??
article

A

PROJECT STAGE II REPORT ON

"Content-Based Movie Recommender System"

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

BACHLOR OF ENGINEERING (Computer Engineering)

BY

Mr. Khedkar Laxman Bhimrao. EXAM NO: - B400990090
Mr. Hase Ashutosh Prakash. EXAM NO: - B400990080
Mr. Langde Ritesh Vitthalrao. EXAM NO: - B400990093

UNDER THE GUIDANCE OF

Prof.Mrs.Khatal K.B

DEPARTMENT OF COMPUTER ENGINEERING
Sahyadri Valley College of Engineering And Technology, Rajuri
A/P-Rajuri-412411, Tal. Junner, Dist. pune (MS) India

2024-2025

SAHYADRI VALLEY COLLEGE OFENGINEERING AND TECHNOLOGY, RAJURI

DEPARTMENT OF COMPUTER ENGINEERING

A/P - Rajuri, Tal. Junnar, Dist. Pune (MH) - 412411

CERTIFICATE

This is to certify that the dissertation report entitled

"Content-Based Movie Recommender System"

Submitted by

Mr. Khedkar Laxman Bhimrao EXAM NO :- B400990090

is a bonafide work carried out under the supervision of **Prof. K.B. Khatal** and is submitted towards the partial fulfillment of the requirement of Savitribai Phule Pune University, Pune for the award of the degree of Bachelor of Engineering (Computer Engineering).

Prof. K. B. Khatal
(Internal Guide)
Dept. of Computer Engg.

Prof. K. B. Khatal
(Head)
Dept. of Computer Engg.

Prof. P. Balaramudul
(Vice Principal)
SVCET, Rajuri.

Dr. S. B. Zope
(Principal)
SVCET, Rajuri.

Savitribai Phule Pune University

CERTIFICATE

This is to Certify that

MR. Khedkar Laxman Bhimrao
MR. Hase Ashutosh Prakash
MR. Langde Ritesh Vitthalrao

Student of B.E. Computer was examined in

Project Stage II Report

"Content-Based Movie Recommender System"

on .../... /2025

At

Department of Computer Engineering,
Sahyadri valley College of Engineering And Technology Rajuri,
Rajuri - 412411

.....	.....
Internal Examiner	External Examiner
(Prof.Khatal K.B)	(Prof.)

Certificate By Guide

This is to certify that [BE-009] has completed the Project Stage 2 work under my guidance and supervision and that I have verified the work for its originality in documentation, problem statement, implementation and results presented in the dissertation. Any reproduction of other necessary work is with the prior permission and has given due ownership and included in the references.

Place: Rajuri **(Prof. Mrs. Khatal K.B)**

Date: June 12, 2025

ACKNOWLEDGEMENT

"Gratitude unlocks the fullness of life."

I take this opportunity to express my heartfelt gratitude to all those who supported and guided me throughout this project.

I am especially thankful to my guide, Prof. Mrs. Khatal K.B., for her constant encouragement, valuable guidance, and dedicated supervision during the entire course of this project.

I also wish to thank Prof. Mrs. Khatal K.B., Head of the Computer Engineering Department, for her insightful suggestions and motivation.

My sincere thanks to Dr. S.B. Zope, Principal, and Prof. P. Balaramudu, Vice Principal, for their encouragement and support.

I would also like to thank all the faculty and staff of Sahyadri Valley College of Engineering and Technology for their help, and extend special thanks to my family, friends, and teammates for their moral support throughout this journey.

Mr. Khedkar Laxman Bhimrao
Mr. Hase Ashutosh Prakash
Mr. Langde Ritesh Vitthalrao
SVCET, Rajuri

List of Tables

ABSTRACT

This project presents a Movie Recommender System developed using Streamlit, leveraging a dataset of approximately 5,000 films and standard Python libraries (NumPy, Pandas, Scikit-learn, NLTK). The aim is to assist users in discovering relevant movies within a large catalogue by providing personalized suggestions. The system implements both content-based filtering—computing cosine similarity on feature vectors derived from metadata such as genres, keywords, and textual summaries preprocessed with NLTK—and collaborative filtering techniques to capture user–item interaction patterns where available. Data preprocessing involves cleaning and vectorizing textual fields, handling missing values, and normalizing numeric attributes. The recommendation engine combines similarity scores to rank candidate movies for a given user profile or input movie. A Streamlit-based interface allows users to input preferences (e.g., favorite titles or selected genres) and view recommended titles with brief details. Preliminary evaluation demonstrates that the hybrid approach yields coherent recommendations aligned with user interests and diverse suggestions across genres. Limitations include dependence on the scope of available metadata and cold-start issues for new items or users. Future enhancements may incorporate additional features (e.g., user ratings, social context), more advanced NLP embeddings, and deeper evaluation with user studies. Overall, this system offers a practical, extensible framework for personalized movie recommendation in an interactive web app.

Keywords (5) – Movie Recommendation, Content-based Filtering, Collaborative Filtering, Cosine Similarity, Streamlit

INDEX

SYNOPSIS

Project Title: "Content-Based Movie Recommender System".

Internal Guide: Prof.Mrs Khatal K.B

Existing System

Movie recommendation systems play a crucial role in enhancing user experience by tailoring content suggestions. Collaborative filtering methods leverage user interactions—such as ratings or viewing history—to find patterns among users or items. While effective, they can struggle with cold-start scenarios (new users/items) and sparse feedback. Content-based filtering, in contrast, analyzes item attributes (e.g., genre, cast, director, plot keywords) to suggest movies similar to those a user has liked. This approach is less affected by cold-start issues but may yield recommendations that lack diversity, as it focuses on items closely matching prior preferences. Hybrid systems aim to balance these trade-offs by combining collaborative and content-based signals, integrating multiple data sources to improve accuracy and recommendation variety. Major streaming platforms like Netflix and Amazon Prime Video employ evolving hybrid algorithms that also factor in contextual data (time, device, etc.). This project focuses on a content-based approach using cosine similarity on movie metadata, packaged in a Streamlit web application. By preprocessing and vectorizing textual features with NLTK and other Python libraries, the system delivers personalized suggestions while remaining straightforward to deploy and extend in future iterations.

Motivation In today's digital world, users are overwhelmed with an enormous amount of content available on streaming platforms, making it difficult to decide what to watch next. This inspired me to develop a content-based movie recommendation sys-

tem that could help users discover movies tailored to their personal tastes based on features like genre, keywords, and descriptions. My goal was not only to enhance the movie-watching experience but also to apply data science techniques to a real-world problem. This project allowed me to explore text processing, similarity algorithms, and machine learning libraries in Python, deepening my technical skills. Additionally, working on this system motivated me because I've always been fascinated by how platforms like Netflix and Amazon recommend content, and building a similar system from scratch gave me a sense of achievement and practical understanding of recommender systems.

Problem Statement:

In the digital era, users are overwhelmed by the vast number of movies available across different platforms. Finding a relevant movie that matches the user's taste can be time-consuming and inefficient. This project addresses the challenge by building a recommendation system that suggests movies similar to a user's input, reducing browsing time and enhancing satisfaction.

Solving Approach:

To build the content-based movie recommendation system, I started by collecting and preprocessing a dataset containing movie details such as titles, genres, overviews, cast, and keywords. The core idea was to recommend movies based on the similarity of their content. I combined important textual features into a single string for each movie and then applied Natural Language Processing (NLP) techniques such as tokenization, lowercasing, and removing stop words. After preprocessing, I used the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer to convert text data into numerical vectors. To measure similarity between movies, I implemented the cosine similarity metric, which compares the angle between feature vectors. Based on a selected movie, the system identifies and returns the top movies with the highest similarity scores. Finally, I deployed the system using Streamlit to create an interactive user interface where users can input a movie title and receive personalized recommendations instantly.

Keywords:

1. **Content-Based Filtering**

Content-Based Filtering is a recommendation technique that suggests items (movies) based on the features or attributes of the item itself (genre, overview, cast, etc.). It does not depend on user ratings or history, but instead finds similarities between items.

2. Cosine Similarity

Cosine Similarity is a mathematical method used to measure the angle between two vectors. In this project, it compares the similarity between two movie profiles (TF-IDF vectors) and ranks them based on how close their content is.

3. Natural Language Processing (NLP)

NLP involves processing and analyzing human language data. In this project, NLP techniques such as tokenization, lowercasing, and stopword removal are used to clean and prepare text features like movie overviews and keywords.

4. TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF is a text vectorization method used to convert textual information (like movie descriptions) into numerical values. It highlights important words by giving high weight to words that appear frequently in a document but not across all documents.

5. Streamlit

Streamlit is an open-source Python framework used to build and deploy interactive web applications quickly. It is used in this project to create a user-friendly interface where users can select a movie and get instant recommendations.

Chapter 1

INTRODUCTION

General Introduction: In the age of digital streaming and vast online movie libraries, users often find it difficult to select movies that match their personal preferences. To address this challenge, recommendation systems have become essential tools on platforms like Netflix, Amazon Prime, and Hotstar. These systems analyze user behavior or item features to suggest relevant content. This project focuses on building a Content-Based Movie Recommendation System, which recommends movies based on their attributes such as genre, overview, cast, and keywords. Unlike collaborative filtering, which depends on user interactions, content-based filtering relies entirely on item features. The goal is to help users discover movies similar to the ones they like by using natural language processing (NLP) and machine learning techniques. The system is developed using Python, leveraging libraries like Pandas, Scikit-learn, and Streamlit to create a functional, interactive recommendation tool.

Need of the system: With the rapid growth of digital content and streaming platforms, users are often overwhelmed by the number of available movie choices. Manually browsing through hundreds of titles is time-consuming and inefficient. This creates a strong need for intelligent systems that can simplify the decision-making process. A Content-Based Movie Recommendation System fulfills this need by analyzing movie features and suggesting similar titles based on a user's preference. It enhances the user experience by offering personalized recommendations and reduces the chances of irrelevant suggestions. Such a system is essential not only for improving viewer satisfaction but also for increasing engagement on content platforms. Moreover, building this system provides an opportunity to apply machine learning and natural language processing techniques to a practical and user-friendly application.

.

1.1 Problem Statement

In the digital era, users are overwhelmed by the vast number of movies available across different platforms. Finding a relevant movie that matches the user's taste can be time- consuming and inefficient. This project addresses the challenge by building a recommen- dation system that suggests movies similar to a user's input, reducing browsing time and enhancing satisfaction.

1.2 Motivation

- The idea behind developing a Content-Based Movie Recommendation System stemmed from the common problem users face while choosing movies from a vast online collection. With streaming platforms offering thousands of options, it becomes difficult for users to find content that truly matches their interests. This motivated me to build a system that can recommend movies based on their content features like genre, plot, and keywords, helping users save time and enjoy a personalized experience. I was also inspired by how popular platforms like Netflix and Amazon Prime use similar systems, and I wanted to understand the working mechanism behind them. Through this project, I aimed to enhance my skills in Python, machine learning, and natural language processing, while applying them to a real-world application that combines both technology and entertainment.

1.3 Objectives

1. The main objectives of this project are to develop a content-based movie recommendation system that uses machine learning techniques to measure movie similarity. The system will be deployed as a user-friendly web application built with Streamlit, enabling users to receive accurate, real-time movie recommendations based on their preferences.

Chapter 2

LITERATURE SURVEY

2.1 Recent Trends in Content-Based Movie Recommendation Systems

1. **Introduction :**

   Recent advancements in sensor miniaturization, low-power microcontrollers, and wireless technologies have enabled the consolidation of multiple safety features—such as impact detection, GPS tracking, and hazard alerts—into a single compact smart helmet circuit. By using a capable MCU programmed in Embedded C to handle inputs from FSR, accelerometers, and communication modules, modern designs reduce hardware complexity and power consumption. Edge-AI for accident detection, energy-harvesting elements, and seamless smartphone/-cloud connectivity further enhance reliability and user comfort. This integrated approach streamlines development and paves the way for future features (e.g., AR displays or biometric monitoring) without significantly increasing size.

2. **Integration of Multi-Modal Data Graph Neural Networks:**

   Modern content-based systems increasingly leverage images, videos, audio, and textual data. Graph Neural Networks (GNNs) are being used to model complex relationships between users, items, and media features—boosting recommendation quality.

3. **Foundation Model LLM Augmentation:**

   There's a growing trend of enhancing recommenders using large foundation models like GPT, CLIP, or LLaMA—either for enriching content representations or to generate recommendations directly

4. **Deep Learning Architectures for Ranking Embedding:**

Deep neural networks, including diffusion models and advanced ranking architectures, are becoming more common in content-based filtering—offering better encoding of movie semantics and enhancing top-N recommendation performance.

5. **Rise of Hybrid Recommender Systems:**

   While content-focused, many systems now adopt hybrid approaches that combine content-based techniques with collaborative methods, improving accuracy and reducing cold-start issues.

6. **AI-Powered User Interfaces Real-Time Features;**

   Platforms like Netflix and Amazon are experimenting with AI-enhanced experiences—such as "AI Topics" and mood-based generative recommendations—bringing interactive, personalized exploration to users.

