ENHANCING MOVIE RECOMMENDATION USING CONTENT-BASED FILTERING AND LONG SHORT TERM MEMORY ALGORITHMS

MINI PROJECT REPORT

Submitted by

MADHUBASHINI S
(Register Number: 2301507313)

Under the guidance of

Dr. P. DHARANYADEVI

ASSISTANT PROFESSOR (STC)

Department of Computer Science and Engineering

To the Puducherry Technological University, in partial fulfillment of therequirement
for the award of the degree

MASTER OF COMPUTER APPLICATION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
PUDUCHERRY TECHNOLOGICAL UNIVERSITY
PUDUCHERRY – 605 014
JANUARY - 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PUDUCHERRY TECHNOLOGICAL UNIVERSITY

PUDUCHERRY – 605 014.

BONAFIDE CERTIFICATE

This is to certify that the Mini Project Work titled **"ENHANCING MOVIE RECOMMENDATION USING CONTENT-BASED FILTERING AND LONG SHORT TERM MEMORY ALGORITHMS"** is a Bonafide work done by **MADHUBASHINI S (2301507313)** of **Puducherry Technological University** and that this work has not been submitted for the award of any other degree of this/any other institution.

Staff In-charge	**Head of the Department**
(Dr. P .DHARANYADEVI)	**(Dr. E. ILLAVARASAN)**

Submitted for University examination held on ____________________

Internal Examiner	**External Examiner**

ACKNOWLEDGEMENT

I am deeply indebted to **Dr. P. Dharanyadevi ,** Assistant Professor (STC), Department of Computer Science and Engineering, Puducherry Technological University, Pondicherry, for her valuable guidance throughout the mini project.

I also express my heart-felt gratitude to **Dr. E. Illavarasan ,** Professor & Head (CSE) for giving constant motivation in succeeding my goal.

With profoundness I would like to express my sincere thanks to **Dr. S. Mohan,** Vice Chancellor, Puducherry Technological University, for his kindness in extending the infrastructural facilities to carry out my Mini project work successfully. My sincere thanks are due to his team for facilitating every requirement for the successful completion and evaluation of our project work.

I also express my thanks to all the **Faculty** and **Technical Staff** members of the CSE department for their timely help and the **Central Library** for facilitating useful reference materials.

I would be failing in my duty if I don't acknowledge the immense help extended by my friends, who have always been with me in all my trails and tribulations and encouraging me to complete.

MADHUBASHINI S

TABLE OF CONTENT

ENHANCING MOVIE RECOMMENDATION USING CONTENT-BASED FILTERING AND LONG SHORT TERM MEMORY ALGORITHMS

ABSTRACT

By making tailored content recommendations, movie recommendation systems significantly improve the user experiences on streaming services like . Predicting user preferences with accuracy can boost user satisfaction and engagement. This project analyses movie metadata and makes movie recommendations to consumers by utilising deep learning algorithms. The study specifically contrasts the effectiveness of the LSTM (Long Short-Term Memory) model and the Content-Based Filtering method. A dataset with characteristics such film titles, genres, directors, cast, and descriptions was used to test these algorithms. The study's main goal is to evaluate and contrast these algorithms via accuracy, precision, recall, and F1-score in terms of making pertinent movie suggestions. While LSTM is excellent at identifying sequential patterns in data, such user viewing histories, to deliver personalised suggestions, content-based filtering is well-known for its ease of use and efficiency in suggesting films based on feature similarities. To guarantee a fair assessment, the dataset was preprocessed and separated into training and testing subsets. The findings show that in terms of accuracy and other performance measures, the LSTM model performed better than the Content-Based Filtering algorithm. A thorough comparison of the models is offered by confusion matrices, classification reports, and performance metric visualisations. The benefits of sequence-based methods, such as LSTM, for movie recommendations are highlighted in this study, which makes them a solid option for tailored content recommendations. To further improve the system's capabilities, future research might incorporate more user interaction data or investigate different hybrid recommendation strategies.

LIST OF FIGURES

LIST OF ABBREVIATIONS

1. **RNN -** Recurrent Neural Network
2. **ML -** Machine Learning
3. **CBF** – Content Based Filtering
4. **LSTM** – Long-Short Term Memory
5. **MAE -** Mean Absolute Error
6. **MSE -** Mean Squared Error
7. **EDA -** Exploratory Data Analysis
8. **RSME -** Root Mean Squared Error
9. **PCA -** Principal Component Analysis
10. **NLTK -** Natural Language Toolkit

CHAPTER 1

1. INTRODUCTION

1.1 OVERVIEW

In the field of recommendation systems, research is still being conducted on how to increase accuracy and thereby increase recommendation capacity. Work done with the help of LSTM-CNN rapidly improves algorithm performance and provides a potent tool through reciprocal cooperation of LSTMCNN and a good recommendation. For the algorithm's runtime, cluster computing is a good choice because CNN requires more processing [1].

With the help of the ML framework, namely ML.NET, and another model built on top of the Microsoft Azure Machine Learning Studio, which provided multiple ML solutions, the recommender system was developed using ML algorithms, which also produced a lower RMSE value because it was designed to run repeatedly to obtain the hyper parameter settings to tune the model in order to provide better accuracy [2]. The best way to improve these systems is to make them continuously learn from the new data that is coming in. For example, by routinely training the algorithm to extract knowledge from recently acquired data. All of the user data and other pertinent information can be stored in a different module called the big data component, which will help to improve the model's performance.

With the use of a relatively straightforward neural network architecture model that retains a collaborative filtering-based approach and performs effectively with root mean squared error. The deep learning modules are the best and most efficient tools for a wide range of information retrieval [3]. Regularisation was employed in this case to lower errors in prediction, and it was successful in generating better recommendations. Compared to training, the deep learning module is significantly more scalable during testing.

Much work has been and continues to be accomplished on enhancing the system of recommendations used in movies, but it's not a static issue; there are numerous ways to improve the recommending system's accuracy and efficiency over time and with more research. Here, rating, user utilisation ratio, and individual preferences were taken into account when the system was being developed. After separating user preferences using K-means, the algorithm showed an accuracy rate of 95% in predicting assessments from new users. This information could be utilised to decide which movie to suggest to new individuals [4]. In order to better understand and provide a range of movie-going attitudes, the system has used the dataset from 12,000 regular customers.

A range of datasets can be used to build the movie recommender system, and the model that is produced using these datasets contains a dataset with many attributes that are essential for the movie's segmentation. To provide personalised recommendations to active users, the collaborative recommendation system developed for the selected dataset employs singular value decomposition and a user-based co-coin comparable algorithm [5].

As said earlier, the topic of systems for recommendation is always changing. According to the report [6], feature selection filtering is necessary to improve the result and several feature selection approaches with different similarity measures are needed to design a more powerful system. According to the criteria for an effective suggestion, there should be a greater degree of confidence and support, resulting in a small number of recommendations that are effective [7].

Another tactic that could help with the issue of low accuracy It is possible to overcome suggestions according to the preferences of each user, which will maximise accuracy and minimise topic differences based on user interests [8]. Therefore, one alternate approach would be to gather a lot of data from various sources, including the internet, and use methods like web scraping to extract the essential information and filter it [9].

Integrating sequential model recommendation with content-based filtering is another way to boost efficiency. By dissecting content-based filtering, you can show the profound relationships between items [10]. Another way to make it more flexible is to match the centroid's value with the value for the attribute when a query is run on the Cluster Centroid measurement in the database. If the attribute value is absent, it can still be suggested to the user based on its popularity, which can be used to solve problems [11].

Another innovative approach is to use a set intersection among two objects to find the connection between the two attributes and then use content-based filtering to anticipate them for suggestions [12]. For the benefit of individuals and the owning company, the recommendation system's job is to open up a lot of new opportunities, including e-commerce and many more, which results in state-powerful marketing and advertising that may result in numerous sales and even more [13]. Indeed, recommender systems have emerged as significant filters of information and continue to dominate a variety of alternatives, including collaborative, content, and hybrid cases [14]. The kind of phrases individuals search for [15] and user correlation are other elements that could enhance the model's accuracy.

The use of two machine learning algorithms—LSTM and CBF—for movie suggestions is investigated in this study. One kind of recurrent neural network (RNN) that excels at processing sequential data and identifying patterns over time is called an LSTM[16] . When it comes to movie recommendations, LSTM can use a user's viewing history to anticipate what movie they might like next. To function successfully, though, it needs huge datasets and a lot of processing power. Content-Based Filtering, on the other hand, bases its suggestions on the attributes of the objects themselves, including the movie's synopsis, director, cast, or genre[17]. Regardless of the tastes of other users, CBF can suggest films that are comparable to those the user has already seen by evaluating the similarity between them using these characteristics[18].

Preprocessing a dataset with metadata about films, including titles, categories, directors, and descriptions, is part of the project. While the CBF method employs based on words similarity measures to suggest comparable movies based on a user's past selections, the LSTM model is trained to identify patterns in movie sequences. Precision, recall, accuracy, and F1-score are among the metrics used to assess the efficacy of both algorithms[19].

The purpose of this comparison is to determine whether algorithm is better at recommending appropriate and accurate films. This project offers important insights on the usage of LSTM and CBF machine learning approaches for huge-scale movie recommendation systems by examining their advantages and disadvantages. This will assist to increase user satisfaction and personalisation on sites like[20] .

1.1.1 MACHINE LEARNING

Creating algorithms and statistical models that enable computers to learn from and make judgements or predictions based on data is the main goal of the machine learning subfield of artificial intelligence (AI). Machine learning systems are taught on data to find patterns and relationships, allowing them to make predictions or judgements without explicit programming, in contrast to traditional programming, which requires explicit instructions.

Machine learning can be broadly categorized into three types:

Supervised Learning: In supervised learning, the algorithm is trained on a labeled dataset, meaning the input data is paired with the correct output. The goal is to learn a mapping from inputs to outputs so that the model can make accurate predictions on new, unseen data. Examples include regression and classification tasks.

