamic, responsive web applications , Packt Publishing, 2023.
2. A. V. Dev and A. Mohan, "Recommendation system for big data applications based on set similarity of user preferences", *2016 International Conference on Next Generation Intelligent Systems (ICNGIS)*, pp. 1-6, 2016.
3. N. Kapoor, S. Vishal and K. K. S., "Movie Recommendation System Using NLP Tools," 2020 5th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2020, pp. 883-888, doi: 10.1109/ICCES48766.2020.9137993.
4. Manoj Kumar and DK Yadav, "(MNNIT Lucknow)", *A Movie Recommender System*, 2015.
5. 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.
6. Debnath, S., Ganguly, N., Mitra, P.: Feature weighting in content based recommendation system using social network analysis. In: Proceedings of the 17th International Conference on World Wide Web. ACM (2008)
7. Bhatt, B.: A review paper on machine learning based recommendation system. Int. J. Eng. Dev. Res. (2014)
8. https://realpython.com/build-recommendation-engine-collaborative-filtering/#:~:text=Collaborative%20filtering%20is%20a%20technique,similar%20to%20a%20particular%20user. (last accessed on)
9. https://flask.palletsprojects.com/en/3.0.x/(last accessed on)
10. https://www.geeksforgeeks.org/what-is-reinforcement-learning/(last accessed on)
11. https://developers.google.com/machine-learning/recommendation/collaborative/basics(last accessed on)
12. https://towardsdatascience.com/collaborative-filtering-based-recommendation-systems-exemplified-ecbffe1c20b1(last accessed on)
13. https://www.xenonstack.com/blog/python-flask-framework(last accessed on)
14. https://www.sciencedirect.com/topics/computer-science/cosine-similarity#:~:text=Cosine%20similarity%20measures%20the%20similarity,document%20similarity%20in%20text%20analysis.(last accessed on)
15. https://www.oreilly.com/library/view/statistics-for-machine/9781788295758/eb9cd609-e44a-40a2-9c3a-f16fc4f5289a.xhtml(last accessed on)