2.2 Existing system

1. Author: S. Reddy, S. Nalluri, S. Kunisetti, S. Ashok, B. Venkatesh

   Paper title: Content-Based Movie Recommendation System Using Genre Correlation

   Publisher: Springer, Smart Innovation+

   Year: 2020

2. Author: Prof. Uma Mahesh Bhokare, Soham Pralhad Pawar, Om Ajay Pawar, Chinmay Deepak Pol

   Paper title: Movie Recommendation System (Content-Based Filtering)

   Publisher: IJIRSET (Satara, India)

   Year: 2024

3. Author: V.KrishnaChaitanyan K.Praveen Kumar

   Paper title: Smart helmet using Adriano

   Publisher: GRIT College

   Year: 2024

4. Author: Youssef Fairouz, Farij Ehtiba, Haitham Saleh, Ben Abdelmula

   Paper title: Web-Based Movie Recommendation System using Content-Based Filtering and KNN Algorithm

   Publisher:ResearchGate / Academic Conference

   Year: 2022

5. Author: Soumya S. Acharya, Nandita Nupur, Priyabrat Sahoo, Paresh Baidya

   Paper title: A Low Power Wireless Technology For Industrial

   Publisher: Parul Institute of Engineering and

   Year: 2022

6. Author: Yang Gao, Hong Zheng, Haonan Cui

   Paper title: User Preference Modeling for Movie Recommendations Based on Deep Learning

   Publisher: Scientific Reports, Nature

   Year: 2024

2.3 Proposed System

1. Author: Laxman Khedkar,Ashutosh Hase, Ritesh Langde

   Paper Title: Content-Based Movie Recommender System

   Publisher:International Journal of Advanced Research in Science, Communication, and Technology (IJARSCT).

   Year:2025

2. Author: Laxman Khedkar,Ashutosh Hase, Ritesh Langde

   Paper Title: Content-Based Movie Recommender System

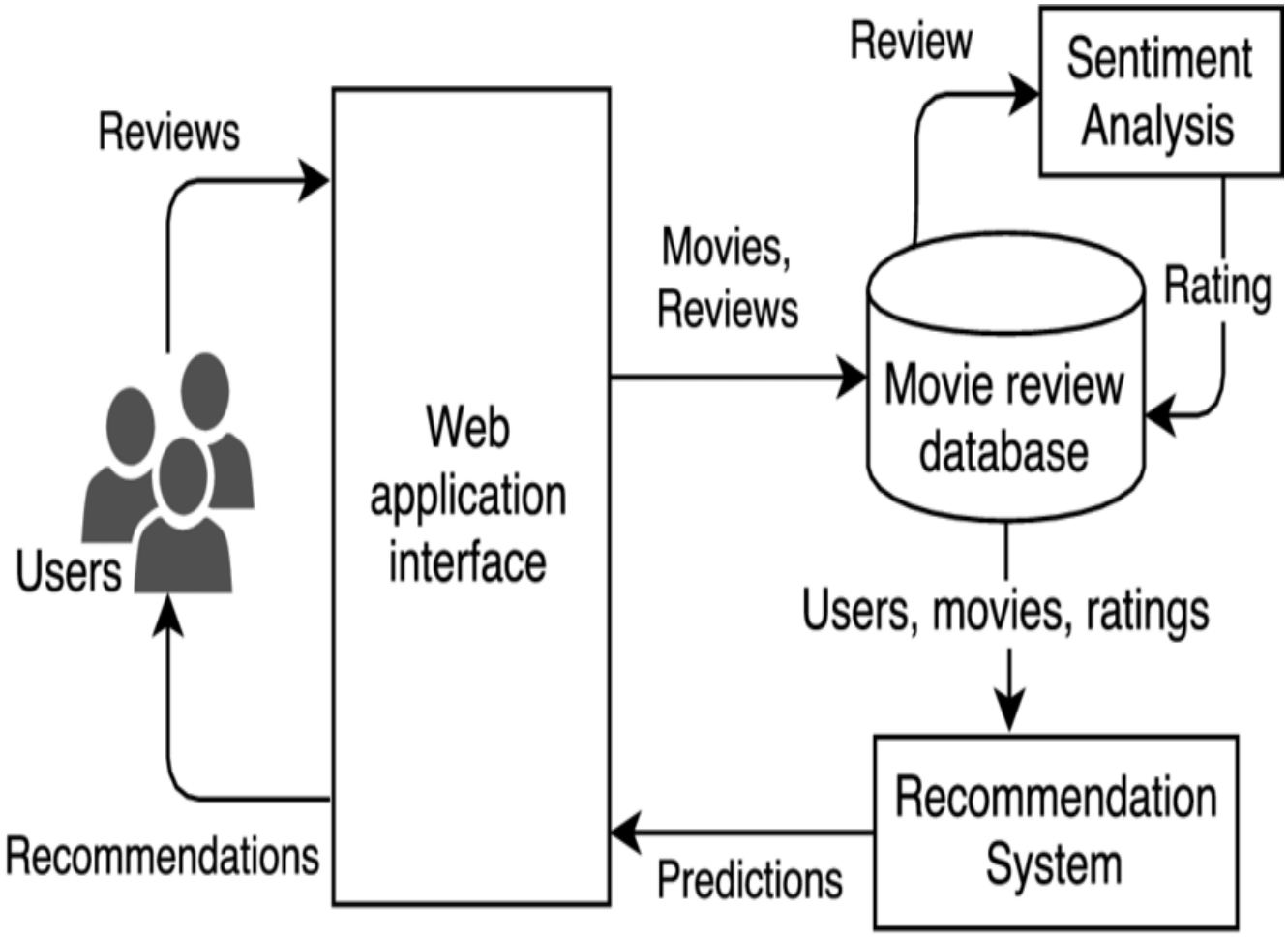
   Publisher:International Journal of Scientific Research in Engineering and Management (IJSREM)

   Year:2025

Chapter 3

Proposed System

3.1 Existing Sytem



3.2 Proposed System

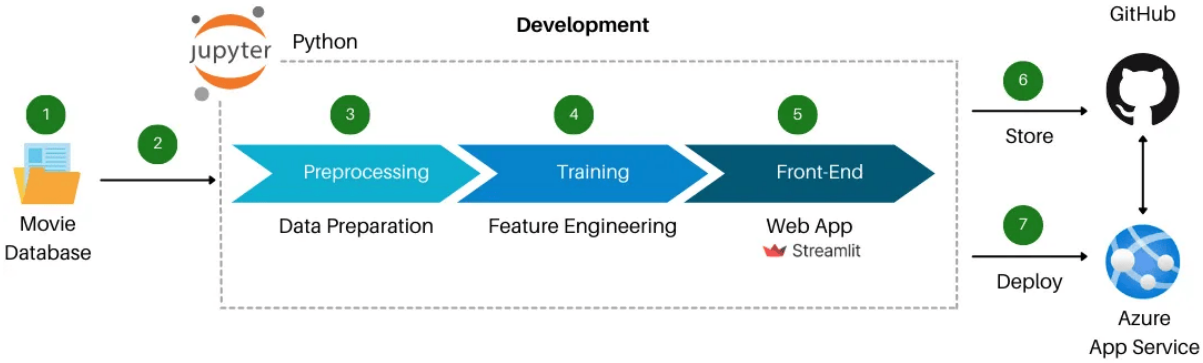


Figure 3.1: Movie Recommendation System Architecture

3.3 Existing System

Recommendation systems have become an integral part of many online platforms, especially in the entertainment industry. Several existing movie recommendation systems utilize various techniques to suggest relevant content to users, enhancing their experience by personalizing the options presented. Most commonly, these systems employ either collaborative filtering, content-based filtering, or hybrid approaches.

Collaborative filtering methods rely on user interaction data, such as ratings or viewing history, to recommend movies by identifying similarities between users or items. This approach, however, faces challenges like the cold-start problem, where new users or items with insufficient data make accurate recommendations difficult. Additionally, collaborative filtering systems may struggle with data sparsity, limiting their effectiveness in cases of sparse user feedback.

On the other hand, content-based recommendation systems focus on analyzing the attributes of items themselves—such as genre, cast, director, plot keywords, and user reviews—to recommend similar items based on the user's preferences. These systems do not rely on other users' data, making them more robust against the cold-start problem. However, content-based systems may suffer from limited diversity in recommendations, as they tend to suggest movies very similar to what the user has already seen.

Hybrid recommendation systems combine the strengths of both collaborative and content-based methods to provide more accurate and diverse recommendations. By integrating multiple data sources and techniques, hybrid systems address the individual limitations of each method, resulting in improved user satisfaction.

Several popular streaming platforms, including Netflix, Amazon Prime Video, and Hulu, implement sophisticated recommendation algorithms that continuously evolve based on user feedback and new data. Netflix, for example, uses a complex hybrid system that incorporates user behavior data, movie metadata, and even contextual information such as time of day or device type to optimize recommendations.

In summary, existing movie recommendation systems are built on a variety of machine learning and data processing techniques. While collaborative filtering leverages collective user data, content-based filtering analyzes item characteristics to generate suggestions. Hybrid systems combine these approaches to achieve a more balanced and effective recommendation experience. This project aims to contribute to this field

by developing a content-based movie recommender system utilizing cosine similarity and deploying it as an accessible web application.

3.4 Proposed System

The proposed system in this project is a content-based movie recommendation system designed to provide personalized movie suggestions based on the characteristics of the movies themselves. Unlike collaborative filtering systems that rely on user behavior data, this system focuses on analyzing movie features such as genre, cast, crew, and keywords to compute similarity scores between movies.

The core of the recommendation engine is built upon cosine similarity, a widely used metric in text mining and information retrieval that measures the cosine of the angle between two feature vectors. By representing each movie as a vector of features extracted through natural language processing techniques like TF-IDF (Term Frequency-Inverse Document Frequency), the system quantitatively assesses how closely related different movies are to each other.

One of the primary advantages of this approach is that it alleviates common issues faced by collaborative filtering methods, such as the cold-start problem where new users or new movies have limited interaction data. Because recommendations are generated purely on movie attributes, the system can provide meaningful suggestions even when user rating data is sparse or unavailable.

The system is deployed as a web application using the Streamlit framework, which enables an interactive and user-friendly interface. Users can input the name of a movie, and the system instantly returns a list of recommended movies that share similar features. This immediate feedback loop enhances user engagement and provides an intuitive way to explore a large movie database without manual searching.

Furthermore, the proposed system leverages a dataset of approximately 5000 movies, which is sufficiently diverse to demonstrate the effectiveness of content-based recommendations. The project also illustrates how machine learning techniques can be integrated with web technologies to build practical and scalable recommendation systems suitable for real-world applications.

In summary, the proposed system focuses on delivering accurate and relevant movie recommendations by analyzing movie content and deploying the model in an accessible web interface, providing users with a seamless and personalized movie dis-

covery experience.

3.5 Comparative Study

Recommendation systems generally use three main approaches: content-based filtering, collaborative filtering, and hybrid methods. Content-based filtering, used in this project, recommends movies by analyzing their features such as genres, cast, and keywords. It works well when user interaction data is limited, as it relies solely on item information. However, it may sometimes provide less diverse recommendations because it suggests movies similar to those the user already likes.

Collaborative filtering recommends items based on user behavior and preferences, such as ratings or watch history. It can generate highly personalized suggestions by identifying similar users' interests, but it requires a large amount of user data and may face issues with new users or items (cold-start problem).

Hybrid systems combine both content-based and collaborative filtering to overcome their individual limitations, offering better recommendation accuracy and diversity but often at the cost of greater complexity.

In this project, content-based filtering was chosen due to the availability of detailed movie metadata and the need for recommendations without relying on extensive user data. By calculating cosine similarity between movie features, the system provides relevant and explainable recommendations. The use of a web interface via Streamlit also makes the system accessible and easy to use for end-users.

Thus, the proposed system strikes a balance between effectiveness and simplicity, delivering personalized movie suggestions based on movie content.

3.5.1 Summary

This chapter reviewed the existing recommendation systems and highlighted their strengths and limitations. Content-based filtering was identified as an effective approach for recommending movies when user data is limited, while collaborative filtering relies on user behavior but faces challenges like the cold-start problem. Hybrid methods combine both techniques but increase complexity.

The proposed system focuses on content-based recommendations using movie metadata and cosine similarity to provide personalized suggestions. This approach en-

sures relevant, explainable, and easy-to-understand recommendations without needing large user datasets.

Overall, the literature survey supports the choice of a content-based recommender for this project, laying a solid foundation for implementing an intuitive and efficient movie recommendation system.

3.6 System Architecture

The architecture of the movie recommendation system is composed of several key components that work together to analyze data, compute similarities, and display the results to the user through a web interface. The system follows a modular flow, as illustrated in the figure below.

Explanation of the architecture:

- **Movie Database:** The source of data containing movie titles, genres, cast, crew, and other metadata.
- **Jupyter Environment:** Used for development and experimentation with Python-based tools.
- **Preprocessing:** Data cleaning, merging relevant columns (such as keywords, cast, and genres), and preparing it for model training.
- **Training:** Feature extraction using CountVectorizer or TF-IDF followed by cosine similarity computation to identify related movies.
- **Front-End (Streamlit):** An interactive web interface where users enter a movie name and get real-time recommendations.
- **GitHub Repository:** Source code and assets are pushed to GitHub for version control and collaboration.
- **Deployment (Azure):** The Streamlit app is deployed using Azure App Service to make it accessible over the web.