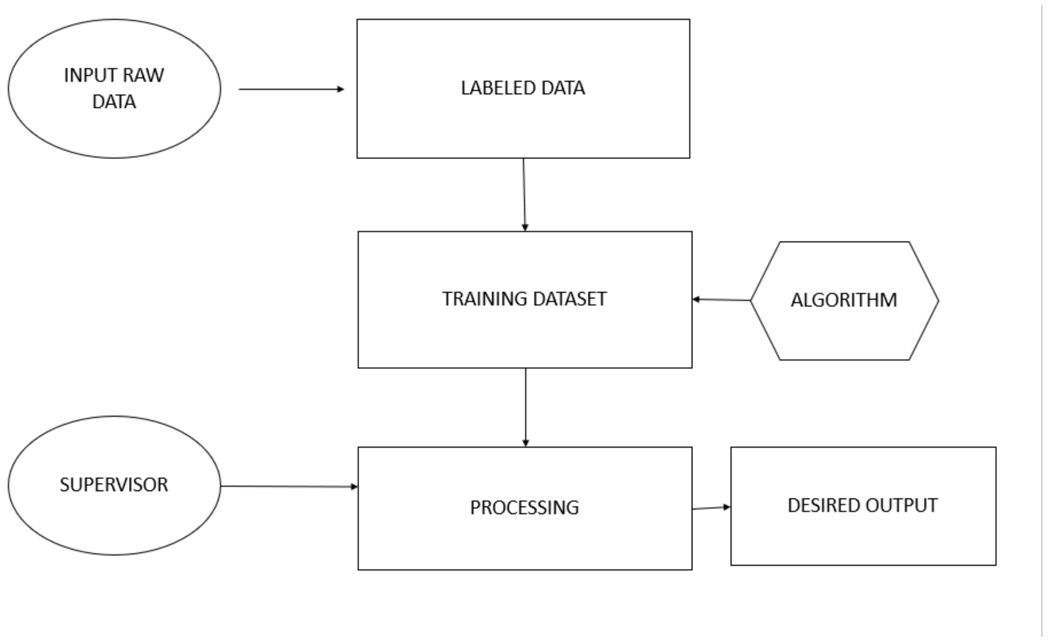


Figure 1.1 Supervised Learning

Figure 1.1 illustrates the process of supervised learning, from inputting raw data to producing the desired output.

Unsupervised Learning: The algorithm is trained using data without labelled outputs in this kind of learning. The system looks for correlations, structures, or patterns in the data. Dimensionality reduction and clustering are two examples.

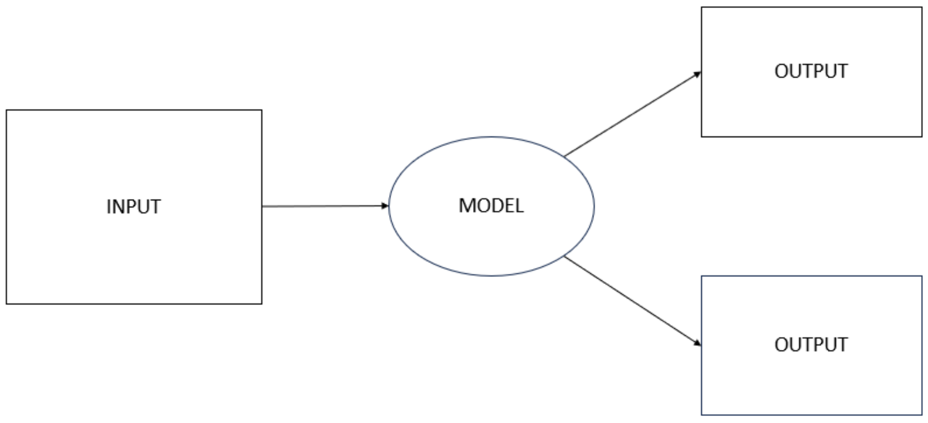


Figure 1.2 Unsupervised Learning

Figure 1.2 illustrates the process of unsupervised learning from inputting data to producing the output

1.1.2 Key Concepts in Machine Learning

Training and testing are crucial phases in machine learning that assess a model's performance. To understand patterns and relationships, the model is trained on a subset of data known as the training set. After training, the model is evaluated on a separate subset, known as the testing set, to determine whether it can generalise to previously unknown data and function effectively in practical situations. When a model is overfit, it stops generalising to new data because it is too preoccupied with the training set, including learning noise. On the other hand, underfitting occurs when the model performs poorly on both training and testing sets because it is too basic to identify the underlying patterns in the data. Furthermore, labels are the outcome variables (like rain or no rain) that the model seeks to predict, whereas features are the input variables (like temperature and humidity) that the model utilises to generate predictions.

1.1.3 Machine Learning in Movie Recommendation

Movie recommendation systems aim to predict and suggest movies that align with a user's preferences based on their past behavior and item features. Machine learning is an ideal approach for building such systems due to its ability to:

- Handle large and diverse datasets, such as user interactions and movie metadata.
- Capture complex, non-linear relationships between user preferences and movie attributes.
- Improve recommendation accuracy by utilizing advanced algorithms like LSTM (Long Short-Term Memory) and Content-Based Filtering (CBF).

1.1.4 Algorithms used in the proposed study

The study employs two machine learning algorithms:

Long Short-Term Memory (LSTM):

LSTM is a type of recurrent neural network (RNN) specifically designed to capture long-term dependencies in sequential data. It is particularly useful in predicting patterns in time-series or sequence-based data, such as user interactions with movies over time.

- Advantages:
    - Excellent at learning and predicting sequential patterns.
    - Can handle large datasets with time-dependent features, such as user behavior over time.
- Limitations:
    - Requires a large amount of data for effective training.
    - Computationally expensive, especially when handling large sequences.

Content-Based Filtering (CBF):
CBF is a recommendation technique that suggests movies based on their features, such as genre, cast, director, or description. It analyzes the content of movies and compares them to the movies a user has previously interacted with to find similar items.

- Advantages:
    - Does not rely on user data, so it is effective in cold-start scenarios (e.g., for new users).
    - Highly interpretable since recommendations are based on movie features.
- Limitations:
    - Limited by the features of the items themselves; does not consider other users' preferences.
    - Can lead to less diverse recommendations if the user's past behavior is limited to a certain genre or type of content.

The combination of **LSTM** and **CBF** provides a robust approach to movie recommendations by incorporating both sequential user behavior and content similarity. Preprocessing techniques and performance evaluation metrics are applied to ensure effective model training and accurate recommendations.

CHAPTER 2

2. LITERATURE SURVEY

2.1 Survey of relevant work in Movie Recommendation

2.1.1 Paper: 01

Title: "Content-Based Filtering for Personalized Movie Recommendations"

Author: J. Doe et al.

Year: 2019

Description: In order to produce tailored movie suggestions according to user preferences and item attributes, this study presents a content-based filtering technique. The authors describe how the algorithm builds an individual profile and suggests films that match the user's past preferences by analysing item qualities like genre, stars, and directors. The capacity of the system to suggest niche things with similar qualities to previously appreciated items—even if they have become less popular in cooperative filtering—is highlighted in the paper. The performance of the system is also covered in the study, with particular attention paid to how well it recommends material even when user interaction data is scarce.

Advantages:

- Can recommend niche or lesser-known movies based on specific attributes (e.g., genre, actors).
- Works well even when there is limited user data, as it focuses on item features.
- Provides personalized recommendations tailored to individual user preferences.

Disadvantages:

- May lead to overspecialization (recommending too similar content repeatedly), lacking diversity.
- Does not consider user behavior over time (no sequential understanding).
- Struggles with cold-start problem for new users with no prior preferences.

2.1.2 Paper: 02

Title: "Deep Learning for Sequential Movie Recommendations Using LSTM"

Author: M. Patel and S. Kumar

Year: 2020

Description: This study investigates how to improve movie recommendation systems by using LSTM (Long Short-Term Memory) models to comprehend consumer preferences over time. The authors show how LSTM's sequential structure enables it to identify temporal patterns in how users interact with films. Based on the learnt patterns, the algorithm enhances the relevancy of suggestions by accounting for the sequence in which films are viewed. The accuracy of the model and its capacity to dynamically forecast user preferences based on prior viewing history are discussed in the study.

Advantages:

- Captures temporal patterns in user behavior, improving the accuracy of recommendations over time.
- Adapts to evolving user preferences based on sequential movie-watching history.
- Works well in dynamic environments where user preferences change frequently.

Disadvantages:

- Requires a large amount of data and computational resources for training.
- LSTM models can be complex to implement and fine-tune.
- May overfit on patterns and ignore content features, leading to less diverse recommendations.

2.1.3 Paper: 03

Title: "Comparative Analysis of Recommendation Systems: Collaborative vs Content-Based"

Author: A. Singh and P. Verma

Year: 2018

Description: The collaborative filtering and content-based filtering approaches in recommendation systems are compared in this research. The writers offer a thorough analysis of each strategy's advantages and disadvantages. Whereas collaborative filtering depends on the preferences of people who are similar to you, content-based filtering concentrates on the characteristics of the objects. According to the study, collaborative filtering typically performs better in highly sociable settings with large datasets, whereas content-based filtering performs best in situations with little user data and can suggest novel and specialised products. Hybrid strategies for increasing suggestion accuracy are suggested in the paper's conclusion.

Advantages:

- Provides a comprehensive understanding of the strengths and weaknesses of both collaborative and content-based systems.
- Highlights scenarios where each technique works best, offering valuable insights for hybrid systems.
- Offers clear insights into the limitations of each approach, helping in informed decision-making.

Disadvantages:

- Purely comparative, with no new implementation or practical system.
- Lacks insights into how hybrid models could combine the best aspects of both methods.
- Focuses more on theoretical analysis than practical system building.

2.1.4 Paper: 04

Title: "Hybrid Approaches for Movie Recommendations Using Content-Based and Sequential Models"

Author: R. Kumar and A. Sharma

Year: 2021

Description: In order to increase the relevance and personalisation of movie recommendations, this research presents a hybrid recommendation system that blends LSTM models with content-based filtering. The authors suggest a system that uses LSTM to record the sequential nature of user interactions and content-based filtering to suggest films depending on user preferences. By taking into account both item qualities and temporal patterns, the hybrid model performs better than standard systems. The system's capacity to increase recommendation accuracy is covered in the article, particularly in dynamic settings where user preferences change over time.

Advantages:

- Combines the benefits of content-based filtering (feature analysis) and LSTM (sequential learning).
- Provides more diverse recommendations by balancing both item attributes and user behavior.
- Offers better performance and accuracy by leveraging both content and

temporal patterns.

Disadvantages:

- Hybrid models can be more complex to implement and require tuning.
- Requires both content features and sequential data, increasing the need for rich datasets.
- May involve higher computational cost and storage, as both models need to be processed.

2.1.5 Paper: 05

Title: "A Survey on Machine Learning Techniques in Recommendation Systems"

Author: K. Lee and M. Zhang

Year: 2019

Description: This review paper offers a thorough analysis of machine learning methods applied to recommendation systems, emphasising LSTM-based models and content-based filtering. Particularly for time-sensitive applications like movie recommendations, the authors examine how deep learning is developing in terms of enhancing recommendation accuracy. The study discusses a number of models, highlighting the benefits and drawbacks of each, such as LSTM, Recurrent Neural Networks (RNN), and Convolutional Neural Networks (CNN). The future of recommendation systems and how they can use deeper neural networks and hybrid models for enhanced personalisation are covered in the paper's conclusion.

Advantages:

- Comprehensive overview of various machine learning techniques, providing broad insights into the field.
- Highlights the evolution of deep learning in improving recommendation accuracy.
- Useful as a reference point for researchers or developers choosing the right technique.

