Intelligent Movie Recommendations System with Collaborative Filtering and Machine Learning Techniques

S AISHWARYA RAO[1], DR. GOLDI SONI[2]

[1] Undergraduate Student, Amity School of Engineering and Technology, Amity University Chhattisgarh

[2] Assistant Professor, Amity School of Engineering and Technology, Amity University Chhattisgarh

Abstract— ***Watching movies and TV shows has become more convenient with the introduction of streaming services. Over time, the film industry has experienced rapid growth. It's difficult for users to decide what to watch these days because there is so much content available. Systems for recommending movies have been created to help consumers choose movies according to their personal tastes. This streamlines and entertains the selection process. These systems use a number of techniques to provide users with tailored recommendations. One of the most popular techniques is collaborative filtering, which suggests movies to users based on their viewing habits and past selections. An additional technique is content-based filtering, which makes use of movie characteristics such as genre, stars, and directors to recommend other films that share those characteristics. Hybrid methods that combine the two approaches have also been developed to offer recommendations that are more accurate. It highlights how important personalization is to recommendation systems because it increases user satisfaction and engagement.***

Index Terms- Collaborative Filtering, Cosine Similarity, SVD, RMSE, Recommender system

I. INTRODUCTION

Movie recommendation engines are becoming a common sight in the age of internet streaming services. With so many movies to choose from on services like Netflix, Hulu, and Amazon Prime Video, it can be difficult to find one that suits your tastes and mood. This is where movie recommendation systems come into play; they analyze your viewing history using sophisticated algorithms and offer tailored recommendations for new films and TV series that you will probably like.

The core elements of a movie recommendation system consist of multiple filtering methods, such as collaborative, content-based, and demographic filtering. Collaborative filtering looks at the viewing patterns and interests of similar users to make recommendations based on their viewing histories. For example, if you enjoy a movie, collaborative filtering will recommend other movies that people who have similar tastes in movies have also loved. Because there is such a large selection of films available on different platforms, users might have problems finding movies that they like.

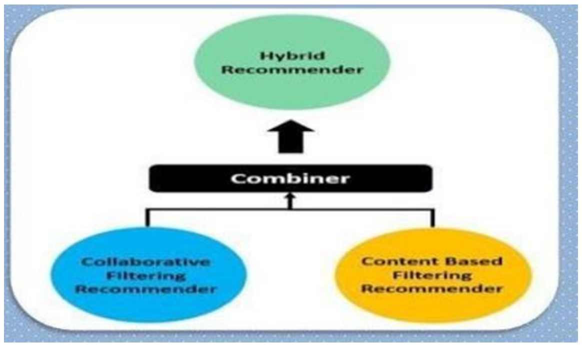

Hybrid Recommender
Combiner
Collaborative Filtering Recommender
Content Based Filtering Recommender

Fig.1. Recommender System

Users may encounter difficulties in discovering films that resonate with them due to the vast array of films that are accessible across various platforms.

Conventional movie recommendation systems might not take into account each user's distinct interests and preferences or offer personalized recommendations. The use of collaborative filtering, which generates movie recommendations based on user behavior and preferences, is a drawback of traditional movie recommendation systems. This approach might not accurately represent the user's distinct preferences and might result in suggestions that are insufficient or irrelevant.

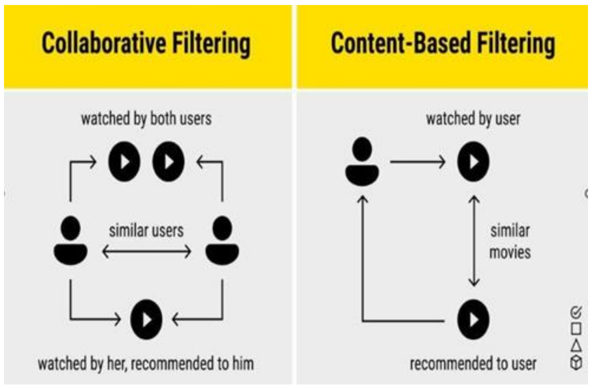


Fig. 2. Comparison between collaborative and content-based filtering

II. LITERATURE REVIEW

1. Utilizing natural language processing (NLP) methods to analyze the textual components of films

For instance, a study used natural language processing (NLP) to analyze movie plot summaries and extract information about the main characters, their actions and emotions, and their locations. These features were used to predict the movie genres, which were then used to train a classifier to generate recommendations.