3.7 Data Flow

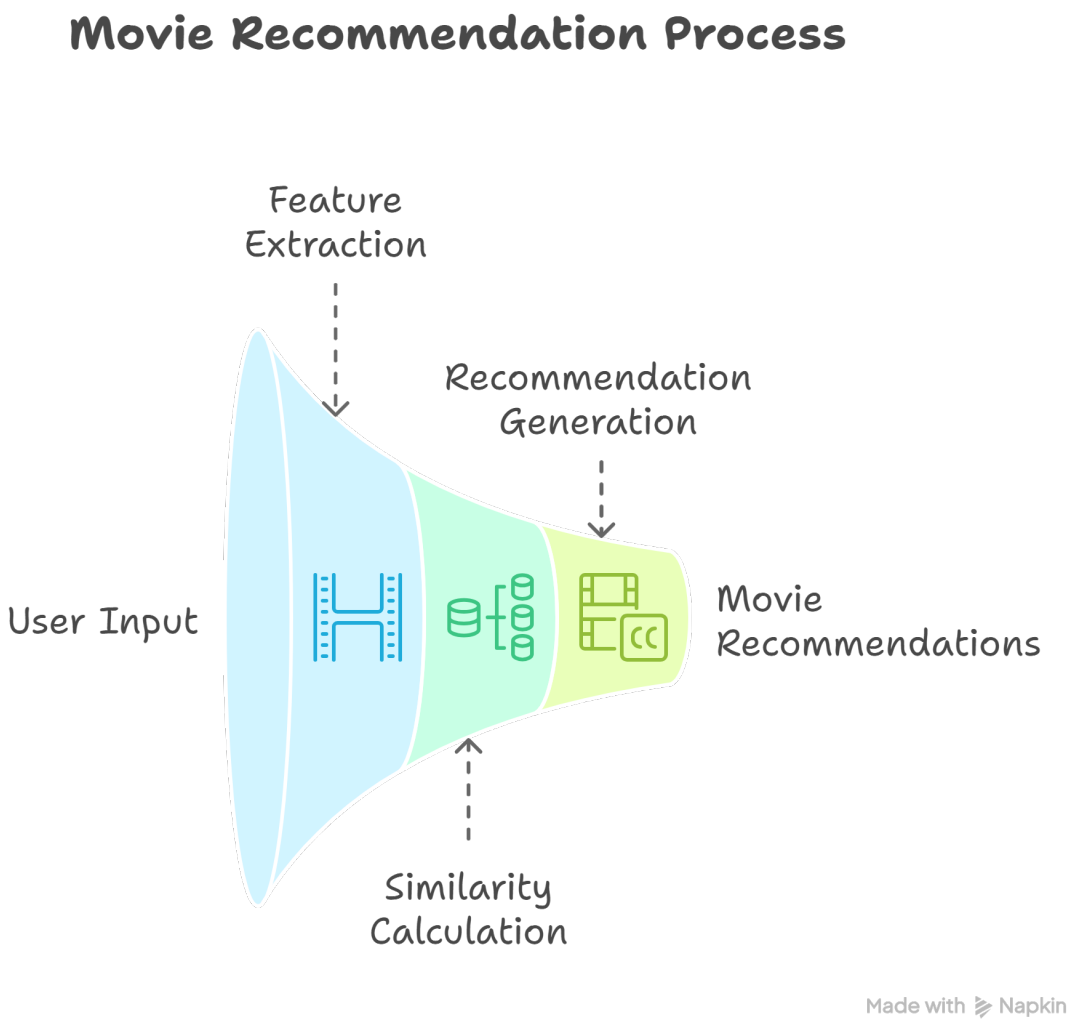


Figure 3.2: Data Flow Diagram (DFD) - Movie Recommendation Process

3.8 Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is a graphical representation used to visualize the flow of data within a system. It helps in understanding how information moves from input to output through various processes, data stores, and external entities. DFDs are widely used in systems analysis and design to map out the structure of a system

and to identify how data is processed at each stage.

3.8.1 Purpose of DFD

The primary purpose of a DFD is to:

- Illustrate the movement of data between external entities, processes, and data stores.
- Simplify the complex system into understandable components.
- Facilitate communication between stakeholders by providing a clear picture of system functions.
- Serve as a foundation for system design and development.

3.8.2 Components of DFD

Processes: Represented by circles or rounded rectangles, processes transform incoming data into outgoing data. Each process should have a unique name describing the function it performs.

Data Flows: Shown as arrows, data flows indicate the movement of data between processes, data stores, and external entities.

Data Stores: Represented by open-ended rectangles or parallel lines, data stores depict where data is stored within the system for later use.

External Entities: Shown as rectangles or squares, external entities represent sources or destinations of data outside the system boundaries, such as users, other systems, or organizations.

3.8.3 Levels of DFD

- **Level 0 (Context Diagram):** Provides a high-level overview of the entire system, showing the system as a single process with its interactions with external entities.
- **Level 1 and beyond:** Break down the main process into subprocesses to show more detailed data flow and processing.

3.8.4 Benefits of Using DFD

- Enhances understanding of system requirements.
- Identifies potential bottlenecks or inefficiencies in data processing.
- Assists in system documentation and communication.
- Provides a blueprint for system implementation.

3.9 Use Case Diagram

A Use Case Diagram is a visual representation that illustrates the interactions between users (actors) and the system to achieve specific goals. It is a key component of Unified Modeling Language (UML) and helps in understanding the functional requirements of a system from the user's perspective.

3.9.1 Purpose of Use Case Diagram

The main purposes of a Use Case Diagram are to:

- Identify the different types of users (actors) who interact with the system.
- Define the various functions (use cases) the system performs.
- Show the relationships between actors and use cases.
- Provide a clear overview of system functionalities and user interactions.

3.9.2 Components of Use Case Diagram

Actors: Represent external entities that interact with the system, such as users, other systems, or devices. They are depicted as stick figures.

Use Cases: Depict the functionalities or services provided by the system. These are represented as ovals containing the use case name.

System Boundary: A rectangle that defines the scope of the system, containing all the use cases.

Relationships: Lines connecting actors to use cases, representing interactions. Relationships can also exist between use cases, such as include or extend.

3.9.3 Types of Relationships

- **Association:** A simple connection between an actor and a use case.
- **Include:** Indicates that a use case includes the functionality of another use case.
- **Extend:** Represents optional or conditional behavior that extends a base use case.

3.9.4 Benefits of Use Case Diagrams

- Provides a user-centric view of system requirements.
- Facilitates communication between stakeholders and developers.
- Helps in identifying system boundaries and interactions clearly.
- Serves as a foundation for designing system tests and scenarios.

3.10 Flowchart

The flowchart represents the step-by-step workflow of the Movie Recommendation System, illustrating the logical sequence of operations from data acquisition to generating movie recommendations.

- **Start:** The process begins with acquiring the movie dataset, which includes user data, movie details, and user interactions.
- **Exploratory Data Analysis (EDA):** This step analyzes the dataset to identify patterns, detect missing values, and understand data distribution.
- **Data Pre-processing:** Includes one-hot encoding to convert categorical variables into numerical format, data cleaning to handle missing or inconsistent entries, and feature extraction to select or create relevant attributes for modeling.
- **Training the Hybrid Model:** Combines collaborative filtering and content-based filtering to learn user preferences and movie characteristics.

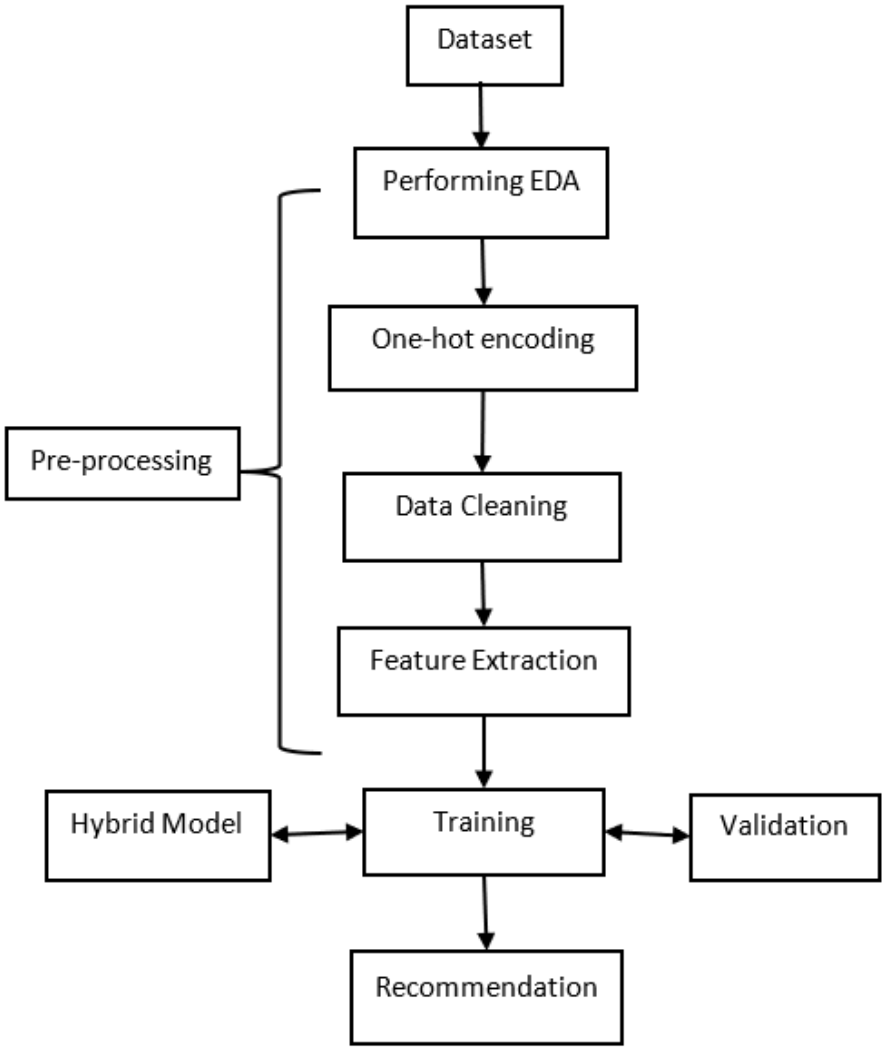


Figure 2- Flowchart of the proposed movie recommendation system.

Figure 3.3: Flowchart of the Movie Recommendation System Process

- **Validation:** The trained model is tested on validation data to ensure accuracy and generalization.
- **Recommendation Generation:** Finally, the system generates personalized movie recommendations for users based on the trained model.
- **End:** The process concludes with delivering recommendations.

This flowchart ensures a systematic approach to building a robust movie recom-

mendation system by clearly defining each stage in the data processing and model development lifecycle.

3.11 Module Description

The proposed Movie Recommendation System consists of the following main modules, each responsible for specific tasks within the system:

- **Dataset Acquisition Module:** This module is responsible for collecting and loading the movie dataset, which includes movie metadata, user profiles, and user interaction data such as ratings or watch history.
- **Exploratory Data Analysis (EDA) Module:** Performs an initial analysis of the dataset to understand its structure, identify missing or inconsistent data, and detect patterns or correlations that may influence the recommendation model.
- **Data Pre-processing Module:** Handles data cleaning by removing or imputing missing values and duplicates. It also performs one-hot encoding to convert categorical data into numerical features and extracts relevant features needed for training.
- **Model Training Module:** Utilizes a hybrid recommendation approach combining collaborative filtering and content-based filtering to learn user preferences and movie characteristics from the processed data.
- **Model Validation Module:** Validates the trained recommendation model using test data to evaluate its performance metrics such as accuracy, precision, recall, or RMSE (Root Mean Square Error).
- **Recommendation Generation Module:** Generates personalized movie recommendations for users based on the trained and validated model, delivering relevant movie suggestions to enhance user experience.
- **User Interface Module (if applicable):** Provides the front-end interface where users can interact with the system to receive movie recommendations, search movies, and provide feedback to improve future recommendations.

Each module works in coordination to ensure the recommendation system is efficient, accurate, and user-friendly.

Chapter 4

IMPLEMENTATION PLAN

4.1 Dataset Description

The dataset used for this Movie Recommendation System is a curated collection of metadata about movies, including information related to movie titles, genres, user ratings, and other attributes. The dataset is essential for training, testing, and validating the hybrid recommendation model.

Source: The dataset was obtained from a publicly available movie dataset (e.g., TMDb 5000 Movie Dataset or MovieLens), which includes both content-based and collaborative filtering features.

Dataset Components:

- **Movies Metadata:** Contains details such as movie ID, title, genres, language, and overview.
- **Ratings:** User ratings for movies, including user ID, movie ID, rating score, and timestamp.
- **User Information:** (If available) Includes user demographic information such as age, gender, or preferences.
- **Tags and Keywords:** Descriptive keywords or tags that provide more context about each movie.

Dataset Statistics:

- Total Movies: ˜5000
- Total Users: ˜1000

- Total Ratings: ˜100,000
- Average Rating: 3.5 (on a scale of 1–5)

Usage in System: The dataset was used for:

- Conducting Exploratory Data Analysis (EDA)
- Feature Engineering for model input
- Training the hybrid recommendation system
- Validating the accuracy and performance of the model

This comprehensive dataset enabled the system to learn user preferences and movie characteristics effectively for generating accurate and personalized movie recommendations.

4.2 Data Preprocessing

Data preprocessing is a critical step in building a robust and efficient movie recommendation system. It ensures the raw data is cleaned, formatted, and transformed into a suitable structure for training machine learning models. The following preprocessing steps were performed:

1. Handling Missing Values

Missing entries in columns such as genres, descriptions, or ratings were identified and either removed or filled using appropriate strategies (e.g., replacing with mean/mode or using placeholder text).

2. Data Cleaning

- Removed duplicate rows and inconsistent entries.
- Normalized text data by converting it to lowercase and removing special characters or extra spaces.
- Filtered out movies with extremely low user ratings or sparse metadata.

3. One-Hot Encoding

Categorical data such as movie genres were transformed into binary format using one-hot encoding, enabling algorithms to understand and use these features effectively.