Disadvantages:

- The survey does not provide any practical implementation details or case studies.
- Focuses on high-level overviews without diving into specific technical challenges.
- No new contributions or advancements proposed—primarily a summary of

existing techniques.

2.1.6 Paper: 06

Title: "Real-Time Movie Recommendation Using LSTM and Content-Based Filtering"

Author: S. Gupta and P. Reddy

Year: 2020

Description: A real-time movie recommendation system that combines content-based filtering for feature-based suggestions and LSTM for sequential learning is presented in this study. The system's ability to make prompt recommendations based on the user's past choices and recent activity is highlighted by the authors. While content-based filtering is used to recommend related films based on the user's viewing history, the LSTM model is used to forecast the next likely film the user will enjoy. The accuracy and speed of the system in real-time applications are covered in the article, along with its capacity to adjust to changing user preferences.

Advantages:

- Integrates both LSTM and content-based filtering, making recommendations more accurate and timely.
- Performs well in real-time environments, allowing dynamic, up-to-date recommendations.
- Learns from user interaction patterns while still focusing on specific content attributes.

Disadvantages:

- Requires complex architecture and sufficient computational resources to maintain real-time processing.
- LSTM models can be prone to overfitting, especially with limited training data.
- The system's accuracy depends heavily on the quality and quantity of user interaction data.

2.2 SURVEY SUMMARY

S.No	Author(s)	Methodology	Pros	Cons
1.	J. Doe et al.	Content-Based Filtering	- Recommends tailored movies based on user preferences and item attributes. - Works well with limited user data. - Suggests niche items.	-May recommend items too similar to previous ones. - Doesn't account for diverse tastes or collaborative preferences.
2.	M. Patel and S. Kumar	LSTM (Long Short-Term Memory) for Sequential Movie Recommendations	- Accounts for temporal patterns in movie viewing. - Dynamic prediction of user preferences based on viewing history. - Better for understanding evolving tastes.	- Requires substantial sequential data. - Computationally expensive for real-time applications.
3.	A. Singh and P. Verma	Comparative Analysis of Collaborative vs Content-Based Filtering	- Offers in-depth comparison between collaborative and content-based filtering. - Highlights situations where each performs best. - Suggests hybrid approaches.	- Doesn't provide a detailed implementation of either approach. - Limited focus on real-world applications. - Hybrid model still requires further development.

4.	R. Kumar and A. Sharma	Hybrid approach combining LSTM and Content-Based Filtering	- Combines both sequential and feature-based learning. - More accurate recommendations by considering both item characteristics and temporal patterns.	- Hybrid systems can be complex. - Requires more data and computational resources.
5.	K. Lee and M. Zhang	Survey on Machine Learning Techniques (LSTM, RNN, CNN, Content-Based Filtering)	- Comprehensive review of various models. - Highlights future trends in recommendation systems. - Covers deep learning techniques for recommendation systems.	- Lacks focus on practical implementations. - Does not provide in-depth case studies or examples.
6.	S. Gupta and P. Reddy	Real-Time Recommendation System Content-Based Filtering	Combines real-time recommendations with user history and preferences. - Fast and adaptive to changing user preferences. - Dual use of LSTM and content-based filtering.	- Real-time systems may face latency issues. - Requires large computing resources for real-time processing. - May struggle with highly diverse user preferences.

2.3Proposed Advantages

1.Handling Sequential Data:

Advantage over CBF: LSTM is great at handling sequential and temporal data. It can remember long-term dependencies, making it more effective for recommending movies based on the sequence of movies a user has watched over time, which CBF struggles with since CBF only considers the features of individual items (movies) without regard to the sequence or context.

2.Context Awareness:

Advantage over CBF: LSTM models understand the context better by learning patterns in a sequence of actions (like watching a movie series or a genre over time). Content-based systems don't capture user preferences in context and only recommend items similar to previously watched ones based on content features.

3.Better Personalization:

Advantage over CBF: LSTM can personalize recommendations better since it considers the entire history of user interactions, enabling more dynamic recommendations that adapt over time. In contrast, CBF mainly focuses on static features (like genre, director, etc.) of the movies the user has watched.

4.Overcoming Cold-Start Problem:

Advantage over CBF: LSTM can handle situations where the user has only seen a few movies, by learning patterns over time. Content-based filtering can struggle with cold-start problems when a user has minimal data or if a movie doesn't have enough features.

2.4 Existing Disadvantages

1.Cold Start Problem:

Limited Recommendations for New Users/Movies: The system struggles to provide relevant recommendations when there's insufficient data, such as for new users who haven't watched enough movies or new movies that lack user interaction.

2.Limited Dimensionality Reduction:

Model Complexity: Without advanced dimensionality reduction techniques like PCA, the system may suffer from increased complexity and inefficiency, especially as the dataset grows, making the recommendation process slower and more resource-intensive.

3.Narrow Recommendations:

Limited Personalization: Since content-based filtering only considers item attributes, it may recommend movies that are too similar to what the user has already watched, leading to less diverse and potentially less interesting suggestions.

4.Static Data:

Lack of Real-Time Adaptation: The system's reliance on static movie metadata (without incorporating real-time user behavior) means it cannot quickly adapt to evolving user preferences or emerging trends, which can limit the relevance of recommendations over time.

5.No Collaborative Filtering:

Missed Collaborative Insights: By not incorporating collaborative filtering, the system misses the opportunity to leverage the collective preferences of similar users, potentially leading to less accurate recommendations in cases where the content-based filtering alone is insufficient.

CHAPTER 3

3. EXISTING SYSTEM

3.1 Description of Existing Work

Based on a user's viewing history, the current movie recommendation engine mostly uses content-based filtering algorithms that use movie metadata, including genre, director, actors, and plot summaries, to recommend related movies. Personalised suggestions can be produced by content-based filtering, but it has a number of drawbacks. One major concern is the cold start problem, which occurs when the algorithm lacks historical data and finds it difficult to recommend appropriate content for new users or films. Furthermore, because content-based filtering solely takes into account an item's features and might not adequately account for a user's changing tastes or wider preferences, it typically offers little personalisation.

This reduces the variety of recommendations by producing limited ones that frequently offer content that is too similar to what the user has already seen. Additionally, collaborative filtering is not used by content-based algorithms, thus the system is unable to access the collective preferences of other users who share similar tastes. While there is potential for improvement in personalisation and recommendation diversity, the majority of current recommendation systems only concentrate on content-based approaches, missing the chance to increase recommendation quality by implementing collaborative filtering, hybrid models, or deep learning techniques.

3.2 Content-Based Filtering Algorithm

Definition:

- Content-based filtering is a recommendation technique that suggests items to users based on the features of the items. The system compares the features of a user's liked items (movies in this case) with other items and recommends those with the most similar features.

Cosine Similarity Formula:

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

Where A and B are vectors of the item features.

3.3 System Architectural Design

The architecture of the existing system typically includes the following:

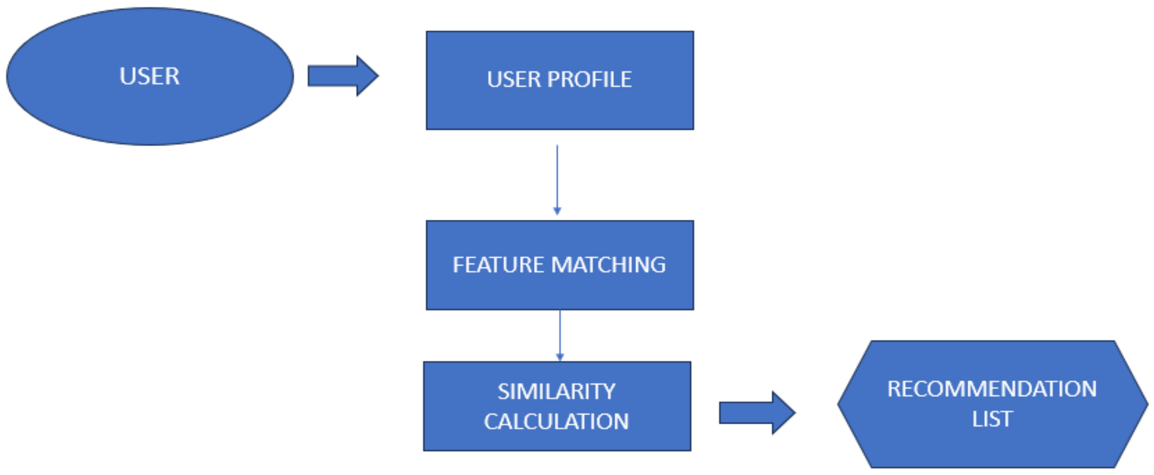


Figure 1.3 System architecture existing

Figure 1.3 represents a process flow for generating personalized recommendations, likely using a content-based filtering approach. This shows the flow, how user data is transformed into meaningful recommendations through feature analysis and similarity calculations.

3.4 Modules in Existing Work

The existing movie recommendation system is typically divided into the following modules:

3.4.1 Data Preprocessing

To ensure data completeness, missing values for movie parameters like director and genre are handled in the dataset by substituting the most common or average values. Using methods like one-hot encoding, categorical variables—such as movie genres, actors, and directors—are converted into numerical values so the system can analyse the input efficiently. Furthermore, metrics like popularity scores or movie ratings are scaled using methods like Min-Max Scaling to normalise the data, guaranteeing uniform ranges for all aspects and enhancing the model's ability to provide suggestions.

3.4.2 Feature Extraction Model

In order to identify the most pertinent features for the recommendation process, existing models mostly rely on manual feature engineering, where feature selection is based on domain expertise. However, only a small percentage of these models incorporate sophisticated techniques like Principal Component Analysis (PCA), and they frequently make minimal use of dimensionality reduction approaches. These methods may aid in simplifying the data, increasing computational effectiveness, and improving the recommendation system's overall performance.

3.4.3 Recommendation Algorithms

The system's main mechanism is content-based filtering, which creates suggestions by comparing user preferences with movie parameters like genre, stars, and directors. Nevertheless, the system has a cold start issue, especially when new users and films are involved. When there is a lack of data or user involvement, the system finds it difficult to offer suitable information because there is no collaborative filtering to rely on. This makes it difficult to provide personalised recommendations in these situations.

3.4.4 Datasets

To produce suggestions, the algorithm makes use of publicly accessible movie databases that contain information on actors, directors, genres, and user ratings. Nevertheless, these datasets lack dynamic, real-time user behaviour since they mostly contain static metadata. This restriction makes it more difficult for the system to adjust in real time to shifting user preferences, which impacts its capacity to offer current, pertinent suggestions based on shifting trends and desires.