Another tactic is to use clustering techniques to arrange movies according to their attributes and then recommend movies based on those clusters. (Poria et al. 's, 2014)

2. Content-Based Movie Recommendation System Using Genre Correlation

For instance, a study classified movies based on their genre, actors, and directors using clustering algorithms and then suggested films from the same cluster as the ones the user had chosen. (Bobadilla et al., 2013)

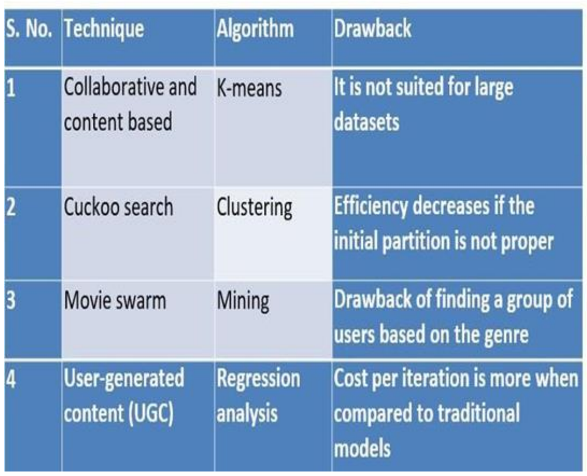
3. Collaborative Filtering vs. Content-Based Filtering: differences and similarities

Recommendation systems (SR) provide suggestions for products that investigate user preferences, assisting users in overcoming information overload. Two methods of social media research have gained increased attention: Content-Based Filtering and Collaborative Filtering. Furthermore, few findings objectively demonstrate their traits, parallels, and distinctions, despite research pointing out their benefits and drawbacks. This work proposes an experimental methodology to compare recommendation algorithms for various approaches that go beyond the "precision of the predictions". (Rafael Glauber, Angelo Loula)

4. A personalized movie recommendation system based on collaborative filtering

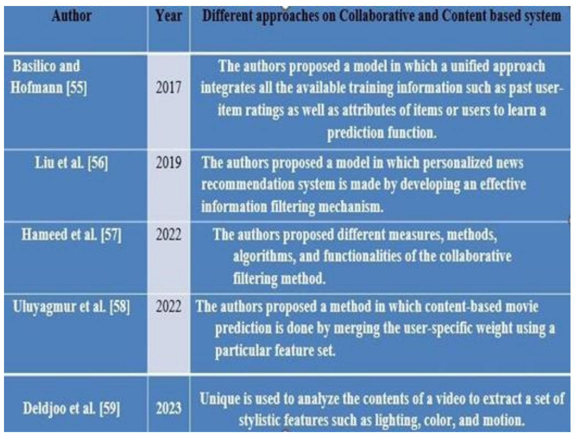
The last ten years have seen an explosion in data because of social media, e-commerce, and the general digitization of businesses. The information is used to forecast market trends, and customer preference patterns, and make well- informed decisions. Since internet services have become widely used, recommendation systems have proliferated. The idea is to suggest items that users might find interesting by using clustering and filtering techniques. Users are given suggestions for media commodities such as movies by locating user profiles of people who share similar interests. Users are first asked to rate the movies they want, in order to determine their preferences. (V. Subramaniyaswamy, R. Loges)

TABLE 1
Comparative study-related works and algorithms



S. No.	Technique	Algorithm	Drawback
1	Collaborative and content based	K-means	It is not suited for large datasets
2	Cuckoo search	Clustering	Efficiency decreases if the initial partition is not proper
3	Movie swarm	Mining	Drawback of finding a group of users based on the genre
4	User-generated content (UGC)	Regression analysis	Cost per iteration is more when compared to traditional models

TABLE 2
Different Machine Learning approaches



Author	Year	Different approaches on Collaborative and Content based system
Basilico and Hofmann [55]	2017	The authors proposed a model in which a unified approach integrates all the available training information such as past user-item ratings as well as attributes of items or users to learn a prediction function.
Liu et al. [56]	2019	The authors proposed a model in which personalized news recommendation system is made by developing an effective information filtering mechanism.
Hameed et al. [57]	2022	The authors proposed different measures, methods, algorithms, and functionalities of the collaborative filtering method.
Uluyagmur et al. [58]	2022	The authors proposed a method in which content-based movie prediction is done by merging the user-specific weight using a particular feature set.
Deldjoo et al. [59]	2023	Unique is used to analyze the contents of a video to extract a set of stylistic features such as lighting, color, and motion.

III. RESEARCH “OBJECTIVES

- Examining a movie's material to find key elements that may be utilized to produce suggestions.
- Creating a list of suggested movies by comparing the movie features to the user's preferences.
- Giving viewers individualized suggestions based on their unique watching interests and histories.
- Offering a satisfying and individualized experience will increase user engagement and retention” rates.