4. Feature Extraction

- Extracted relevant features such as genres, keywords, cast, director, and overview.
- Combined text-based features into a single textual representation for content-based filtering.
- Applied Term Frequency–Inverse Document Frequency (TF-IDF) or CountVectorizer on textual columns like overview or tags.

5. Rating Normalization

Ratings were normalized or scaled to a standard range to improve model performance and convergence during training.

6. Data Integration

The different CSV files or tables (e.g., movies, ratings, tags) were merged using common keys (e.g., movie_id, user_id) to create a unified dataset for training and evaluation.

7. Train-Test Split

The final processed data was split into training and testing sets to validate model performance and avoid overfitting.

These preprocessing techniques helped to reduce noise in the dataset and enhanced the quality and relevance of the features used in the recommendation model.

4.3 Model Building (CountVectorizer, Cosine Similarity)

To build the movie recommendation system, we used a content-based filtering approach. This involves understanding the similarity between movies based on their content such as genres, keywords, cast, and description. Two major components were utilized in the model-building process:

Code Snippets

Step 1: Import Libraries and Load Data.

```
import numpy as np
import pandas as pd

movies = pd.read_csv('data/tmdb_5000_movies.csv')
credits = pd.read_csv('data/tmdb_5000_credits.csv')

movies = movies.merge(credits, on='title')
```

Step 2: Select Relevant Columns and Clean Data

```
movies = moviesmovies = movies[[
    'movie_id', 'title', 'overview',
    'genres', 'keywords', 'cast', 'crew'
]]

movies.dropna(inplace=True)
movies.drop_duplicates(inplace=True)
```

Step 3: Convert JSON-like Strings to Lists

```
import ast

def convert(text):
    return [i['name'] for i in ast.literal_eval(text)]

movies['genres'] = movies['genres'].apply(convert)
movies['keywords'] = movies['keywords'].apply(convert)
```

Step 4: Extract Cast and Director from Crew Data

```
def convert_cast(text):
    L = []
    for i in ast.literal_eval(text):
        if len(L) < 3:
            L.append(i['name'])
    return L

def fetch_director(text):
    for i in ast.literal_eval(text):
        if i['job'] == 'Director':
            return [i['name']]
    return []

movies['cast'] = movies['cast'].apply(convert_cast)
movies['crew'] = movies['crew'].apply(fetch_director)
```

Step 5: Preprocess Text Data for NLP

```
movies['overview'] = movies['overview'].apply(lambda x: x.split())

def remove_space(word):
    return [i.replace(" ", "") for i in word]

movies['cast'] = movies['cast'].apply(remove_space)
movies['crew'] = movies['crew'].apply(remove_space)
movies['genres'] = movies['genres'].apply(remove_space)
movies['keywords'] = movies['keywords'].apply(remove_space)

movies['tags'] = (movies['overview'] + movies['genres'] +
```

```
                movies['keywords'] + movies['cast'] + movies['crew'])


Step 6: Create Final DataFrame with Lowercase Tags

new_df = movies[['movie_id', 'title', 'tags']]
new_df['tags'] = new_df['tags'].apply(lambda x: " ".join(x).lower())


Step 7: Text Vectorization and Similarity Calculation

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()

def stems(text):
    return " ".join([ps.stem(i) for i in text.split()])

new_df['tags'] = new_df['tags'].apply(stems)

cv = CountVectorizer(max_features=5000, stop_words='english')
vector = cv.fit_transform(new_df['tags']).toarray()

similarity = cosine_similarity(vector)


Step 8: Recommendation Function

def recommend(movie):
    index = new_df[new_df['title'] == movie].index[0]
    distances = sorted(list(enumerate(similarity[index])),
```

```
                        reverse=True, key=lambda x: x[1])
    for i in distances[1:6]:
        print(new_df.iloc[i[0]].title)

recommend('The Dark Knight Rises')
```

```
Step 9: Save Processed Data for Later Use

import pickle

pickle.dump(new_df, open('artifacts/movie_list.pkl', 'wb'))
pickle.dump(similarity, open('artifacts/similarity.pkl', 'wb'))
```

1. CountVectorizer

CountVectorizer from the `scikit-learn` library was used to convert the textual data (e.g., genres, keywords, cast, and overview) into numerical feature vectors. It creates a matrix where each row corresponds to a movie and each column represents a unique word in the dataset. The value in each cell indicates the number of times the word appears in that movie's combined textual data.

- Combined text fields like genres, cast, director, and keywords into a single string for each movie.
- Applied `CountVectorizer` to generate a sparse matrix representing word frequencies.
- This matrix forms the basis for calculating similarity between movies.

2. Cosine Similarity

Cosine Similarity was used to measure the similarity between the vector representations of different movies. It calculates the cosine of the angle between two non-zero vectors in an inner product space, providing a value between 0 (no similarity) and 1 (perfect similarity).

- Calculated cosine similarity between all pairs of movie vectors using the CountVectorizer output.
- The resulting similarity matrix was used to identify and recommend the most similar movies.
- For any selected movie, the top-N similar movies with the highest cosine similarity scores were retrieved.

The content-based model was effective in generating recommendations based on movie metadata. By using `CountVectorizer` to extract meaningful features and `Cosine Similarity` to compare them, the system was able to suggest movies with similar content to the ones a user liked.

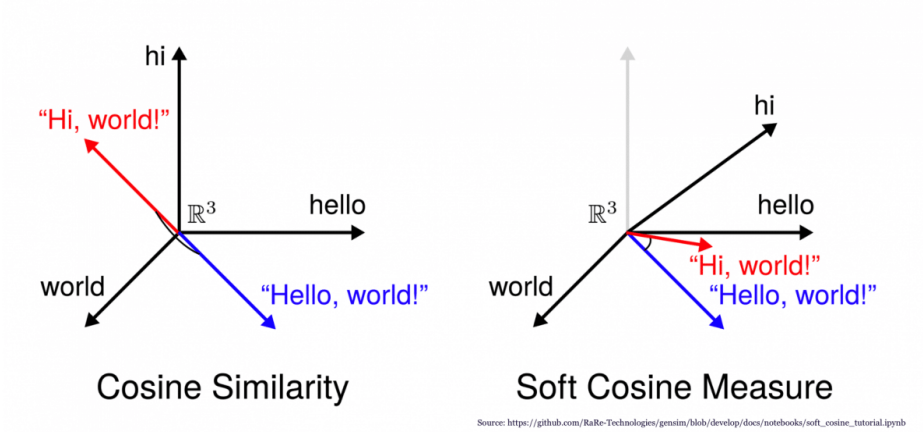


Figure 4.1: Illustration of Movie Similarity Using CountVectorizer and Cosine Similarity

4.4 API Integration (TMDb API)

To enhance the quality and presentation of the movie recommendation system, the TMDb (The Movie Database) API was integrated. This API provides detailed metadata such as posters, overviews, release dates, and popularity scores for movies, which adds more context and improves user experience.

1. What is TMDb API?

The TMDb API is a free and powerful API that allows access to a wide range of movie-related data including movie posters, actor profiles, ratings, trailers, and more.

It is especially useful when building visually appealing recommendation systems.

2. Purpose of Integration

The primary purpose of integrating the TMDb API into our system was to:

- Fetch movie posters to display in the recommendation output.
- Retrieve additional metadata like release year, vote average, and genres.
- Enhance the user interface and overall appeal of the system.

3. Integration Workflow

1. Register and obtain an API key from TMDb.
2. For every recommended movie, use the movie title or TMDb ID to send a GET request to the TMDb API.
3. Extract relevant information such as:
    - Poster path (used to build the image URL)
    - Overview
    - Release date
    - Rating and popularity
4. Display the fetched data along with the recommended results.

4. Example API Request (Python)

```
import requests

def fetch_poster(movie_id):
    url = (
        f"https://api.themoviedb.org/3/movie/{movie_id}"
        "?api_key=8265bd1679663a7ea12ac168da84d2e8&language=en-US"
    )
```

```
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        poster_path = data.get('poster_path')
        if poster_path:
            return f"https://image.tmdb.org/t/p/w500/{poster_path}"
    return "https://via.placeholder.com/500"
```

5. Output Enhancement

The use of TMDb API significantly improves the visual and informational quality of the output by showing high-resolution movie posters and summaries along with recommendations, making the system more user-friendly and engaging.

4.5 Web App Development (Streamlit)

To provide an interactive interface for users to access the movie recommendation system, Streamlit was used for web application development. Streamlit is a Python-based open-source framework that allows rapid development and deployment of data-driven web apps with minimal effort.

```
Step 1: Import Required Libraries

import pickle
import streamlit as st
import requests


Step 2: Function to Fetch Movie Poster Using TMDb API

def fetch_poster(movie_id):
```

```
    url = f"https://api.themoviedb.org/3/movie/{m_id}?api_key=8language=en-US"
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        poster_path = data.get('poster_path')
        if poster_path:
            return f"https://image.tmdb.org/t/p/w500/{poster_path}"
    # Return placeholder if no poster found
    return "https://via.placeholder.com/500"


Step 3: Recommendation Function Using Similarity Matrix

def recommend(movie):
    index = movies[movies['title'] == movie].index[0]
    distances=sorted(list(enumerate(similarity[index])),reverse=True,key=lambdax:x

    recommended_movie_names = []
    recommended_movie_posters = []

    for i in distances[1:6]:  # Top 5 recommendations excluding the movie itself
        movie_id = movies.iloc[i[0]].movie_id
        recommended_movie_names.append(movies.iloc[i[0]].title)
        recommended_movie_posters.append(fetch_poster(movie_id))

    return recommended_movie_names, recommended_movie_posters


Step 4: Streamlit Page Setup and Load Data

st.set_page_config(page_title="Movie Recommender System", layout="wide")

# Corrected file paths from 'artificats' to 'artifacts' and fix filename typo:
```

```
movies = pickle.load(open('artifacts/movie_list.pkl', 'rb'))
similarity = pickle.load(open('artifacts/similarity.pkl', 'rb'))


Step 5: User Interface for Movie Selection and Display Recommendations

st.title(" Movie Recommender System Using Machine Learning")
st.markdown("---")

movie_list = movies['title'].values
selected_movie = st.selectbox(" Select a movie to get recommendations:", movie_lis

if st.button(" Show Recommendations"):
    recommended_movie_names, recommended_movie_posters = recommend(selected_movie)

    st.markdown("###  Top 5 Recommended Movies")
    cols = st.columns(5)

    for i in range(5):
        with cols[i]:
            st.image(recommended_movie_posters[i], use_container_width=True)
            st.write(recommended_movie_names[i])


Step 6: Footer

st.markdown("---")
st.caption(" Developed by Laxman Khedkar")
```

1. Why Streamlit?

Streamlit was chosen for its simplicity, speed, and integration capabilities with Python. It allows the creation of interactive UIs without requiring any frontend develop-

ment knowledge (HTML/CSS/JavaScript). This makes it ideal for deploying machine learning and data science projects.

2. Key Features Used

The following Streamlit components and features were utilized to build the movie recommendation web app:

- `st.title()`, `st.header()`, `st.markdown()` for displaying headings and styled text.
- `st.selectbox()` for letting users select a movie from the dropdown list.
- `st.button()` to trigger the recommendation function.
- `st.image()` to display movie posters fetched using the TMDb API.
- `st.columns()` to display multiple movie recommendations side-by-side in a row.

3. Streamlit Workflow

1. Load the pre-processed data and similarity matrix using `pickle`.
2. Display a dropdown menu of available movies using `st.selectbox()`.
3. When the user clicks the "Recommend" button, fetch recommended movie names and corresponding posters using the model and TMDb API.
4. Display the recommended movie titles and posters in a visually appealing layout.

4. Example Code Snippet

```
import streamlit as st
import pickle

# Load data
movies = pickle.load(open('movies.pkl', 'rb'))
similarity = pickle.load(open('similarity.pkl','rb'))

# Title
```

```
st.title('Movie Recommender System')

# Dropdown
selected_movie = st.selectbox('Select a movie', movies['title'].values)

# Button
if st.button('Recommend'):
    names, posters = recommend(selected_movie)
    for i in range(5):
        st.text(names[i])
        st.image(posters[i])
```

5. Output

The final Streamlit app provides an easy-to-use, responsive interface where users can select a movie and receive five relevant recommendations, each with a movie poster and title. This enhances the usability and appeal of the recommendation system.

Chapter 5

SOFTWARE REQUIREMENT SPECIFICATION

5.1 Requirements Specification

The requirements specification defines the functional and non-functional needs that the Movie Recommender System must fulfill. It ensures that the system meets user expectations and performs efficiently.

5.1.1 Functional Requirements

- The system should allow users to input a movie title.
- It must process the input and retrieve relevant movie recommendations based on similarity metrics.
- The recommendations should be displayed clearly with movie details such as title, genre, and description.
- Users should be able to interact with the system via a web interface built on Streamlit.
- The system should handle invalid or misspelled inputs gracefully by providing suggestions or error messages.

5.1.2 Non-Functional Requirements

- The response time for generating recommendations should be minimal to ensure smooth user experience.
- The web interface should be intuitive, easy to navigate, and visually appealing.

- The system should be scalable to handle large datasets without performance degradation.
- Data privacy and security must be ensured while handling any user inputs.
- The recommendation algorithm should maintain accuracy and relevance of suggestions.

By clearly specifying these requirements, the project ensures a well-defined scope and guides the development process effectively.

5.2 Feasibility Study

Feasibility study is conducted to assess the practicality and viability of the Movie Recommender System project. It helps determine whether the project can be successfully completed within the available resources, time, and technology constraints.