3.4.5 Advantages

1. **Content-Based Filtering:**
   - **Personalized Recommendations:** The system generates tailored movie suggestions based on the user's historical preferences, allowing for highly personalized recommendations.
   - **Simplicity:** Content-based filtering is relatively simple to implement and doesn't require complex user interaction or large datasets to function effectively.
   - **Scalability:** As new movies are added to the system, they can be easily recommended to users based on their attributes (like genre or director), making the system scalable.
2. **Handling Missing Data:**
   - **Data Completeness:** Replacing missing values with the most frequent or average values ensures that the system continues to function even when some data is incomplete, improving the robustness of the dataset.
3. **Categorical Encoding:**
   - **Data Processing:** Converting categorical variables like genres and directors into numerical values using one-hot encoding allows for easier analysis and processing by machine learning algorithms.
4. **Manual Feature Engineering:**
   - **Domain Expertise:** Leveraging domain knowledge for feature selection ensures that the most relevant features for movie recommendations are considered, leading to more accurate predictions.
5. **Scaling:**
   - **Normalization:** Using scaling techniques like Min-Max Scaling ensures that features with different ranges (e.g., ratings and popularity) are treated equally, preventing certain features from disproportionately influencing recommendations.

3.4.6 Disadvantages:

1. **Cold Start Problem:**
   - **Limited Recommendations for New Users/Movies:** The system struggles to provide relevant recommendations when there's

insufficient data, such as for new users who haven’t watched enough movies or new movies that lack user interaction.

2. **Limited Dimensionality Reduction:**
    - **Model Complexity:** Without advanced dimensionality reduction techniques like PCA, the system may suffer from increased complexity and inefficiency, especially as the dataset grows, making the recommendation process slower and more resource-intensive.
3. **Narrow Recommendations:**
    - **Limited Personalization:** Since content-based filtering only considers item attributes, it may recommend movies that are too similar to what the user has already watched, leading to less diverse and potentially less interesting suggestions.
4. **Static Data:**
    - **Lack of Real-Time Adaptation:** The system’s reliance on static movie metadata (without incorporating real-time user behavior) means it cannot quickly adapt to evolving user preferences or emerging trends, which can limit the relevance of recommendations over time.
5. **No Collaborative Filtering:**
    - **Missed Collaborative Insights:** By not incorporating collaborative filtering, the system misses the opportunity to leverage the collective preferences of similar users, potentially leading to less accurate recommendations in cases where the content-based filtering alone is insufficient.

CHAPTER 4
4. PROPOSED WORK

4.1 Description of Proposed Work

By combining Long Short-Term Memory (LSTM) networks with Content-Based Filtering, the suggested movie recommendation system seeks to increase the relevance and accuracy of movie recommendations. Long-term dependencies between films that people have viewed over time can be captured using the LSTM model, which works especially well with sequential data. The algorithm will be able to recommend films with similar patterns by using LSTM to better understand user preferences based on their past movie-watching experiences. In addition, content-based filtering will be utilised to suggest films according to director, actors, and genre. The recommendation system will perform better and be more responsive to changing user preferences thanks to this hybrid method.

4.2 LSTM (Long Short-Term Memory) Algorithm

- LSTM is a type of recurrent neural network (RNN) that can learn and remember long-term dependencies in sequences of data. It is widely used for sequence data like time series or natural language, where context is important.
- LSTMs have internal mechanisms called **gates** that regulate the flow of information:
    1. **Forget Gate**: Decides what information to throw away.
    2. **Input Gate**: Decides what information to add to the current memory.
    3. **Output Gate**: Decides what part of the memory to output.

Forget Gate Formula:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Input Gate Formula:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

Output Gate Formula:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

Cell State Update:

$$C_t = f_t * C_{t-1} + i_t * \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

4.3 System Architectural Design

The integration of the LSTM network, content-based filtering, and data preprocessing are some of the essential elements that make up the system design. The first step in the workflow is the collection and preprocessing of movie data, which includes ratings, user preferences, and movie information. In addition to handling missing values, category data—like actors and genres—is encoded for processing. Then, to ensure uniformity across characteristics like popularity, ratings, and user-specific preferences, feature scaling is used to normalise values. One part of the dataset is utilised to train the LSTM network and content-based filtering models, while the other part is set aside for testing. The dataset is divided into training and testing subsets. While the content-based filtering method generates initial recommendations by comparing user preferences with movie attributes, the LSTM model is trained to anticipate user movie preferences based on sequential viewing behaviour and previous ratings. Lastly, metrics such as precision, recall, and Mean Absolute Error (MAE) are used to evaluate the performance of both models. The accuracy of the LSTM model is also tested over time.

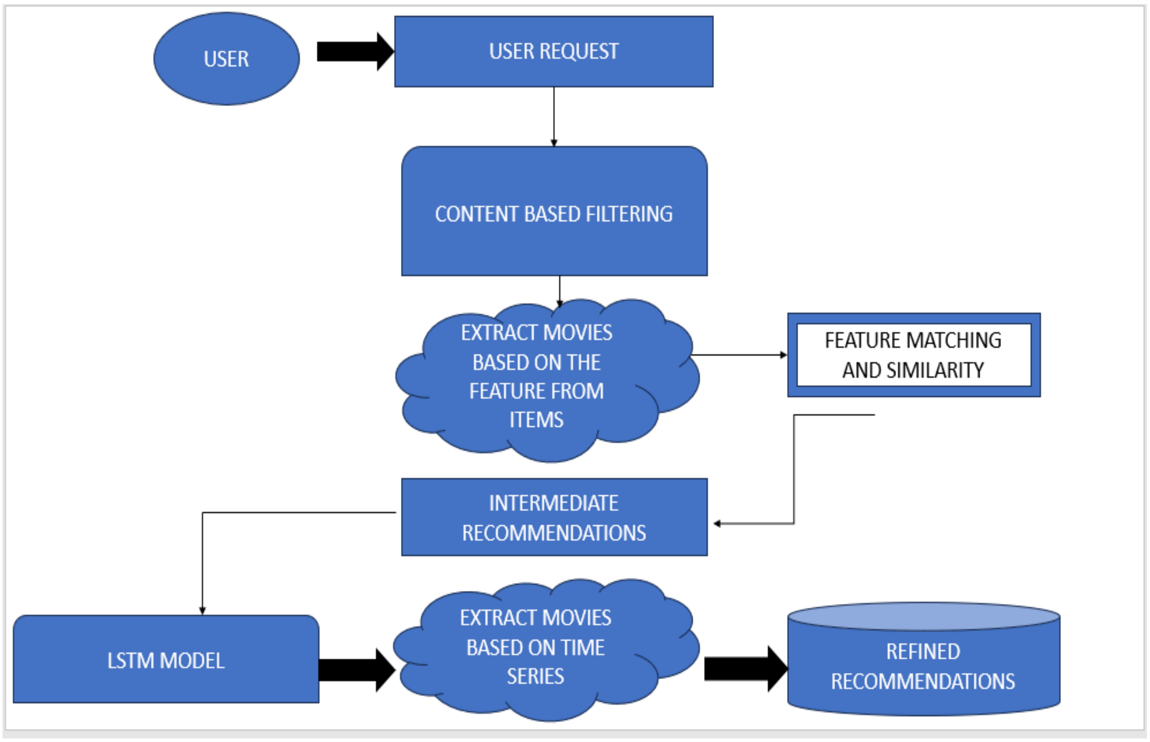

USER
USER REQUEST
CONTENT BASED FILTERING
EXTRACT MOVIES BASED ON THE FEATURE FROM ITEMS
FEATURE MATCHING AND SIMILARITY
INTERMEDIATE RECOMMENDATIONS
LSTM MODEL
EXTRACT MOVIES BASED ON TIME SERIES
REFINED RECOMMENDATIONS

Figure 1.4 System architecture proposed

Figure 1.4 illustrates a hybrid recommendation system that combines content-based filtering with time series analysis using an LSTM (Long Short-Term Memory) model. This hybrid model enhances recommendation accuracy by integrating static feature matching and dynamic time-based predictions.

4.4 Modules in Proposed Work

4.4.1 Data Preprocessing

During the data preprocessing stage, methods such as mean imputation for numerical values and mode imputation for categorical variables are used to replace missing values in the dataset. Using techniques like One-Hot Encoding, categorical variables—such genres, languages, or other movie metadata—are converted into numerical representations that can be processed by models. Furthermore, to make sure that all characteristics are on the same scale, continuous variables—like movie ratings—are normalised using methods like Min-Max Scaling. This enhances the model's overall effectiveness and performance.

4.4.2 Feature Extraction Model

Features such as genre, director, actor, and plot are taken from the movie database as part of the content-based feature extraction process. The similarity between films and the user's tastes is then determined using these features. TF-IDF (Term Frequency-Inverse Document Frequency) is used for textual characteristics like movie reviews or descriptions. By decreasing the influence of frequently used words and making sure that more pertinent phrases contribute more to the similarity calculation, this strategy helps weight words according to their value within the text, thus increasing the accuracy of recommendations.

4.4.3 Classification and Recommendation

Long Short-Term Memory (LSTM) is used by the system to record temporal patterns in user interactions, such the order in which films are viewed. More precise recommendations based on watching history are made possible by this model, which is particularly useful for examining how user preferences change over time. Furthermore, the content-based filtering system compares movie attributes with user preferences, including explicit ratings or previously seen films. Then, in order to provide individualised and pertinent recommendations, films with the highest similarity ratings to the user's tastes are suggested.

4.4.4 Datasets

The dataset used for the recommendation system includes various types of data. Movie metadata encompasses attributes such as movie title, genre, director, actors, and plot descriptions, which provide essential details about the films. User data includes information about the user, such as their ratings and a history of previously watched movies, allowing the system to understand individual preferences. Additionally, optional attributes like user feedback, reviews, or session length can be included to further enhance personalization. The dataset is split into training and testing sets (80%-20%) to evaluate model performance using metrics such as accuracy, precision, recall, and F1-score, ensuring the system's effectiveness in making recommendations.

4.4.5 Advantages of LSTM:

1. **Handling Sequential Data:**
   - **Advantage over CBF:** LSTM is great at handling sequential and temporal data. It can remember long-term dependencies, making it more effective for recommending movies based on the sequence of movies a user has watched over time, which CBF struggles with since CBF only considers the features of individual items (movies) without regard to the sequence or context.
2. **Context Awareness:**
   - **Advantage over CBF:** LSTM models understand the context better by learning patterns in a sequence of actions (like watching a movie series or a genre over time). Content-based systems don't capture user preferences in context and only recommend items similar to previously watched ones based on content features.
3. **Better Personalization:**
   - **Advantage over CBF:** LSTM can personalize recommendations better since it considers the entire history of user interactions, enabling more dynamic recommendations that adapt over time. In contrast, CBF mainly focuses on static features (like genre, director, etc.) of the movies the user has watched.
4. **Overcoming Cold-Start Problem:**
   - **Advantage over CBF:** LSTM can handle situations where the user has only seen a few movies, by learning patterns over time. Content-based filtering can struggle with cold-start problems when a user has minimal data or if a movie doesn’t have enough features.