IV. RESEARCH METHODOLOGY

Step 1: Mounting Google Drive and Importing Libraries
To manipulate data and perform numerical computations, we import the necessary libraries, such as pandas and numpy. Additionally, we import warnings to suppress any warning messages during the execution of our code.

Step 2: Data Collection
To build the movie recommender system, the first step is to gather relevant data. We will use movie datasets for this project, which include details about titles, genres, keywords, cast members, and crew. These datasets will serve as the foundation for our recommendation engine.

Loading Datasets: Exploring Dataset Shapes:
Data Merging and Column Selection

Merging Datasets:

Step 3: Data Cleaning
To guarantee data integrity and dependability, we will clean the data in this step by dealing with missing values and getting rid of duplicates.

Checking for Null Values
Handling Null Values
Checking for Duplicate Rows

Step 4: Data Preprocessing
Formatting 'Genres' and 'Keywords' Columns
The 'genres' column contains data in the form of a dictionary. We'll convert these entries into a list and keep only the genre names such as 'Action', 'Adventure', 'Fantasy', etc.

We'll start by converting the 'genres' and 'keywords' columns from strings of lists to actual lists, keeping only the genre and keyword names.

Extracting Top Cast Members Extracting Director's Name Creating New DataFrame

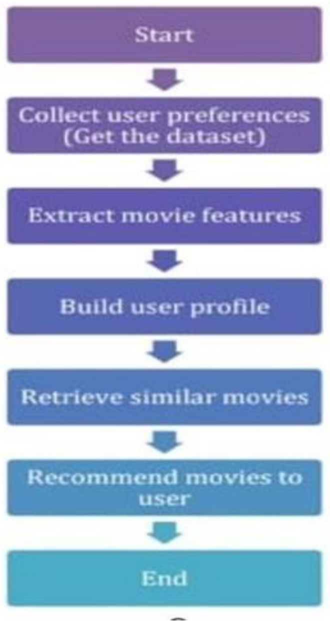


Fig. 3. Data Flow Diagram of recommender system

Step 5: Feature Engineering
Text Vectorization :
- Text vectorization is the process of converting text data into numerical vectors or arrays that can be processed by machine learning algorithms. Text

data is usually unstructured and in a format that is difficult for algorithms to interpret.

- Text can be made into a format that is simple to understand and used for modeling by vectorizing it. There are several machine-learning techniques for text vectorization, including:

1. Bag-Of-Words
2. TF-IDF
3. Word Embeddings

Using CountVectorizer for Text Vectorization

To apply text vectorization using the Bag-of-Words technique, we'll utilize the CountVectorizer class from scikit-learn. This class allows us to convert text data into numerical vectors by creating a vocabulary of words and counting their frequencies in each document.

Let's break down the steps involved in using CountVectorizer for text vectorization:

1. Importing Required Libraries and Initializing CountVectorizer
2. Applying Vectorization and Converting to Array
3. Obtaining Feature Names
4. Displaying Feature Names

Finally, we can loop through the feature names to display the words one by one:

This loop prints each word in the vocabulary, which represents a feature in the numerical vectors.

Applying Stemming to Reduce Redundancy

To address redundancy in our text data and reduce the vocabulary size, we'll apply stemming using the Porter Stemmer from the NLTK library.

Here's how we'll do it:

1. Importing NLTK and Initializing Porter Stemmer

First, we need to install NLTK (Natural Language Toolkit) if it's not already installed. Then, we import NLTK and initialize the Porter Stemmer:

2. Defining the Stemming Function

Next, define a function called stemming that takes a string of text as input, splits it into individual words, utilizes the Porter Stemmer to apply stemming to each word, and then reassembles the stemmed words into a string:

3. Applying Stemming to 'tags' Column

Now, apply the stemming function to the 'tags' column of our DataFrame:

This will transform each tag in the 'tags' column by reducing words to their root forms using stemming. The result is a DataFrame with reduced redundancy and a more compact representation of the text data, which is suitable for text vectorization.

4. Viewing the Transformed 'tags' Column

After applying stemming, we can examine the transformed 'tags' column to see how the words have been reduced to their root forms:

This column now contains the stemmed versions of the original tags.

Calculating Cosine Similarity for Movie Recommendation

To recommend movies based on similarity, we'll calculate the cosine similarity between movie vectors. Here's how we'll do it:

1. Importing Cosine Similarity

We will import the cosine_similarity function from the sklearn.metrics.pairwise module. This function computes the cosine similarity between two vector pairs.