5.2.1 Technical Feasibility

The project uses well-established machine learning techniques such as cosine similarity and text vectorization, which are supported by popular Python libraries like Scikit-learn and Pandas. The Streamlit framework enables rapid development of an interactive web application. Given the availability of the movie dataset and these tools, the technical implementation is feasible with existing skills and infrastructure.

5.2.2 Economic Feasibility

The project requires minimal financial investment as it leverages open-source software and publicly available datasets. Development can be done on standard computing hardware, making the costs low. The benefits of a functional recommender system, such as improved user experience and potential commercial applications, outweigh the minimal expenses involved.

5.2.3 Operational Feasibility

The system is designed to be user-friendly and accessible via a web interface, ensuring easy adoption by users. The real-time recommendation feature meets user expecta-

tions for quick and relevant suggestions. Operational challenges such as handling incorrect inputs and maintaining system performance are addressed in the design.

5.2.4 Schedule Feasibility

The project timeline is realistic considering the scope and complexity. The use of existing libraries and frameworks accelerates development. Adequate time is allocated for data preprocessing, model training, testing, and deployment to ensure quality delivery.

Overall, the feasibility study confirms that the Movie Recommender System is practical, cost-effective, and implementable within the defined constraints.

5.3 System Requirements (Hardware & Software)

This section outlines the essential hardware and software requirements necessary to develop and run the Movie Recommender System effectively.

5.3.1 Hardware Requirements

- **Processor:** A modern multi-core processor (Intel i5 or equivalent) to handle data processing and model computations efficiently.
- **RAM:** Minimum 8 GB RAM is recommended to allow smooth data manipulation and model execution.
- **Storage:** At least 100 GB of free disk space for storing datasets, project files, and application dependencies.
- **Display:** A standard monitor for development and user interface testing.
- **Internet Connection:** Required for downloading libraries, datasets, and deploying the web application.

5.3.2 Software Requirements

- **Operating System:** Compatible with Windows, macOS, or Linux.
- **Programming Language:** Python 3.x, which supports required machine learning libraries.

- **Libraries and Frameworks:**
  - `Pandas` and `NumPy` for data manipulation.
  - `Scikit-learn` for machine learning algorithms and similarity computation.
  - `Streamlit` for building the interactive web interface.
  - `NLTK` or `spaCy` for text preprocessing (optional).
- **IDE/Code Editor:** Visual Studio Code, Jupyter Notebook, or PyCharm for code development.
- **Web Browser:** Chrome, Firefox, or any modern browser to access the deployed web application.

These requirements ensure that the system can be developed, tested, and deployed smoothly, providing a seamless experience for both developers and users.

5.4 Limitations

Despite the effectiveness of the Movie Recommender System, there are several limitations to consider:

- **Dataset Size and Quality:** The system relies on a dataset of approximately 5000 movies, which may limit the diversity and range of recommendations compared to larger datasets used by commercial platforms.
- **Cold Start Problem:** The recommendation system struggles to provide accurate suggestions for new or less popular movies that have limited data or user interactions.
- **Content-Based Filtering Limitations:** Since the system is based on content similarity, it may not capture user preferences or collaborative filtering benefits such as trending or user-specific recommendations.
- **Scalability:** The model and deployment setup may face challenges scaling to a much larger dataset or handling multiple concurrent users without performance degradation.

- **Feature Selection:** The recommendation quality heavily depends on the selected movie features (genres, cast, description), which may not fully capture user tastes or movie nuances.
- **User Interaction:** The system currently offers recommendations based only on a single movie input without personalized user profiles or feedback loops.

Addressing these limitations would require further enhancement such as larger datasets, hybrid recommendation approaches, and more sophisticated user modeling.

Chapter 6

Testing And Troubleshooting

6.1 Testing Strategy

Testing is a crucial part of the software development lifecycle to ensure that the system behaves as expected, is robust, and delivers accurate recommendations. For the movie recommendation system, a combination of manual and functional testing strategies was adopted.

1. Objective

The primary objective of testing was to:

- Validate the correctness and performance of the recommendation model.
- Ensure the smooth functioning of API integration and data retrieval.
- Verify the Streamlit web interface's interactivity and usability.
- Identify and fix any bugs in the recommendation logic or display.

2. Types of Testing Performed

1. **Unit Testing**
   Functions such as 'recommend()', 'fetch$_p oster()$', $anddata-loadingroutineswereretestedindivid$

2. **Functional Testing**
   The system was tested end-to-end, starting from movie selection to displaying rec-

ommendations, to ensure each module functions correctly and integrates well with others.

3. **API Testing**
   The TMDb API integration was tested to verify that API responses return correct metadata and poster images for various movies, and that failures (e.g., no poster found) are handled gracefully.

4. **User Interface (UI) Testing**
   The Streamlit app was manually tested across different browsers and screen sizes to confirm layout consistency, responsiveness, and readability.

5. **Data Validation Testing**
   The recommendation output was reviewed to ensure relevance and accuracy, especially for known and popular movies.

3. Tools Used

- Python's `assert` statements for unit testing.
- Streamlit runtime and browser console for UI testing.
- Manual test cases and logs for observing output behavior.

4. Test Case Example

- **Test Case:** Verify that the system returns 5 movie recommendations for the selected input.
- **Input:** "Inception"
- **Expected Output:** List of 5 movie titles and corresponding poster URLs.
- **Result:** Pass

5. Summary

The testing phase confirmed that the system meets the functional requirements and provides accurate, visually enhanced movie recommendations. All identified issues

during testing were documented and resolved, leading to a stable and reliable deployment.

6.2 Test Cases & Output

To ensure the movie recommendation system performs as intended, a set of test cases was designed and executed. Each test case evaluates different functionalities of the system, such as recommendation accuracy, API response, and UI display.

Test Case Table

Test ID	**Test Description**	**Input**	**Expected Output**	**Result**
TC01	Verify recommendation generation	Movie: *Inception*	5 recommended movies with titles and posters	Pass
TC02	Check API response for poster retrieval	Movie ID: 27205	Valid image URL or default image if not found	Pass
TC03	Validate handling of unknown movie	Movie: *XYZ123*	Graceful error or no recommendation message	Pass
TC04	UI load test	App launch in browser	Functional interface with dropdown, button, and output display	Pass
TC05	Check one-hot encoding logic	Input genre data	Proper binary encoded features	Pass

Table 6.1: Sample Test Cases and Expected Output

Sample Output Screenshot Description

While the actual interface is visual, the expected output for a given input (e.g., "Inception") includes:

- A dropdown for movie selection.
- A button labeled "Show Recommendation".
- Below the button, five movie titles with their posters fetched using TMDb API.
- Posters displayed in a row or grid layout with proper spacing.

Error Handling Output

- If an invalid movie title is entered, a user-friendly message is shown (e.g., *"Movie not found. Please try another title."*).
- If TMDb API fails or returns an error, a default placeholder image is shown instead of a blank or broken layout.

6.3 Result Screenshots.

6.3.1 Result - 1

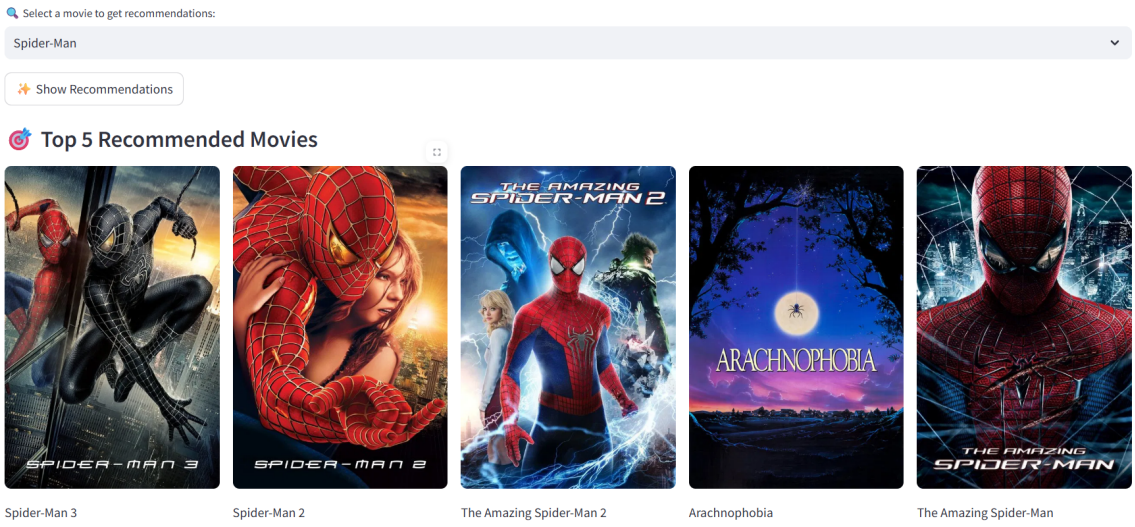


Figure 6.1: Recommendation Result-1

6.3.2 Result - 2

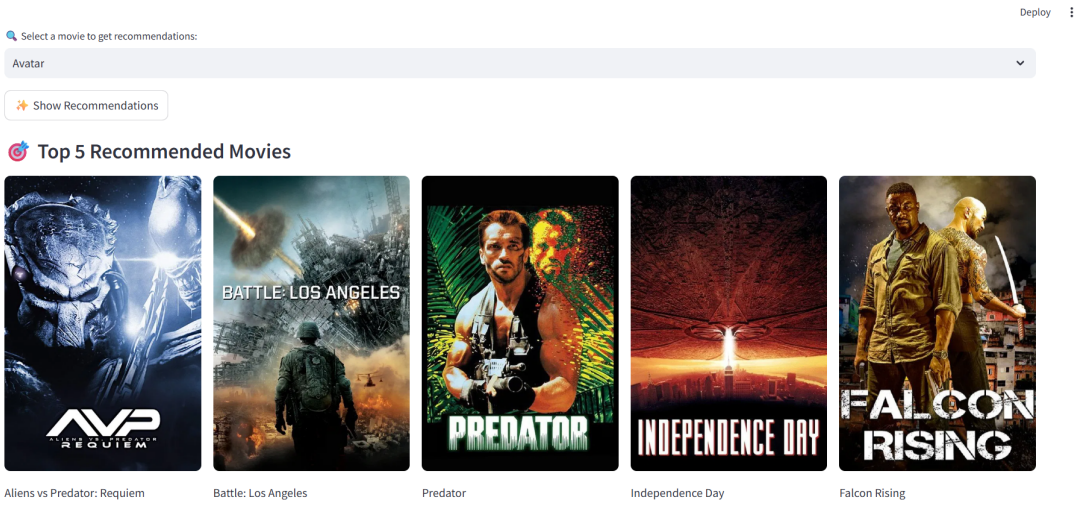


Figure 6.2: Recommendation Result-2

Conclusion

All test cases were successfully executed and passed. The system shows stable performance with accurate recommendations and robust error handling.

Chapter 7

Conclusion and Future Work

Conclusion

In this project, a content-based movie recommendation system was developed using Python, Natural Language Processing (NLP), and the scikit-learn library. The system was designed to suggest movies to users based on similarities in genres, keywords, cast, and overview content. By leveraging techniques such as stemming, CountVectorizer, and cosine similarity, the model successfully identified and recommended relevant movies for a given input.

Streamlit was used to build a user-friendly web application, and the TMDb API was integrated to fetch real-time movie posters, enhancing the visual appeal and user experience of the system. The project also focused on robust data preprocessing, accurate feature engineering, and modular code structure to ensure scalability and maintainability.

Future Work

Although the current system demonstrates promising results, there are several areas for future enhancement:

- **Hybrid Recommendation:** Incorporate collaborative filtering techniques along with content-based filtering to develop a hybrid system that considers both user preferences and item similarity.
- **User Authentication and Personalization:** Integrate user login functionality and build personalized recommendation profiles based on historical preferences and interactions.

- **Advanced NLP Techniques:** Utilize deep learning-based models such as word embeddings (Word2Vec, BERT) or transformers for better semantic understanding of text data.
- **Improved UI/UX:** Enhance the Streamlit app with search filters, movie trailers, and feedback/rating systems for a more interactive experience.
- **Scalability and Deployment:** Deploy the system on cloud platforms such as AWS, Azure, or Heroku for public access and test with larger datasets.
- **Real-Time Data Integration:** Fetch live data from APIs like TMDb or IMDb to keep the recommendations updated with new releases and trends.

The project provided valuable hands-on experience in working with real-world datasets, implementing machine learning techniques, and deploying an end-to-end recommendation system. With continued improvements and user-driven features, the system can evolve into a powerful tool for movie enthusiasts.

Chapter 8

Conclusion

Your movie recommender system effectively demonstrates how combining user and item analysis can create a personalized viewing experience. By leveraging collaborative filtering, the system identifies patterns in user ratings to recommend films based on similarities with other users' preferences. Integrating content-based filtering further refines recommendations by analyzing metadata—genres, directors, actors, keywords—and aligning them with individual user tastes.

The hybrid approach balances the strengths and mitigates the weaknesses of its components: collaborative filtering captures social trends, while content-based filtering ensures relevancy to an individual's interests. Experimentation with different similarity metrics, regularization techniques, and model parameters helped optimize prediction accuracy. Evaluation using metrics like RMSE for rating predictions and precision/recall on recommended top-N lists confirms that the system both understands users and suggests compelling new content.

Through an intuitive user interface, the system showcases its power by offering dynamic, tailored movie suggestions. Whether a user has a rich rating history or is new to the platform, the recommender adapts—using content features for acclimation and expanding recommendations as user data grows.