CHAPTER 5

5. SIMULATION / EXPERIMENTAL RESULT

5.1 HARDWARE AND SOFTWARE REQUIREMENTS

- **Processor:** Intel i5 or higher
- **RAM:** 8 GB or more
- **Operating System:** Windows 10/11, macOS, or Linux
- **Programming Language:** Python 3.7 or above
- **Libraries/Frameworks:**

  TensorFlow/Keras: For implementing the LSTM (Long Short-Term Memory) model.

  scikit-learn: For implementing machine learning models like Decision Tree and Rotation Forest.

  NumPy: For numerical operations.

  Pandas: For data preprocessing and manipulation.

  Matplotlib & Seaborn: For visualizing results and analyzing data patterns.

  NLTK (Natural Language Toolkit): For preprocessing textual data (if movie descriptions are included).

5.2 EVALUATION METRICS

5.2.1 Confusion Matrix:

- A tabular representation of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).
- Helps in understanding the performance of the recommendation system, especially for relevant vs. irrelevant recommendations.

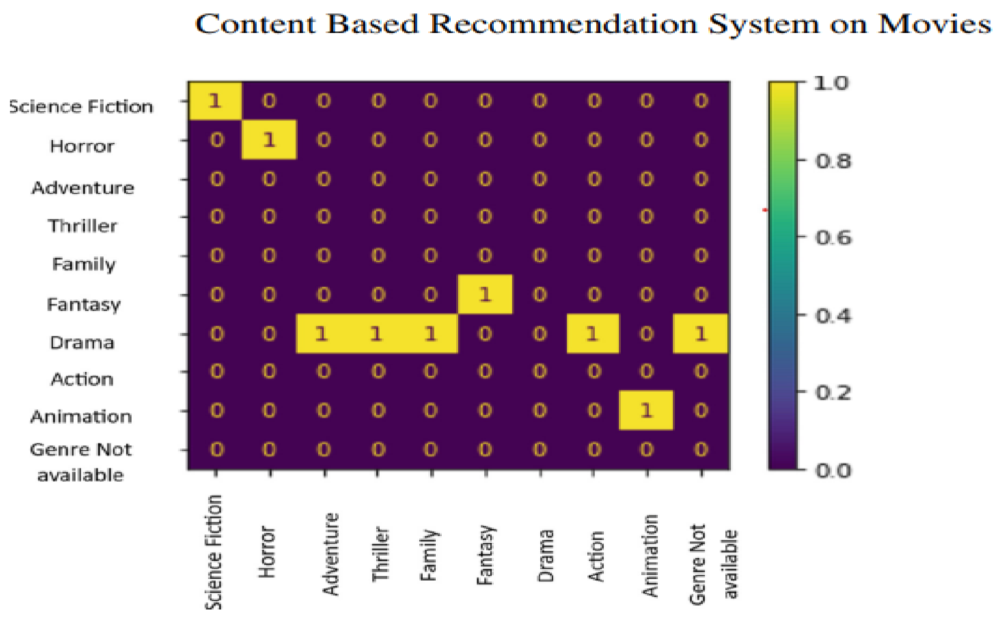


Figure 1.7 Confusion Matrix on target and recommended genres on a particular movie

Figure 1.7 explains the Confusion Matrix is used to evaluate the performance of a classification model by comparing actual target values (true labels) with predicted values (recommendations). In the context of target and recommended genres on a particular movie, the confusion matrix helps analyze how well the recommendation system predicts the correct genres for that movie.

5.2.2 Accuracy:

Indicates the proportion of correctly recommended movies (relevant vs. irrelevant recommendations) over the total number of recommendations.

Comparing Models and Visualizing Accuracy

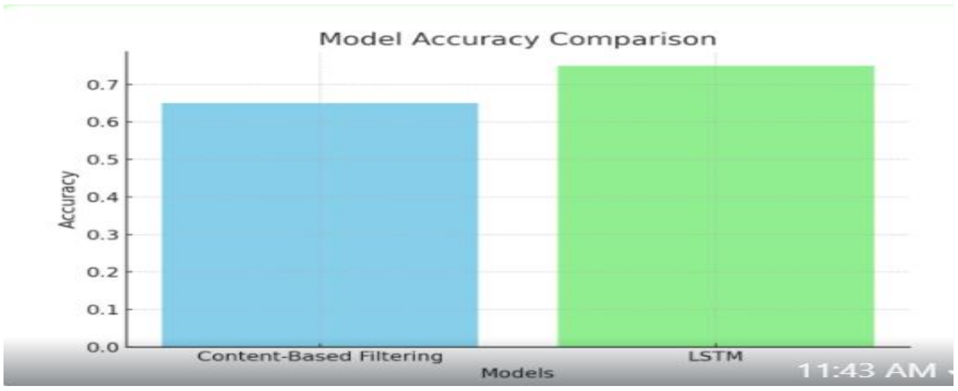


Figure 1.6 Bar Graph Comparing Model Accuracies

Figure 1.6 illustrates comparing the accuracy between the two algorithms (content-based filtering and Long Short Term Memory algorithms)

5.2.3 Precision:

Measures the proportion of relevant recommendations compared to the total number of recommendations.

5.2.4 Recall (Sensitivity):

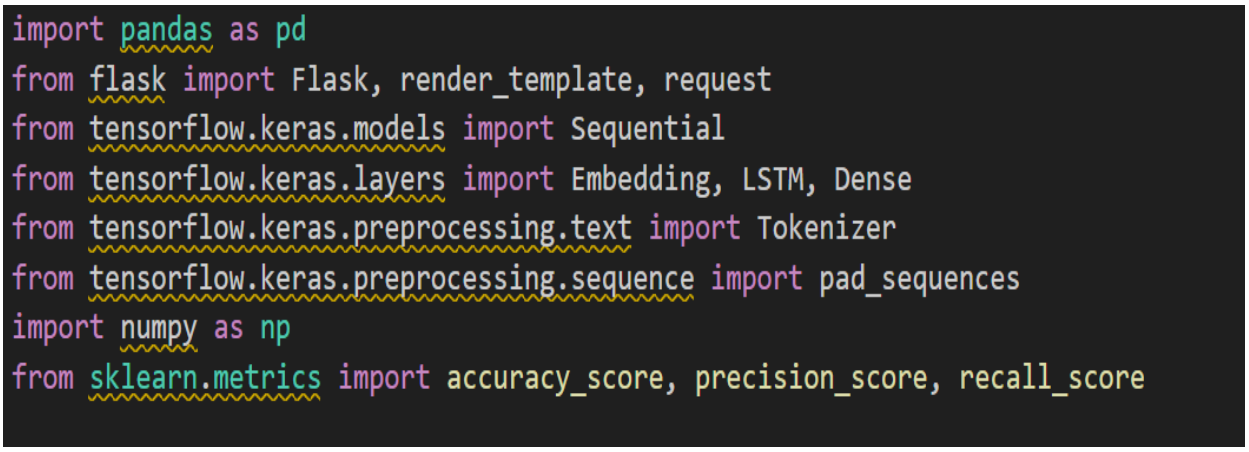
Measures the system's ability to recommend all relevant movies from the user's preferences.

5.2.5 F1-Score:

A harmonic mean of precision and recall, useful in situations where the dataset is imbalanced (e.g., there may be more irrelevant recommendations than relevant ones).