2. Calculating Cosine Similarity

The cosine similarity between each pair of movie vectors in our dataset will be computed.

3. Viewing the Shape of Similarity Matrix

Each element (i, j) in the similarity matrix, which has dimensions (4806, 4806), represents the cosine similarity between movie vectors i and j.

Step 6: Model Building

Creating Recommendation Function

To recommend movies based on similarity to a given movie, we'll create a recommendation function. Here's how it works:

Explanation:

- Input for the recommendation function is a movie title.
- It locates the DataFrame's input movie index.
- It obtains the cosine distances, or similarity values, between the input movie and every other movie.
- The enumerate function associates each similarity value with its index position.
- The sorted function sorts the list of similarity values while maintaining the original index positions.
- The list is sorted in descending order by using the reverse=True argument.
- The argument key=lambda x: x[1] indicates that the similarity value, which is the second element of each tuple, should be the basis for sorting.
- The [1:6] slice returns the top five similar movies, excluding the input movie itself.
- Finally, it prints the titles and details of the recommended movies.

Step 7: Creating the User Interface
Streamlit Library :
Streamlit is a Python library that simplifies the creation of data-focused web applications. Without requiring knowledge of web development languages like HTML, CSS, or JavaScript, it enables developers to create interactive web apps straight from Python scripts. With Streamlit, developers can focus on writing Python code to analyze data and create visualizations, while Streamlit takes care of rendering the user interface and managing user interactions.

Creating UI for a Content-Based Movie Recommendation System using Streamlit

Ensure that you have the necessary packages installed and the pickle files (moviesDict.pkl and similarity.pkl) available in your directory. Also, replace the API key 'Your API Key' with your own if needed.

Step 8: Deploying the Application

1. Prepare Application Files: Ensure all necessary files, including app.py containing Streamlit code, and data files (`moviesDict.pkl`, similarity.pkl), are up-to-date and ready for deployment.
2. Dependencies: Document dependencies in a requirements.txt file listing all required Python packages and their versions. This includes Streamlit and any other libraries used in the application.
3. Hosting Platform: Streamlit Sharing serves as the hosting platform, facilitating easy deployment of Streamlit applications without additional setup.
4. Create Account (if necessary): If required, sign up for a Streamlit Sharing account to proceed with the deployment process.
5. Deployment Method: Connect the GitHub repository containing the application code to the Streamlit Sharing account. This enables a seamless deployment process directly from GitHub.
6. Environment Setup: Streamlit Sharing automatically configures the deployment environment based on the dependencies specified in the requirements.txt file, eliminating the need for manual setup.
7. Deploy Application: Initiate the deployment process either by triggering a deployment from the Streamlit Sharing dashboard or with a single click, leveraging the connected GitHub repository.
8. Testing: After deployment, thoroughly test the application on Streamlit Sharing to ensure proper functionality. Explore various features, user interactions, and edge cases to identify any errors or bugs.
9. Monitoring and Maintenance: Regularly monitor the deployed application on Streamlit Sharing for performance, reliability, and security. Update dependencies as needed and address any reported issues promptly to maintain a smooth user experience.
10. Scale (if necessary): Streamlit Sharing automatically manages scaling based on application demand. As the application gains users and traffic, Streamlit adjusts resources to ensure optimal performance without manual intervention.

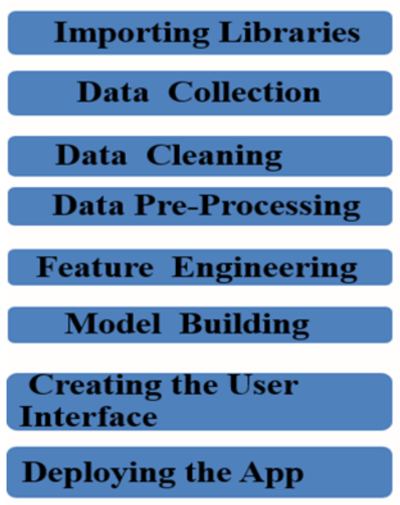


Fig. 4. Steps of recommendation system

V. FINDINGS

Collaborative filtering typically operates in two "ways:

1. User-based collaborative filtering: Using this method, the computer finds people who have similar tastes in movies to the target user and suggests movies that the target user might like. The algorithm might suggest an action film that User B has already seen and highly rated, for instance if User A and User B enjoy watching action movies together and User A is looking for a new movie to watch".
2. Item-based collaborative filtering: With this method, the system finds films that the intended user has already seen and recommends those that are comparable. For instance, the system may suggest other romantic comedies with comparable themes, plots, and characters to a user who has enjoyed watching romantic comedies in the past.