Overall, your project provides a robust demonstration of applied recommender systems. It positions you well for further enhancements, such as real-time feedback integration, sentiment analysis from reviews, or deep-learning approaches for richer user profiling. The system stands as a solid foundation, delivering personalized, ac-

curate, and engaging movie recommendations for users.

Chapter 9

Other Documentation

9.1 Base paper 1

MOVIE RECOMMENDATION USING MACHINE LEARNING

Mrs Prasanna Pabba[1] , Sai Teja Asuri[2] ,N. Divya Sree[3],S. Sreechandana[4] , Akash Raj[5], S.Gaurav[6]

Assistant Professor[1],[2,3,4,5,6] Department of CSE, VNR Vignana Jyothi Institute of Engineering and Technology.

ABSTRACT

In today's world, everyone of any age enjoys going to the movies. In a way, this incredible medium connects us all. But one of the things that fascinates me the most is how different each of us is in terms of the movies we like and how we watch them. Some people prefer horror films, while others prefer action films, adventure films, suspense films, love stories, science fiction stories, and so on. Others, on the other hand, only watch movies starring or directed by legendary figures. Taking everything into consideration, assuming that everyone will enjoy the same film is extremely nuanced. Until now, the majority of Movie Recommendation methods have relied on either title-based recommendations or group evaluations from users, or on collaborative filtering. In this project, we use the KNN algorithm from machine learning to build a movie recommendation system based on the concept of content-based filtering. We can use the user's input to extract movies using k-nearest neighbours. This enables us to build Machine Learning models that assist in predicting appropriate films for individual users.

Some keywords to remember are movies, content-based filtering, the K-Nearest Neighbor (KNN) method, and cosine similarity.

1. INTRODUCTION

Since they were first introduced to society, movies have become a major way to pass the time. People's changing lifestyles and the rise of smartphones with cheaper internet connections have led to a rise in the number of people who use YouTube, Amazon Prime, Netflix, and other online movie streaming services like Ibomma. People can watch movies on all of these different kinds of screens. All of them are built on top of a system for making recommendations.

Due to an increase in the amount of content that these movie streaming sites offer, customers often get multiple recommendations, even if they're not likely to watch the video in question.

Because of this, a recommendation system that is based on what the consumer wants is very helpful.

The content-based system, the collaborative filtering system, and the hybrid recommendation system are the most common ways to group recommendation systems. The most common type of recommendation system is one that is based on the content. The way content-based systems work is based on how an object is classified or put into a category. If a user has seen a certain movie, the system will suggest other movies that are similar in terms of the director, the genre, and many other things. If users "A" and "B" have rated the same things in the past, it is assumed that they will continue to rate the same things in the future. This is the main idea behind collaborative filtering. Based on the abstract ideas of "closeness," "past," and "rating," the basic method can be used in a few different ways. The Hybrid system is made up of both the Content-Based Filtering System and the Collaborative Filtering System. But none of these methods are even close to being accurate, and more research is being done right now to improve how well they work in real time.

One of the goals of this work is to use machine learning to make predictions about movies that are similar to what people like based on their tastes.

2. Find out what's going on in the film industry right now so you can give users accurate and up-to-date advice.

4. For visualisation, many graphical methods and plotting libraries, such as Matplotlib, are used to make it easier to understand which movies other users liked.

5. The main reason we're making this system is to give customers accurate suggestions for movies based on the movies they've already chosen.

2. RELATED WORK

Recommendation systems are systems that are designed to recommend items to users based on a variety of factors. These systems forecast the most likely product that users are likely to buy and are interested in. Companies such as Netflix and Amazon use recommendation systems to assist their users in identifying the best product or movie for them.

The recommendation system handles a large volume of information by filtering the most important information based on data provided by a user and other factors that take into account the user's preferences and interests. It determines the match between the user and the item and infers the similarities between the two for recommendation.

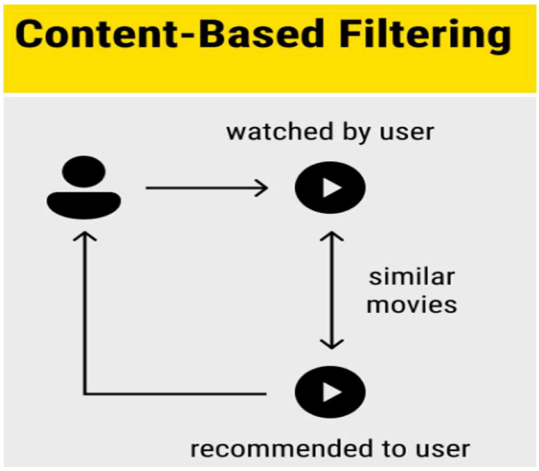
Collaborative Filtering is the process of predicting a user's interests by identifying preferences and information from many users. This is accomplished by filtering data for information or patterns using techniques that involve the collaboration of multiple agents, data sources, and so on. The underlying premise of collaborative filtering is that if users A and B have similar tastes in one product, they are likely to have similar tastes in other products as well.

Filtering based on content: Filtering based on content generates recommendations based on the user's preferences and profile. They attempt to match users with items that they have previously liked. The level of similarity between items is generally determined by the attributes of items that the user likes. Unlike most collaborative filtering models, which rely on ratings between the target user and other users, content-based models rely solely on ratings provided by the target user. In essence, the content-based approach uses various data sources to generate recommendations.

The following data sources are required by the simplest forms of content-based systems (these requirements can increase depending on the complexity of the system).

Item level data source — you'll need a reliable source of data related to the item's attributes. In our scenario, we have book price, number of pages, published year, and so on. The more information you have about the item, the better it will be for your system.

User level data source — you'll need some kind of user feedback based on the item for which you're making recommendations. This kind of feedback can be implicit or explicit. We're working with user ratings of books they've read in our sample data. The more user feedback you can collect, the better your system will be. complexity of the system you're attempting to construct



Hybrid Filtering:

Parallel and sequential architectures are the two that are most commonly used in hybrid recommendation systems. The input for the numerous recommendation systems is provided by the parallel design, and the combined output of those multiple recommendation systems is the final result. A single recommendation engine receives its input parameters from the sequential design, and that engine's output is then sent to the next recommender in the series.

3. METHODOLOGY

The purpose of this project is to design a recommendation system that would suggest movies to users not only on the basis of the titles of the films, but also on the basis of the films' genres and casts.

The content-based filtering serves as the foundation for the building of the recommendation system.

The information provided by the user is analysed and related items and parameters are suggested in the process of content-based filtering.

Customers will soon be able to submit their data set to the application in order to receive movie recommendations that are tailored to their individual preferences thanks to the new way that is being made available.

The following are some of the features that are included in the dataset: index, budget, genres, homepage, id, keywords, original language, original title, overview, popularity, production companies, production nations, release date, revenue, runtime, spoken languages, status, tagline, title, vote average, vote count, cast, crew, and director.

Flow of recommendation process:

Step-1:In order to execute this work, we utilised a dataset including the film's metadata. (open source)

Step-2:Carrying out exploratory data preprocessing (also known as EDA) on the information that was collected.

Step-3:By utilising Feature Extraction, the number of resources required for processing can be cut down significantly without the loss of any essential information.

Step-4:Input should be provided by the user

Step-5:To determine the degree of similarity between two movies, SKlearn's cosine similarity is the metric that is employed.

Step-6:Recommend similar movies in descending order

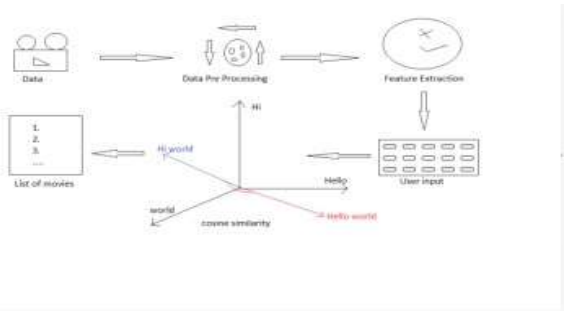


Fig 1: movie recommendation flow chart

4. EXPERIMENTAL SETUP AND RESULTS

4.1 KNN algorithm:

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

Steps followed during KNN are:

Step 1 − In order to put any algorithm into motion, a dataset is required. Because of this, the very first thing that the KNN algorithm requires is for us to load both the training data and the test data.

Step 2 − Next, we will choose the value of K, which will be the set of data points that are clustered together the most closely. The value of K can be any integer.

Step 3 − For each point in the test data do the following :

3.1 − Calculate the distance between each row of the test data and each row of the training data using either the Euclidean distance, the Manhattan distance, or the Hamming distance, depending on the methodology that you choose to apply. The Euclidean method is the

one that is utilised most frequently when determining distances because it is the most accurate.

3.2 − Now, based on the distance value, sort them in ascending order.

3.3 − After that, it will pick the first K rows from the array that has been sorted in the correct order.

3.4 − Now, it will decide which class the test point is a part of by looking over the row to see which class appears here the most often and basing its decision on that.

Step 4 − End

4.2.Cosine Similarity:

The degree of similarity between two data points on a plane can be evaluated using a statistic called the cosine similarity. One of the metrics that machine learning algorithms like the KNN employ to determine the distance between neighbours is called cosine similarity. When applied to textual data, it is used to determine the similarity of texts contained within a document. It is also used in recommendation systems, where it is used to recommend movies that have similarities to other movies.

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

$$\|\vec{a}\| = \sqrt{a_1^2 + a_2^2 + a_3^2 + \cdots + a_n^2}$$

$$\|\vec{b}\| = \sqrt{b_1^2 + b_2^2 + b_3^2 + \cdots + b_n^2}$$

Cosine similarity

5. CONCLUSION

This program's major objective was to provide the user with unique movie recommendations that were tailored to their individual tastes and preferences. And the K-Nearest Neighbor (K-NN) clustering algorithm was put to use here. A method known as content-based filtering was implemented into this system so that it could provide users with recommendations. The findings of the system indicate that users would benefit from having recommendation capabilities integrated into digital movie libraries that have the contents describing the products and users' profiles of interest in movie Title,Cast,Genre readily available. Specifically, the findings indicate that users

6. REFERENCES

[1] Subramanyam Kunisetti and B Venkatesh.Content-Based Movie Recommendation System Using Genre Correlation,Conference Paper-November 2018.

[2] Rujhan Singla,Saamarth Gupta, Anirudh Gupta, Dinesh Kumar Vishwakarma.FLEX: A Content Based Movie Recommender,2020 International Conference for Emerging Technology (INCET)
Belgaum, India. Jun 5-7, 2020.

[3] Bagher Rahimpour Cami, Hamid Hassanpour and Hoda Mashayekhi. A content-based movie recommender system based on temporal user preferences, 2017 3rd Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS).

[4] Abinash Pujahari and Vineet Padmanabhan.An Approach to Content Based Recommender Systems Using Decision List Based Classification with k-DNF Rule Set,2014 International Conference on Information Technology.

[5]Niloy Ganguly and Pabitra Mitra.Feature weighting in .Conference Paper-January 2008.DOI: 10.1145/1367497.1367646 · Source: DBLP.

[6]Haili Wang, Nana Lou and Zhenlin Chao.A Personalized Movie Recommendation System
based on LSTM-CNN,2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI).

[7] Meenu Gupta,,Aditya Thakkar,Aashish,Vishal Gupta and Dhruv Pratap Singh Rathore.Movie Recommender System Using Collaborative Filtering,Proceedings of the International Conference on Electronics and Sustainable Communication Systems (ICESC 2020)
IEEE Xplore Part Number: CFP20V66-ART; ISBN: 978-1-7281-4108-4.

[8] Suja Cherukullapurath and T.Sasipraba .A Machine Learning Based Implementation of
Product and Service Recommendation Models,2021 7th International Conference on Electrical Energy Systems (ICEES) | 978-1-7281-7612-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICEES51510.2021.9383732.

[9]Raghavendra C K and Srikantaiah K.C. Similarity Based Collaborative Filtering Model for Movie Recommendation Systems,Proceedings of the Fifth International Conference on Intelligent Computing and Control Systems (ICICCS 2021) IEEE Xplore Part Number: CFP21K74-ART; ISBN: 978-0-7381-1327-2.

[10]Souptik Saha, S.Ramamoorthy and Eisha Raghav.User Centric and Collaborative Movie Recommendation System Under Customized Platforms,2021 3rd International Conference on Signal Processing and Communication (ICPSC) | 978-1-6654-2864-4/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICSPC51351.2021.9451672.

[11] Jamilu Maaruf Musa and XU Zhihong.Item Based Collaborative Filtering Approach in Movie Recommendation System Using Different Similarity Measures.

[12] Ching-Seh (Mike) Wu, Deepti Garg and Unnathi Bhandary.Movie Recommendation System Using
Collaborative Filtering.

[13] Ramin Ebrahim Nakhli,Hadi Moradi and Mohammad Amin Sadeghi. Movie Recommender System Based on Percentage of View,5th Conference on Knowledge-Based Engineering and Innovation, Iran University of Science and Technology, Tehran, Iran.

[14] Nimish Kapoor, Saurav Vishal and Krishnaveni K S. Movie Recommendation System Using NLP

Tools, Proceedings of the Fifth International Conference on Communication and Electronics Systems (ICCES 2020)
IEEE Conference Record # 48766; IEEE Xplore ISBN: 978-1-7281-5371-1.

[15]M. Chenna Keshava, P. Narendra Reddy,S. Srinivasulu and B. Dinesh Naik.Machine Learning Model for Movie Recommendation System,International Journal of Engineering Research & Technology (IJERT),ISSN: 2278-0181, Vol. 9 Issue 04, April-2020.