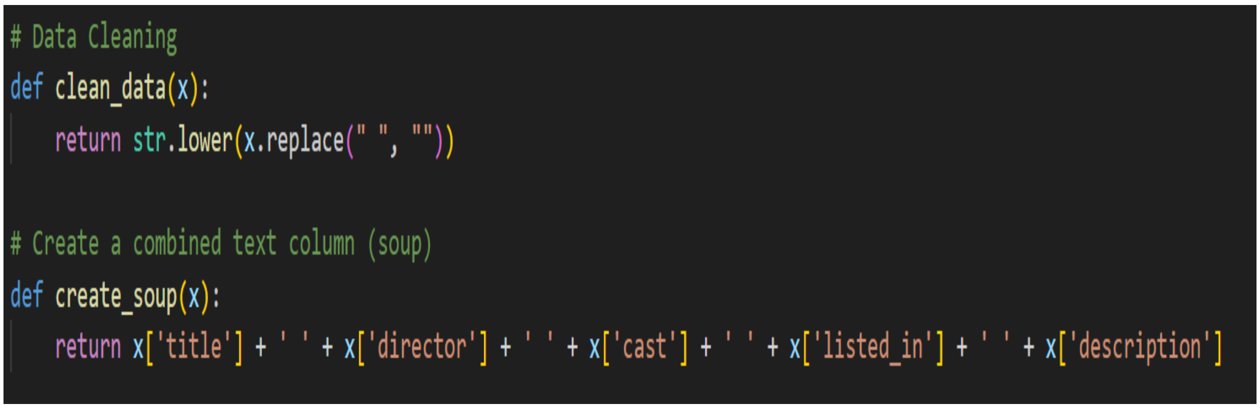
5.3 IMPLEMENTATION

5.3.1 Importing Libraries



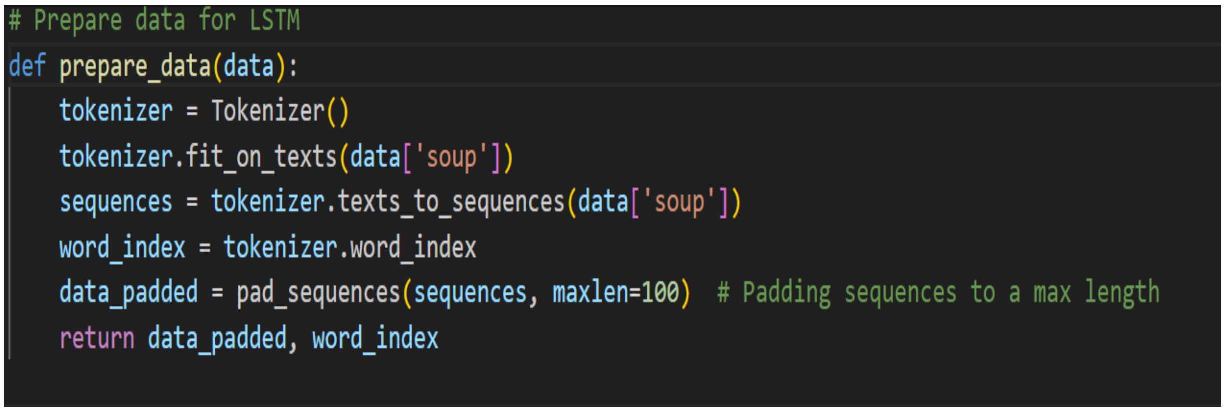
```
import pandas as pd
from flask import Flask, render_template, request
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

5.3.2 Exploratory Data Analysis (EDA)



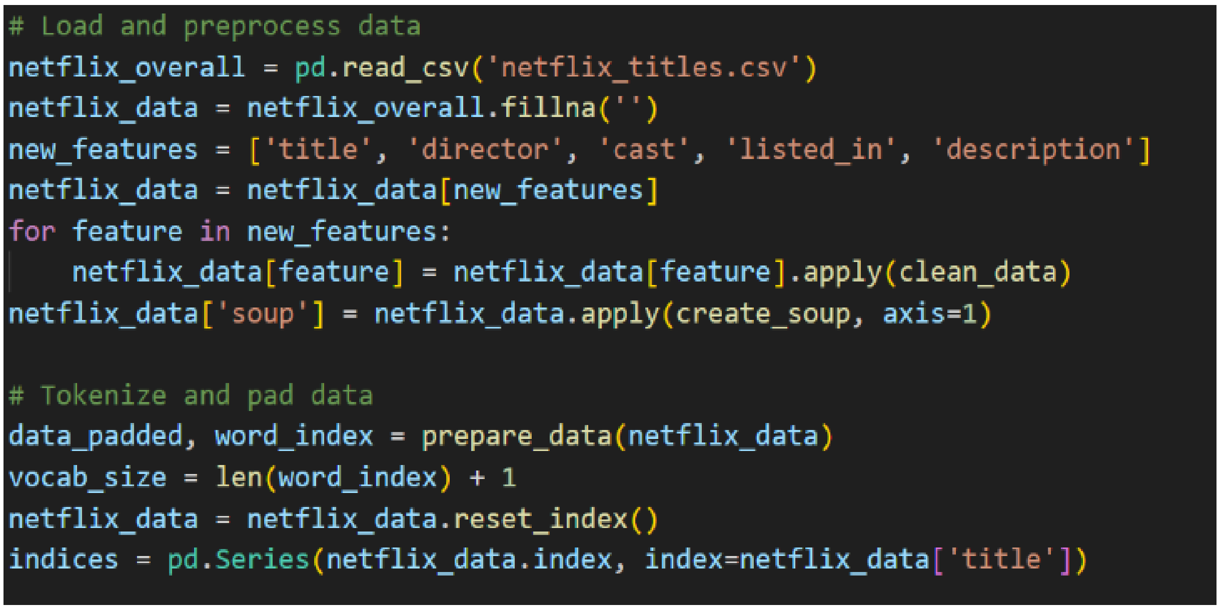
```python
# Data Cleaning
def clean_data(x):
    return str.lower(x.replace(" ", ""))

# Create a combined text column (soup)
def create_soup(x):
    return x['title'] + ' ' + x['director'] + ' ' + x['cast'] + ' ' + x['listed_in'] + ' ' + x['description']
```

5.3.4 Data Preprocessing



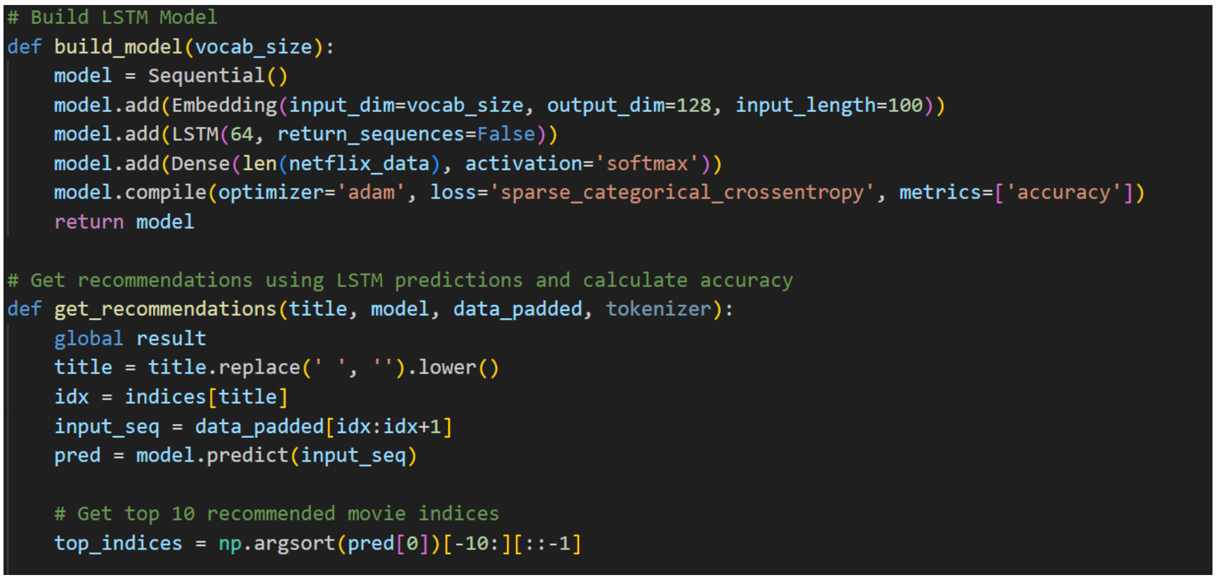
```python
# Prepare data for LSTM
def prepare_data(data):
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(data['soup'])
    sequences = tokenizer.texts_to_sequences(data['soup'])
    word_index = tokenizer.word_index
    data_padded = pad_sequences(sequences, maxlen=100)  # Padding sequences to a max length
    return data_padded, word_index
```

5.3.5 Splitting and Scaling Data

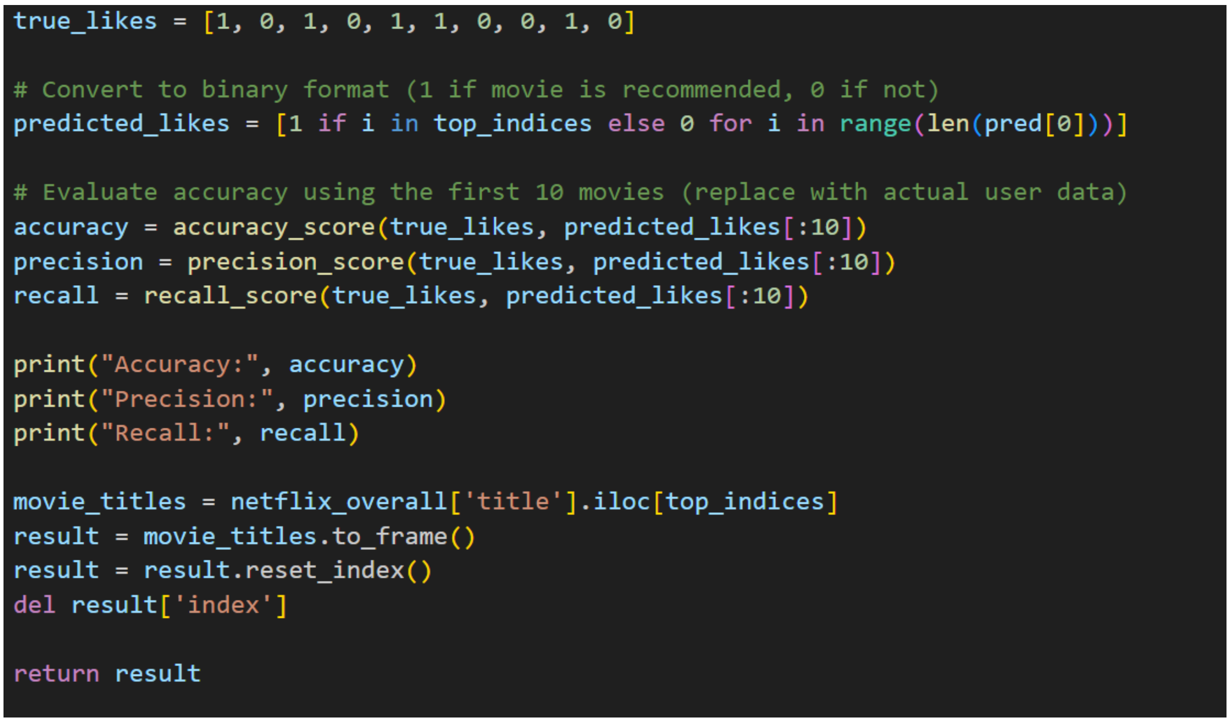


```python
# Load and preprocess data
netflix_overall = pd.read_csv('netflix_titles.csv')
netflix_data = netflix_overall.fillna('')
new_features = ['title', 'director', 'cast', 'listed_in', 'description']
netflix_data = netflix_data[new_features]
for feature in new_features:
    netflix_data[feature] = netflix_data[feature].apply(clean_data)
netflix_data['soup'] = netflix_data.apply(create_soup, axis=1)

# Tokenize and pad data
data_padded, word_index = prepare_data(netflix_data)
vocab_size = len(word_index) + 1
netflix_data = netflix_data.reset_index()
indices = pd.Series(netflix_data.index, index=netflix_data['title'])
```

5.3.6 Training and Evaluating LSTM

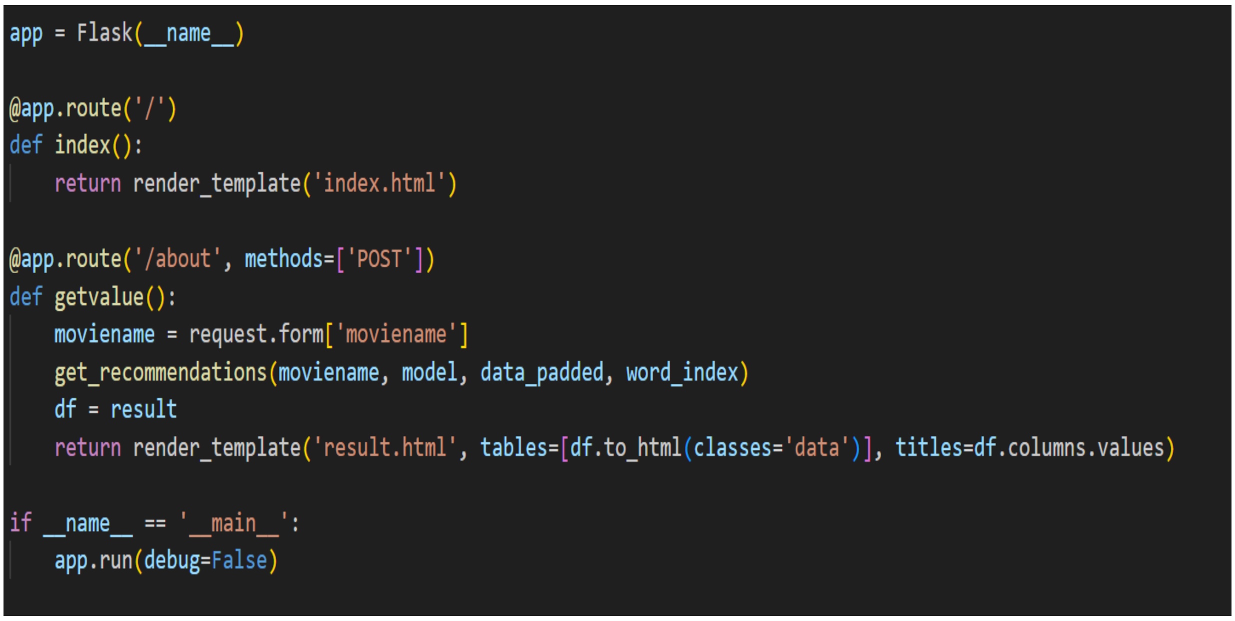