One advantage of collaborative filtering over demographic data is that it can offer highly personalized recommendations to customers based on their preferences. One limitation of collaborative filtering is that its effectiveness depends heavily on user data. Furthermore, it might not work well for new or niche films with little to no user feedback. Lastly, a useful technique for movie recommendation systems that may provide users with personalized recommendations based on their preferences and actions is collaborative filtering. It should be combined with other tactics, like demographic and content-based filtering, to provide more detailed and accurate recommendations.

As opposed to other distance metrics, why choose Cosine Similarity?
A content-based movie recommendation system looks for patterns in user-submitted movie recommendations, comparing the features of the user's favorite films with those of other films in the system. Using distance metrics, like the Manhattan distance or the Euclidean distance, is a popular way to gauge similarity. Nonetheless, Cosine Similarity is frequently chosen above these techniques for a number of reasons:

1. Resilience to Magnitude Differences
2. Dimensionality Reduction
3. Cosine Similarity is computationally efficient

VI. FUTURE SCOPE

The future of human-computer interaction and user experience design is poised for significant advancements, driven by cutting-edge technologies and innovative methodologies. As we progress, three critical areas promise to enhance the depth and quality of user engagement:

1. Multimodal Sentiment Analysis: Integrating various data sources such as text, audio, and visual inputs will enable more accurate sentiment analysis. By capturing and interpreting emotions across multiple modalities, systems can respond more empathetically and effectively to user needs.
2. Context-Aware Recommendations: Leveraging contextual data such as location, time, user activity, and historical behavior, recommendation systems will become more personalized and relevant. These context-aware recommendations will significantly improve user satisfaction and engagement by providing timely and pertinent suggestions.
3. Integration with Virtual Reality (VR) and Augmented Reality (AR): The fusion of VR and AR with existing technologies will create immersive and interactive user experiences. This integration will not only enhance entertainment and gaming but also revolutionize fields like

education, healthcare, and remote work by providing realistic and context-rich environments.

CONCLUSION

In the digital age, the entertainment sector is using recommendation algorithms more and more to improve customer engagement and " experience. Content-based movie recommendation systems are among the most widely used in the film industry. This system examines movie content to provide customers with personalized recommendations.

With the use of complex algorithms, content-based movie recommendation systems examine a variety of cinematic components, such as genre, director, actors, and" story. By identifying patterns in the similarities and differences between movies, the algorithm might offer users personalized recommendations based on their preferences and interests. By making it possible for people to locate new movies, it allows viewers to learn about new genres and directors by suggesting movies that are similar to ones they have already seen.

Nevertheless, content-based movie recommendation systems also have certain disadvantages. They cannot account for external factors that could influence user choices, such as mood or social influence. Moreover, the recommendation system can only make suggestions for films that are comparable to those that users have already found enjoyable; it cannot make suggestions for films that are unrelated to the user's known preferences.

Despite these drawbacks, companies and moviegoers can benefit from content-based movie recommendation systems. Businesses that provide users with personalized suggestions can increase user engagement and profitability. Content-based movie recommendation algorithms can lead movie buffs to new films they might not have looked into otherwise, making for a more interesting and varied viewing experience.

In conclusion, content-based movie recommendation systems offer several advantages such as a greater variety of viewing options, ease of use, and personalized suggestions. Despite a few drawbacks, these systems are a vital resource for companies as well as movie buffs. As technology advances, successful content-based movie recommendation systems will become possible.

REFERENCES

[1] Choi, Sang-Min, Sang-Ki Ko, and Yo-Sub Han. "A movie recommendation algorithm based on genre correlations." Expert Systems with Applications 39.9 (2012): 8079-8085.

[2] S. S. K. Kumar, R. Srinivasan and Bobadilla et al. "Recommendation system using content-based filtering. In 2013 International Conference on Computer Communication and Informatics (ICCCI), pages 1–5, 2013".

[3] Panniello et al., X. Liu, Y. Xu, and X. Li. A hybrid recommendation algorithm based on content-based filtering and collaborative filtering. In 2014 IEEE 8th International Conference on Communication Software and Networks (ICCSN), pages 123–127, 2014.

[4] Sedhain et al., G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6):734–749, 2015

[5] R. M. El-Khoury, M. Khalil, and M. Shouman. Survey of content-based recommender systems. ACM Computing Surveys, 49(3):43:1–43:34, 2016.

[6] J. Lops, M. De Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. Recommender Systems Handbook, pages 73–105. Springer, 2015.