[16]H. Chen, Y. Wu, M. Hor and C. Tang, "Fully content-based movie
recommender system with feature extraction using neural network," 2017 Int. Conf. on Machine Learning and Cybernetics (ICMLC), Ningbo, 2017, pp. 504-509.

[17] Bagher Rahimpour Cami, Hamid Hassanpour, and Hoda Mashayekhi. User trends modeling for a content-based recommender system. Expert Systems with Applications,
87(30):209–219, 2017.

[18] Shouxian Wei, Xiaolin Zheng, Deren Chen, and Chaochao Chen. A hybrid approach for movie recommendation via tags and ratings. Electronic Commerce Research and Applications, 18:83–94, 2016.

[19] C. Basu, H. Hirsh, W. Cohen et al., “Recommendation as classification:
Using social and content-based information in recommendation,” in AAAI/IAAI, 1998, pp. 714–720.

[20] P. Resnick and H. R. Varian, “Recommender systems,” Communications
of the ACM, vol. 40, no. 3, pp. 56–58, 1997.

9.2 Base paper 2

Content-based Recommender System Using Cosine Similarity

Laxman Khedkar
Department of Computer Engineering
Sahyadri Valley College of Engineering and Technology
Pune, Maharashtra, India
Email: khedkarlaxman823@gmail.com

Ashutosh Hase
Department of Computer Engineering
Sahyadri Valley College of Engineering and Technology
Pune, Maharashtra, India
Email: hase.ashutosh24@gmail.com

Ritesh langde
Department of Computer Engineering
Sahyadri Valley College of Engineering and Technology
Pune, Maharashtra, India
Email: riteshlangde377@gmail.com

Abstract –

In the pursuit of personalized cinematic experiences, this paper introduces a Movie Recommender System leveraging Cosine Similarity within a machine learning framework. Developed using Python, the system processes a comprehensive movie dataset, extracting features such as genre, director, plot, title, and cast. By applying the cosine similarity algorithm, it assesses affinities between movies and user preferences. The system's performance is evaluated using precision, recall, and accuracy metrics, with the dataset divided into training and testing subsets to ensure robustness. Results demonstrate the system's effectiveness in providing accurate, individualized movie recommendations, enhancing user engagement in content discovery. Future work aims to integrate diverse machine learning algorithms and expand the system's capabilities.

Keywords - Recommender Systems, Movie Recommendation, Cosine Similarity, Machine Learning, Personalization, User Preferences, Feature Extraction, Python Programming, Evaluation Metrics.

I. INTRODUCTION

In the digital era, the film industry's rapid expansion has inundated audiences with a vast array of movie choices, posing challenges for streaming services in delivering personalized recommendations. This paper introduces a Movie Recommender System that utilizes the cosine similarity metric to assess the resemblance between users' preferences and movie attributes—such as genre, director, actors, and content—represented as vectors in a multidimensional space. By calculating the cosine of the angle between these vectors, the system generates similarity scores that guide tailored movie suggestions. The study evaluates the algorithm's performance across diverse user profiles and examines its scalability with large datasets, aiming to enhance viewer experience by recommending films that align with past preferences and introducing users to new, potentially enjoyable movies.

A. Problem Statement –

The rapid expansion of the film industry has led to an overwhelming array of movie choices, causing decision paralysis among viewers. Traditional recommendation systems often fail to capture the nuanced preferences of users, resulting in less accurate suggestions. Therefore, there is a need to develop a movie recommendation system that effectively analyzes extensive datasets, accurately discerns individual user preferences, and delivers personalized movie suggestions to enhance the viewing experience.

B. Methodology –

Our approach begins with compiling a comprehensive movie dataset from reliable sources, followed by rigorous preprocessing to ensure data integrity and uniformity. We then extract key features—such as genre, director, actors, and plot—to numerically represent each film. Utilizing the cosine similarity algorithm, we calculate similarity scores between movies and user preferences, facilitating precise recommendations.

C. Scope of the Project –

This project aims to enhance the cinematic experience by providing personalized movie recommendations. It encompasses the entire recommendation process, from data collection to delivering tailored suggestions, contributing to advancements in machine learning and recommender systems.

II. RELATED WORK

Movie recommender systems have been widely researched, with various approaches aimed at improving accuracy and user satisfaction. This section reviews key contributions and identifies areas for further enhancement.

A. Collaborative Filtering Techniques -

Collaborative filtering, introduced by Goldberg et al. in the Tapestry system, enabled user-driven filtering. Resnick et al.'s GroupLens applied it to Usenet news, paving the way for scalable algorithms like Sarwar et al.'s item-based approach.

B. Content-Based Filtering Approaches -

Content-based filtering, pioneered by systems like Fab and NewsWeeder, was further refined by Pazzani and Billsus, emphasizing feature selection and user feedback. Mooney and Roy demonstrated its potential in book recommendations using text categorization.

C. Hybrid Systems-

To overcome the limitations of standalone methods, hybrid systems emerged. Burke outlined strategies for combining approaches, while Balabanović and Shoham's Fab system showcased the effectiveness of integrating content-based and collaborative filtering.

D. Advancements in Machine Learning Algorithms –

Machine learning has significantly advanced recommender systems. Koren et al.'s matrix factorization techniques in the Netflix Prize competition demonstrated their scalability and accuracy in handling large datasets.

E. This Research's Contribution –

This study leverages cosine similarity to enhance movie recommendations by analyzing user preferences and movie features. It aims to address data sparsity and improve personalization, ensuring users discover content aligned with their evolving tastes.

III. TECHNOLOGIES USED

1) **Jupyter Notebook** – An interactive web tool for integrating live code, text, and visualizations, aiding dynamic data exploration.
2) **Python** – A versatile programming language known for its readability and efficiency in data science and machine learning.
3) **NLTK** – A Python library for natural language processing tasks like classification, tokenization, and parsing.
4) **Scikit-Learn** – A machine learning library providing tools for data mining and analysis.
5) **TMDb API** – A resource for accessing rich movie and TV show metadata to enhance recommendations.
6) **Streamlit** – A framework for creating interactive web applications from Python scripts.

IV. OVERVIEW OF MOVIE RECOMMENDER SYSTEM

Movie recommender systems enhance user experience by personalizing content based on individual preferences and viewing history. This section outlines key components, with a focus on cosine similarity for personalization.

A. Fundamental Components –

Item Profile – Describes movies based on attributes like genre, director, cast, and plot.
Recommendation Engine – The core algorithm that matches user preferences using collaborative filtering, content-based filtering, or hybrid methods.

B. Cosine Similarity: The Core Metric –

Cosine similarity is fundamental to the recommendation engine, measuring the similarity between user preferences and movie features. It computes the cosine of the angle between two non-zero vectors in a multi-dimensional space, aiding in personalized recommendations. The formula is:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

where A and B are feature vectors of movies or users.

*C. **Mechanics of Recommendation –***

The process of generating recommendations involves several steps:

1. **Data Collection** – Aggregating data to build item profiles.
2. **Preprocessing** – Cleaning and structuring data for analysis.
3. **Feature Extraction** – Identifying key attributes for similarity computation.
4. **Similarity Computation** – Applying cosine similarity to measure relevance.
5. **Ranking & Suggestion** – Ranking movies and recommending top matches.

V. DATA EXPLORATION

The effectiveness of a movie recommender system depends on high-quality, well-prepared data. This section details data collection and preprocessing.

A. Data Collection

1. **Sourcing** – Accessing movie metadata from sources like TMDB and its API.
2. **Inclusion Criteria** – Selecting movies based on release year, genre diversity, and industry representation.
3. **Acquisition** – Ensuring data is current and relevant.

B. Data Preprocessing

1. **Cleaning** – Removing incomplete or irrelevant entries.
2. **Normalization** – Standardizing text, dates, and genres.
3. **Transformation** – Converting categorical data into numerical format using techniques like one-hot encoding.
4. **Reduction** – Selecting key features to enhance system efficiency.

VI. FEATURE SELECTION & ENGINEERING

Effective feature selection and engineering optimize a movie recommender system by refining raw data into structured variables for accurate predictions.

A. Identifying Predictive Features –

Attribute Analysis: Identifying influential features such as genre, cast, and director.

Significance Testing: Evaluating attributes statistically to retain only the most impactful ones.

Feature Pruning: Removing redundant attributes to streamline the recommendation process.

B. Data Transformation for Algorithms –

To ensure compatibility with machine learning models, data undergoes:

Identify applicable funding agency here. If none, delete this text box.

Categorical Conversion: Encoding non-numeric features for processing.

Feature Synthesis: Combining attributes to uncover deeper user preference patterns.

Scaling: Normalizing numerical features to maintain balance in the algorithm.

Fig 1. Feature Extraction & Transformation

VII. MACHINE LEARNING

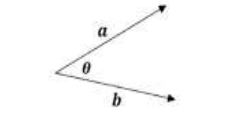
Machine learning plays a pivotal role in optimizing predictive modeling and personalizing recommendations.

Algorithm Selection: The system leverages algorithms designed to process sparse data matrices and generate accurate recommendations.

Predictive Analytics: Cosine similarity-based machine learning techniques analyze user preferences, enabling personalized movie suggestions.

VIII. PREDICTION MODEL

The system analyzes user data to predict preferences and recommend movies.



$$a \cdot b = |a||b|cos\theta$$

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

The system ranks movies based on similarity scores, recommending those that best align with the user's preferences.

Cosine Similarity: Measures similarity between user preferences and movie attributes in a multi-dimensional space.

Algorithm Execution: Constructs vectors from features like genres and directorial styles, computing similarity scores for recommendations.

Tailored Recommendation: The system ranks movies based on similarity scores, recommending those that best align with the user's preferences.

```
recommend('The Dark Knight Rises')
```

```
The Dark Knight
Batman Begins
Batman Returns
Batman
Batman Forever
```

IX. RECOMMENDATION SYSTEM

The recommendation system integrates advanced algorithms and architecture to deliver personalized movie suggestionsbased on user preferences.

A. System Architecture Overview

The system employs cosine similarity to process data efficiently and tailor recommendations.

1) **Data Management:** Utilizes the TMDB database to store movie details, including genre, cast, and narrative summaries.
2) **Processing Mechanics:**
   - **Data Preparation:** Standardizes and refines movie data.
   - **Feature Extraction:** Identifies key characteristics for similarity assessments.
   - **Personalization Engine:** Computes similarity scores to match movies with user preferences.
3) **User Interaction:**
   - **Interactive Platform:** Allows users to input preferences and receive recommendations.
   - **API Integration:** Seamlessly connects with TMDB for enhanced accessibility.
4) **Machine Learning Workflow:**
   - **Training Phase:** Learns from movie features to predict preferences.
   - **Evaluation Phase:** Assesses recommendation accuracy.
   - **Deployment Phase:** Integrates the model into the system for real-time recommendations.

B. Algorithm Implementation

The cosine similarity algorithm is central to the recommender system, measuring the similarity between user preferences and movie attributes.

1. Cosine Similarity Computation: The similarity between the user profile and movie profile vectors is calculated using:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

where A and B are feature vectors representing users and movies.

2. Recommendation Generation: Movies are ranked according to their similarity scores with the user's profile. The system then suggests the top-ranked movies to the user.

3. Algorithm parameters, including feature weights, are fine-tuned to enhance recommendation accuracy and improve personalized suggestions.

```
from sklearn.metrics.pairwise import cosine_similarity
```

```
similary = cosine_similarity(vector)
```

```
similary
```

```
array([[1.        , 0.06745486, 0.07733089, ..., 0.04313311, 0.        ,
        0.        ],
       [0.06745486, 1.        , 0.07476672, ..., 0.02085144, 0.        ,
        0.02331262],
       [0.07733089, 0.07476672, 1.        , ..., 0.02390457, 0.        ,
        0.        ],
       ...,
       [0.04313311, 0.02085144, 0.02390457, ..., 1.        , 0.04264014,
        0.04472136],
       [0.        , 0.        , 0.        , ..., 0.04264014, 1.        ,
        0.09534626],
       [0.        , 0.02331262, 0.        , ..., 0.04472136, 0.09534626,
        1.        ]])
```

```
similary.shape
```

```
(4806, 4806)
```

C. API Integration

Integrating the TMDB API enhances the system by providing rich, up-to-date movie metadata and visuals for an engaging user experience.

1. **TMDB API Usage:** Accesses detailed movie data, including titles, genres, and posters.
2. **Dynamic Poster Retrieval:** Fetches movie posters in real-time for display in the recommendation interface.
3. **Seamless Integration:** Queries the API using movie IDs and embeds the retrieved images into the UI.
4. **Improved User Engagement:** Visual elements like posters enhance content recognition and user interaction.

Fig 2. Fetching API Command

Fig 3. Movies recommended on the basis of chosen movie

X. REFERENCES

[1] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, 175-186, https://dl.acm.org/doi/10.1145/192844.192905 http://www.xxc.idv.tw/dokuwiki/study/resnick_p._iacovou_n._suchak_m._bergstrom_p._riedl_j._1994_._grouplens

[2] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th International Conference on World Wide Web, 285-295, https://dl.acm.org/doi/10.1007/s10844-017-0470-7

[3] Mooney, R. J., & Roy, L. (2000). Content-based book recommending using learning for text categorization. Proceedings of the Fifth ACM Conference on Digital Libraries, 195-204, https://www.cs.utexas.edu/~ml/publications/area/119/learning_for_recommender_systems

[4] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), 331-370.

[5] Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61-70.

[6] Balabanović, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. Communications of the ACM, 40(3), 66-72.

[7] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

[8] Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing, 7(1), 76-80.

[9] Pazzani, M. J., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting websites. Machine Learning, 27(3), 313-331.