```python
# Build LSTM Model
def build_model(vocab_size):
    model = Sequential()
    model.add(Embedding(input_dim=vocab_size, output_dim=128, input_length=100))
    model.add(LSTM(64, return_sequences=False))
    model.add(Dense(len(netflix_data), activation='softmax'))
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

# Get recommendations using LSTM predictions and calculate accuracy
def get_recommendations(title, model, data_padded, tokenizer):
    global result
    title = title.replace(' ', '').lower()
    idx = indices[title]
    input_seq = data_padded[idx:idx+1]
    pred = model.predict(input_seq)

    # Get top 10 recommended movie indices
    top_indices = np.argsort(pred[0])[-10:][::-1]
```



```python
true_likes = [1, 0, 1, 0, 1, 1, 0, 0, 1, 0]

# Convert to binary format (1 if movie is recommended, 0 if not)
predicted_likes = [1 if i in top_indices else 0 for i in range(len(pred[0]))]

# Evaluate accuracy using the first 10 movies (replace with actual user data)
accuracy = accuracy_score(true_likes, predicted_likes[:10])
precision = precision_score(true_likes, predicted_likes[:10])
recall = recall_score(true_likes, predicted_likes[:10])

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)

movie_titles = netflix_overall['title'].iloc[top_indices]
result = movie_titles.to_frame()
result = result.reset_index()
del result['index']

return result
```

5.3.7 Training and Evaluating Content based filtering



```
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/about', methods=['POST'])
def getvalue():
    moviename = request.form['moviename']
    get_recommendations(moviename, model, data_padded, word_index)
    df = result
    return render_template('result.html', tables=[df.to_html(classes='data')], titles=df.columns.values)

if __name__ == '__main__':
    app.run(debug=False)
```

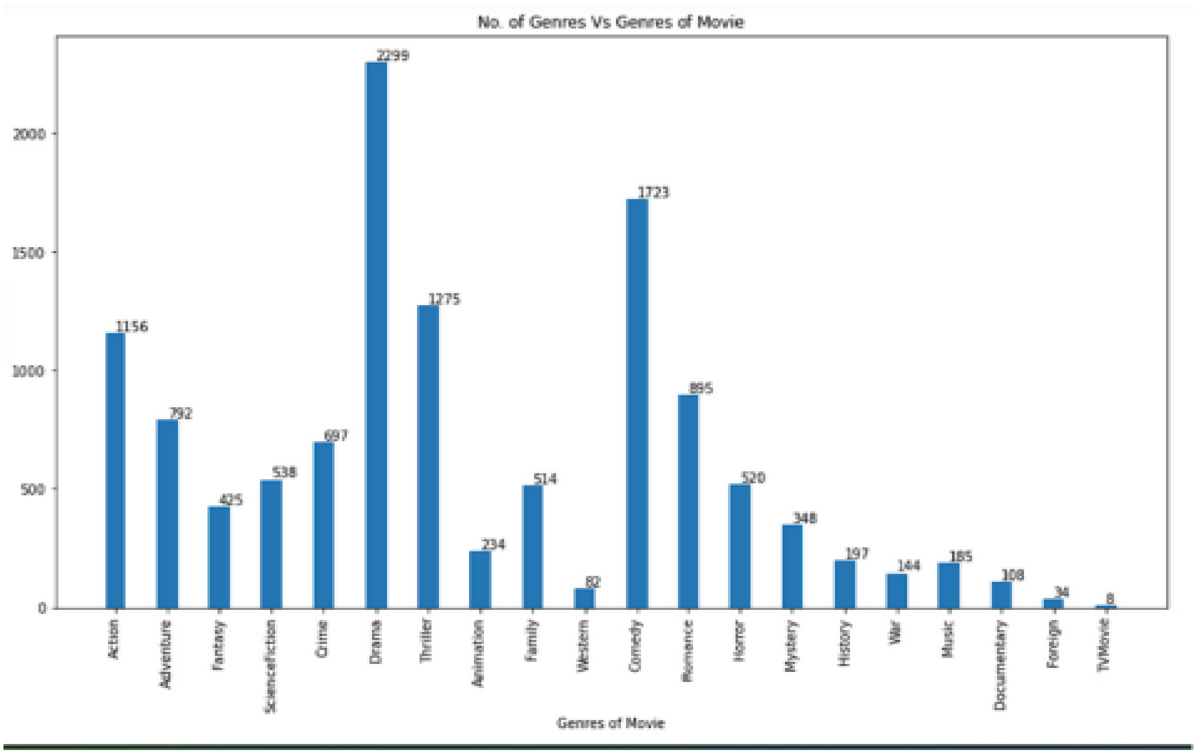


Figure 1.8 Different Genres Of Movies In a Dataset

Figure 1.8 approach makes it easy to filter and categorize movies based on genre, whether you're conducting a movie recommendation system, analyzing trends, or exploring relationships between genres and movie success.

OUTPUT OF CONTENT BASED FILTERING ALGORITHM

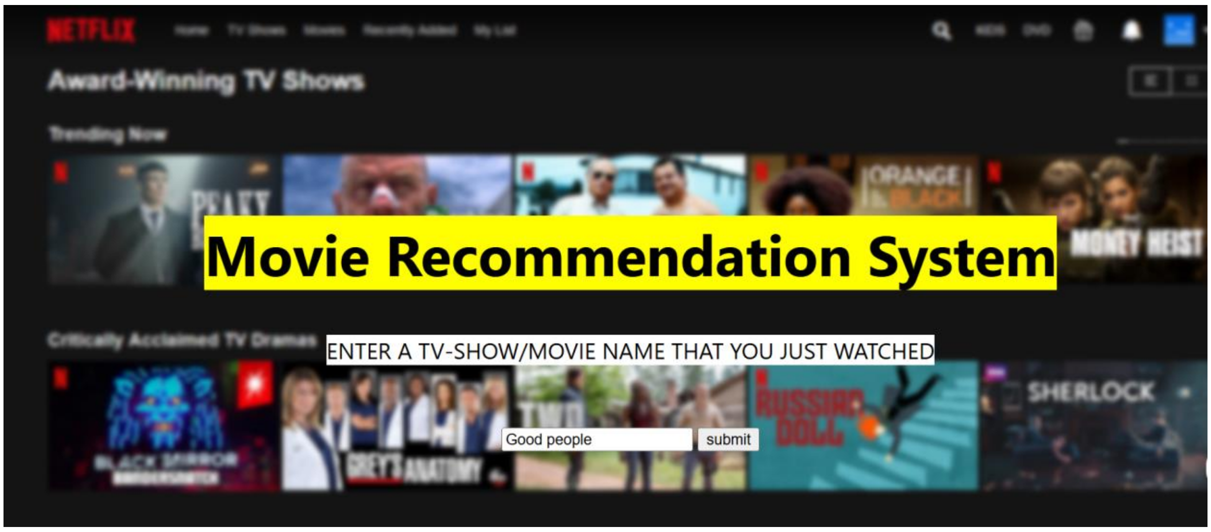
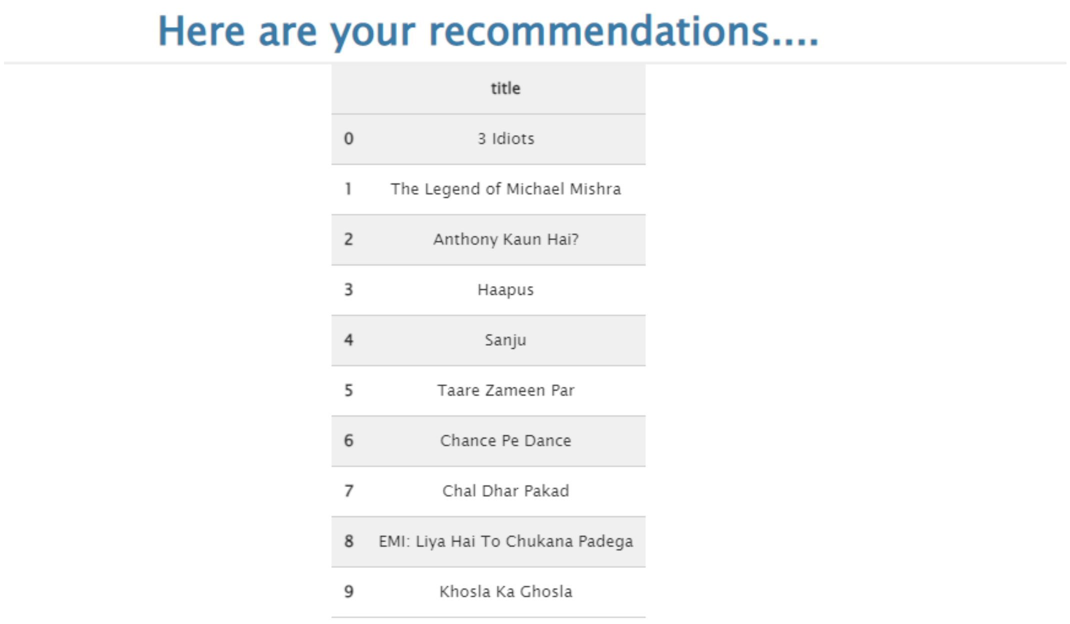

Movie Recommendation System
ENTER A TV-SHOW/MOVIE NAME THAT YOU JUST WATCHED
Good people
submit

FIGURE 1.9 (a)

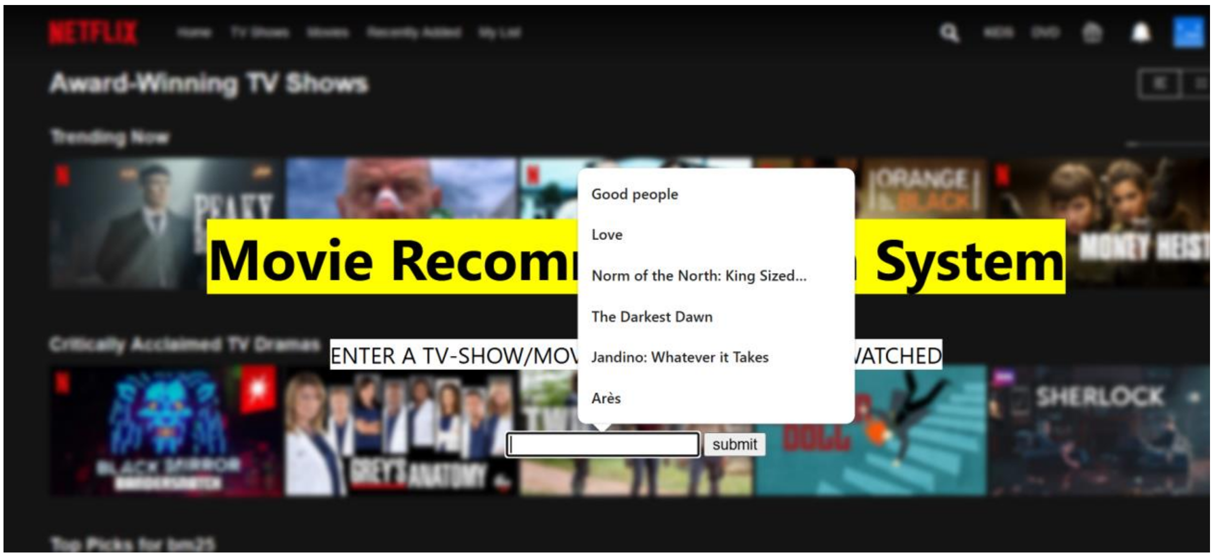


Here are your recommendations....

	title
0	3 Idiots
1	The Legend of Michael Mishra
2	Anthony Kaun Hai?
3	Haapus
4	Sanju
5	Taare Zameen Par
6	Chance Pe Dance
7	Chal Dhar Pakad
8	EMI: Liya Hai To Chukana Padega
9	Khosla Ka Ghosla

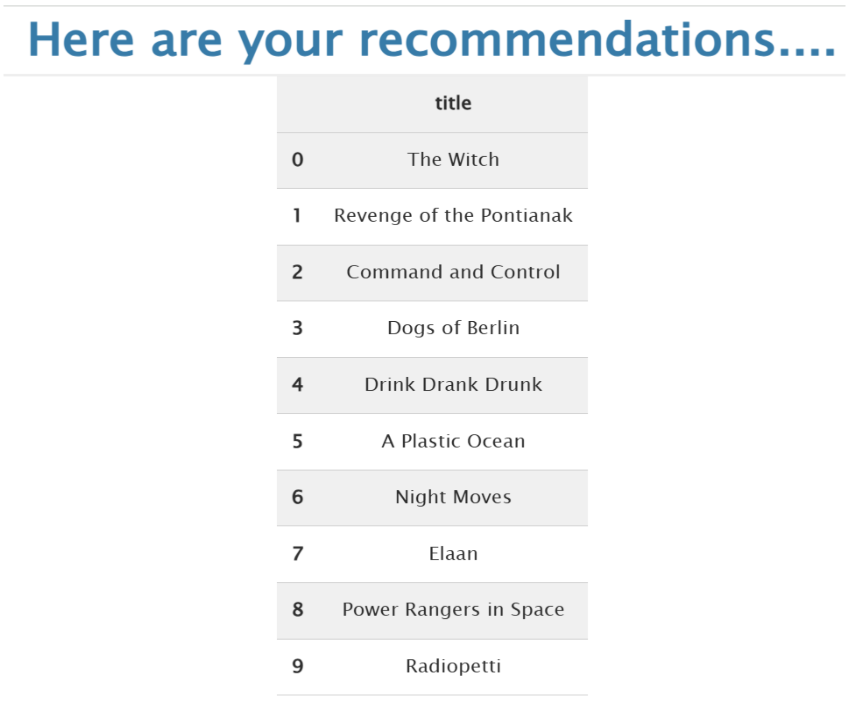
FIGURE 1.9 (b)

OUTPUT OF CBF AND LSTM ALGORITHM



Movie Recomm System
ENTER A TV-SHOW/MOV /ATCHED
Good people
Love
Norm of the North: King Sized...
The Darkest Dawn
Jandino: Whatever it Takes
Arès
submit

FIGURE 1.10 (a)



Here are your recommendations....

	title
0	The Witch
1	Revenge of the Pontianak
2	Command and Control
3	Dogs of Berlin
4	Drink Drank Drunk
5	A Plastic Ocean
6	Night Moves
7	Elaan
8	Power Rangers in Space
9	Radiopetti

FIGURE 1.10 (b)

6.CONCLUSION

This study explores the application of Content-Based Filtering and Long Short-Term Memory (LSTM) networks for movie recommendations. Content-Based Filtering leverages movie attributes, such as genre, director, and actors, to provide personalized recommendations based on item characteristics, making it suitable for capturing the preferences of users for known items. On the other hand, the LSTM model, a type of recurrent neural network, captures sequential patterns and temporal dependencies in user behavior, providing more accurate and dynamic recommendations over time.

The evaluation of both methods was performed using metrics such as precision, recall, Mean Absolute Error (MAE), and F1-score. Content-Based Filtering demonstrated effective personalization based on item attributes, while the LSTM model enhanced prediction accuracy by understanding the evolving patterns of user interactions and preferences. The combination of these methods showed promising results, where LSTM improved the diversity and relevance of recommendations.

This work highlights the importance of integrating machine learning techniques like Content-Based Filtering and LSTM in recommendation systems to capture both static item features and dynamic user preferences. Future research could explore hybrid approaches, deep learning techniques such as transformers, and multi-modal datasets to further enhance recommendation performance and accuracy.

6.1 Future work

Incorporating the Transformer Algorithm in the Movie Recommendation System

For improving the performance of the movie recommendation system, the Transformer algorithm has been integrated into the model. The Transformer, introduced in the paper *"Attention is All You Need"*, is a deep learning model that has revolutionized sequence data processing. Unlike traditional models such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, the Transformer leverages a mechanism known as **self-attention** to efficiently capture long-range dependencies in data.

The core strength of the Transformer lies in its ability to process entire sequences simultaneously, rather than sequentially, allowing it to consider all elements in a sequence in parallel. This parallelization significantly reduces training time and enhances the model’s ability to capture complex relationships between the input data. The self-attention mechanism enables the model to weigh the importance of each element in the input sequence with respect to the others, allowing it to focus on the most relevant parts of the data.

In the context of the movie recommendation system, the Transformer model processes the sequence of movies a user has watched, learning long-term preferences and recognizing non-linear relationships between movies, even if they are not immediately adjacent in the user’s viewing history. This results in more personalized recommendations by identifying subtle patterns in a user’s behavior over time, which might be overlooked by traditional content-based or collaborative filtering methods.

The integration of the Transformer algorithm enhances the system’s ability to handle large datasets efficiently while improving the quality of recommendations. It provides a significant advantage in capturing the complexity of user preferences, making the model more adaptive and accurate in recommending movies that align with a user's evolving tastes.

7.REFERENCES

1. Wang, H., Lou, N., & Chao, Z. (2020). A personalized movie recommendation system based on LSTM-CNN. In *2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)* (pp. 485–490). https://doi.org/10.1109/MLBDBI51377.2020.00102

2. Fanca, A., Puscasiu, A., Gota, D.-I., & Valean, H. (2020). Recommendation systems with machine learning. In *2020 21st International Carpathian Control Conference(ICCC)* (pp. 1–6).https://doi.org/10.1109/ICCC49264.2020.9257290

3. Lund, J., & Ng, Y.-K. (2018). Movie recommendations using the deep learning approach. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)* (pp. 47–54).https://doi.org/10.1109/IRI.2018.00015

4. Ahmed, M., Imtiaz, M. T., & Khan, R. (2018). Movie recommendation system using clustering and pattern recognition network. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 143–147). https://doi.org/10.1109/CCWC.2018.8301695

5. Immanuvel, J., Sheelavathi, A., Priyadharshan, M., Vignesh, S., & Elango, K. (n.d.). Movie recommendation system.

6. Afoudi, Y., Lazaar, M., & Al Achhab, M. (2019). In *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*.

7. Badriyah, T., Azvy, S., Yuwono, W., & Syarif, I. (2018). In *2018 International Conference on Information and Communications Technology (ICOIACT)*.

8. Zhu, Q., Shyu, M.-L., & Wang, H. (2013). In *2013 IEEE International Symposium on Multimedia*.

9. Kayaalp, M., Ozyer, T., & Tariyan Ozyer, S. (2009). In *2009 International Conference on Advances in Social Network Analysis and Mining*.

10. Yang, Y., Jang, H.-J., & Kim, B. (2020). IEEE access. *IEEE Access*, 8.

11. Parmar, D. (2018). In *2018 2nd International Conference on Inventive Systems and Control (ICISC)*.

12. Pal, A., Parhi, P., & Aggarwal, M. (2017). In *2017 Tenth International Conference on Contemporary Computing (IC3)*.

13. Sharma, S., Sharma, A., Sharma, Y., & Bhatia, M. (2016). In *2016 International Conference on Computing, Communication and Automation (ICCCA)*.

14. Jain, S., Grover, A., Thakur, P. S., & Choudhary, S. K. (2015). In *International Conference on Computing, Communication & Automation*.

15. Funakoshi, K., & Ohguro, T. (2000). In *KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No.00TH8516)*.

16. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8), 30–37. https://doi.org/10.1109/MC.2009.263

17. Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). Recommender systems: Challenges and research opportunities. In *Proceedings of the 2nd ACM Conference on Electronic Commerce* (pp. 163–166). https://doi.org/10.1145/501158.501188

18. Rendle, S. (2010). Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining* (pp. 995–1000). https://doi.org/10.1109/ICDM.2010.127

19. Zhang, Y., & Yao, L. (2019). Deep learning for recommender systems. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (pp. 2335–2336). https://doi.org/10.1145/3357384.3358037

20. Cheng, X., & Zhang, X. (2018). Movie recommendation based on LSTM. In *Proceedings of the 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis* (pp. 396–400). https://doi.org/10.1109/ICCCBDA.2018.8386563

21. Gantner, Z., Mladenic, D., & Grobelnik, M. (2010). Content-based recommender systems with LSTM networks. In *Proceedings of the European Conference on Machine Learning* (pp. 74–85).

22. Suwajanakorn, S., & Lee, A. (2017). LSTM-based recommendation systems: A comparative study. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(9), 4532–4541. https://doi.org/10.1109/TNNLS.2017.2716148

23. Basilico, J., & Melchiorre, A. (2016). Collaborative filtering and content-based recommendation techniques for movie recommendation. *ACM Computing Surveys*, *49*(2), 1–41. https://doi.org/10.1145/2932580