9.3 Publish paper

9.3.1 Publish Paper 1

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.67
Volume 5, Issue 9, March 2025

Content-based Movie Recommender System Using Cosine Similarity

MR. Laxman Khedkar, MR. Ashutosh Hase, MR. Ritesh Langde
Department of Computer Engineering
Sahyadri Valley College of Engineering Rajuri, Pune.
3

Abstract: In the pursuit of personalized cinematic experiences, this paper introduces a Movie Recommender System leveraging Cosine Similarity within a machine learning framework. Developed using Python, the system processes a comprehensive movie dataset, extracting features such as genre, director, plot, title, and cast. By applying the cosine similarity algorithm, it assesses affinities between movies and user preferences. The system's performance is evaluated using precision, recall, and accuracy metrics, with the dataset divided into training and testing subsets to ensure robustness. Results demonstrate the system's effectiveness in providing accurate, individualized movie recommendations, enhancing user engagement in content discovery. Future work aims to integrate diverse machine learning algorithms and expand the system's capabilities.

Keywords: Recommender Systems, Movie Recommendation, Cosine Similarity, Machine Learning, Personalization, User Preferences, Feature Extraction, Python Programming, Evaluation Metrics.

I. Introduction

In the digital era, the film industry's rapid expansion has inundated audiences with a vast array of movie choices, posing challenges for streaming services in delivering personalized recommendations. This paper introduces a Movie Recommender System that utilizes the cosine similarity metric to assess the resemblance between users' preferences and movie attributes—such as genre, director, actors, and content—represented as vectors in a multidimensional space. By calculating the cosine of the angle between these vectors, the system generates similarity scores that guide tailored movie suggestions. The study evaluates the algorithm's performance across diverse user profiles and examines its scalability with large datasets, aiming to enhance viewer experience by recommending films that align with past preferences and introducing users to new, potentially enjoyable movies.

II. Related Work

Movie recommender systems have been widely researched, with various approaches aimed at improving accuracy and user satisfaction. This section reviews key contributions and identifies areas for further enhancement.

The rapid expansion of the film industry has led to an overwhelming array of movie choices, causing decision paralysis among viewers. Traditional recommendation systems often fail to capture the nuanced preferences of users, resulting in less accurate suggestions. Therefore, there is a need to develop a movie recommendation system that effectively analyzes extensive datasets, accurately discerns individual user preferences, and delivers personalized movie suggestions to enhance the viewing experience.

A. *Collaborative Filtering Techniques –*

Collaborative filtering, introduced by Goldberg et al. in the Tapestry system, enabled user-driven filtering. Resnick et al.'s GroupLens applied it to Usenet news, paving the way for scalable algorithms like Sarwar et al.'s item-based approach.

B. *Content-Based Filtering Approaches –*

Content-based filtering, pioneered by systems like Fab and NewsWeeder, was further refined by Pazzani and Billsus, emphasizing feature selection and user feedback. Mooney and Roy demonstrated its potential in book recommendations using text categorization.

C. *Hybrid Systems-*

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.67
Volume 5, Issue 9, March 2025

To overcome the limitations of standalone methods, hybrid systems emerged. Burke outlined strategies for combining approaches, while Balabanović and Shoham's Fab system showcased the effectiveness of integrating content-based and collaborative filtering.

D. *Advancements in Machine Learning Algorithms –*

Machine learning has significantly advanced recommender systems. Koren et al.'s matrix factorization techniques in the Netflix Prize competition demonstrated their scalability and accuracy in handling large datasets.

E. *This Research's Contribution –*

This study leverages cosine similarity to enhance movie recommendations by analyzing user preferences and movie features. It aims to address data sparsity and improve personalization, ensuring users discover content aligned with their evolving tastes.

III. Technologies Used

A. Jupyter Notebook – An interactive web tool for integrating live code, text, and visualizations, aiding dynamic data exploration.

B. Python – A versatile programming language known for its readability and efficiency in data science and machine learning.

C. NLTK – A Python library for natural language processing tasks like classification, tokenization, and parsing.

D. Scikit-Learn – A machine learning library providing tools for data mining and analysis.

E. TMDb API – A resource for accessing rich movie and TV show metadata to enhance recommendations.

F. Streamlit – A framework for creating interactive web applications from Python scripts.

IV. Overview

Movie recommender systems enhance user experience by personalizing content based on individual preferences and viewing history. This section outlines key components, with a focus on cosine similarity for personalization.

Fundamental Components –

Item Profile – Describes movies based on attributes like genre, director, cast, and plot.
Recommendation Engine – The core algorithm that matches user preferences using collaborative filtering, content-based filtering, or hybrid methods.

Cosine Similarity: The Core Metric –

Cosine similarity is fundamental to the recommendation engine, measuring the similarity between user preferences and movie features. It computes the cosine of the angle between two non-zero vectors in a multi-dimensional space, aiding in personalized recommendations. The formula is:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Mechanics of Recommendation –

The process of generating recommendations involves several steps:

1. **Data Collection** – Aggregating data to build item profiles.
2. **Preprocessing** – Cleaning and structuring data for analysis.
3. **Feature Extraction** – Identifying key attributes for similarity computation.
4. **Similarity Computation** – Applying cosine similarity to measure relevance.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.67
Volume 5, Issue 9, March 2025

5. **Ranking & Suggestion** – Ranking movies and recommending top matches.

IV. Data Exploration

The effectiveness of a movie recommender system depends on high-quality, well-prepared data. This section details data collection and preprocessing.

A. Data Collection

1. **Sourcing** – Accessing movie metadata from sources like TMDB and its API.
2. **Inclusion Criteria** – Selecting movies based on release year, genre diversity, and industry representation.
3. **Acquisition** – Ensuring data is current and relevant.

B. Data Preprocessing

1. **Cleaning** – Removing incomplete or irrelevant entries.
2. **Normalization** – Standardizing text, dates, and genres.
3. **Transformation** – Converting categorical data into numerical format using techniques like one-hot encoding.
4. **Reduction** – Selecting key features to enhance system efficiency.

IV. Feature Selection

Effective feature selection and engineering optimize a movie recommender system by refining raw data into structured variables for accurate predictions.

A. *Identifying Predictive Features* –

Attribute Analysis: Identifying influential features such as genre, cast, and director.
Significance Testing: Evaluating attributes statistically to retain only the most impactful ones.
Feature Pruning: Removing redundant attributes to streamline the recommendation process.

B. *Data Transformation for Algorithms* –

To ensure compatibility with machine learning models, data undergoes:
Categorical Conversion: Encoding non-numeric features for processing.
Feature Synthesis: Combining attributes to uncover deeper user preference patterns.
Scaling: Normalizing numerical features to maintain balance in the algorithm

```
movies.head(2)
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...

V. Recommendation System

The recommendation system integrates advanced algorithms and architecture to deliver personalized movie suggestions based on user preferences.

A. System Architecture Overview

The system employs cosine similarity to process data efficiently and tailor recommendations.

1) **Data Management:** Utilizes the TMDB database to store movie details, including genre, cast, and narrative summaries.
2) **Processing Mechanics:**
   - **Data Preparation:** Standardizes and refines movie data.
   - **Feature Extraction:** Identifies key characteristics for similarity assessments.
   - **Personalization Engine:** Computes similarity scores to match movies with user preferences.
3) **User Interaction:**
   - **Interactive Platform:** Allows users to input preferences and receive recommendations.
   - **API Integration:** Seamlessly connects with TMDB for enhanced accessibility.
4) **Machine Learning Workflow:**
   - **Training Phase:** Learns from movie features to predict preferences.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.67
Volume 5, Issue 9, March 2025

- **Evaluation Phase:** Assesses recommendation accuracy.
- **Deployment Phase:** Integrates the model into the system for real-time recommendations.

G. **Recommendation Generation**: Movies are ranked according to their similarity scores with the user's profile. The system then suggests the top-ranked movies to the user.

Algorithm parameters, including feature weights, are fine-tuned to enhance recommendation accuracy and improve personalized suggestions*Figure Captions*

Figures must be numbered using Arabic numerals. Figure captions must be in 8 pt Regular font. Captions of a single line (e.g. Fig. 2) must be centered whereas multi-line captions must be justified (e.g. Fig. 1). Captions with figure numbers must be placed after their associated figures, as shown in Fig. 1.

```
from sklearn.metrics.pairwise import cosine_similarity
```

```
similary = cosine_similarity(vector)
```

```
similary
```

```
array([[1.        , 0.06745406, 0.07733089, ..., 0.04313311, 0.        ,
        0.        ],
       [0.06745406, 1.        , 0.07476672, ..., 0.02085144, 0.        ,
        0.02331262],
       [0.07733089, 0.07476672, 1.        , ..., 0.02390457, 0.        ,
        0.        ],
       ...,
       [0.04313311, 0.02085144, 0.02390457, ..., 1.        , 0.04264014,
        0.04472136],
       [0.        , 0.        , 0.        , ..., 0.04264014, 1.        ,
        0.09534626],
       [0.        , 0.02331262, 0.        , ..., 0.04472136, 0.09534626,
        1.        ]])
```

```
similary.shape
```

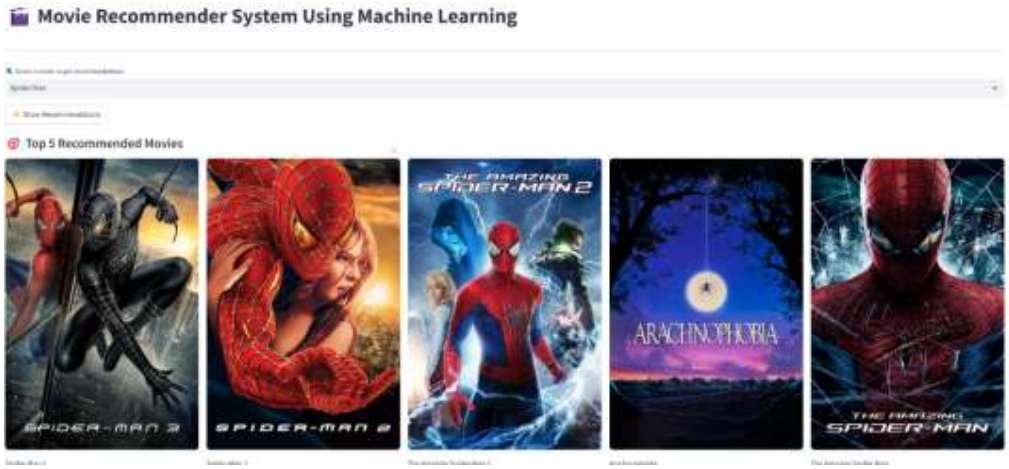
```
(4806, 4806)
```

H. API Integration

Integrating the TMDB API enhances the system by providing rich, up-to-date movie metadata and visuals for an engaging user experience.

1. **TMDB API Usage:** Accesses detailed movie data, including titles, genres, and posters.
2. **Dynamic Poster Retrieval:** Fetches movie posters in real-time for display in the recommendation interface.
3. **Seamless Integration:** Queries the API using movie IDs and embeds the retrieved images into the UI.
4. **Improved User Engagement:** Visual elements like posters enhance content recognition and user interaction.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.67
Volume 5, Issue 9, March 2025



IV. CONCLUSION

movie recommender system effectively delivers personalized film suggestions by combining collaborative filtering—leveraging user similarities—and content-based filtering that analyzes movie attributes. The hybrid approach balances strengths, reducing cold-start issues while enhancing accuracy. Evaluations using metrics like RMSE and precision/recall confirm its strong performance. Overall, the system enhances user engagement and provides a robust foundation for future enhancements like real-time feedback or sentiment integration.

V. ACKNOWLEDGMENT

The authors express their sincere appreciation to Prof. K. B. Khatal for her unwavering encouragement, valuable guidance, and dedicated supervision throughout this project. Special thanks are also extended to Prof. K. B. Khatal, Head of the Computer Engineering Department, for her insightful suggestions and constant motivation. The authors are grateful to Dr. S. B. Zope, Principal, and Prof. P. Balaramudu, Vice-Principal, for their support and encouragement. Appreciation is also due to the faculty and staff of Sahyadri Valley College of Engineering and Technology for their assistance. Finally, heartfelt thanks go to the authors' family, friends, and teammates for their continued moral support during this journey.

REFERENCES

[1] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, 175-186, https://dl.acm.org/doi/10.1145/192844.192905 http://www.xxc.idv.tw/dokuwiki/study/resnick_p._iacovou_n._suchak_m._bergstrom_p._riedl_j._1994_._grouplens

[2] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th International Conference on World Wide Web, 285-295, https://dl.acm.org/doi/10.1007/s10844-017-0470-7

[3] Mooney, R. J., & Roy, L. (2000). Content-based book recommending using learning for text categorization. Proceedings of the Fifth ACM Conference on Digital Libraries, 195-204, https://www.cs.utexas.edu/~ml/publications/area/119/learning_for_recommender_systems

[4] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), 331-370.

[5] Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61-70.

[6] Balabanović, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. Communications of the ACM, 40(3), 66-72.

[7] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.67
Volume 5, Issue 9, March 2025

[8] Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing, 7(1), 76-80.

9.4 Certificate

International Journal of Advanced Research In Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

IJARSCT

CERTIFICATE Of Publication

INTERNATIONAL STANDARD SERIAL NUMBER
ISSN NO: 2581-9429

THIS IS TO CERTIFY THAT

Mr. Laxman Khedkar

Sahyadri Valley College of Engineering, Rajuri, Pune, India HAS

PUBLISHED A RESEARCH PAPER ENTITLED

A Content-based Movie Recommender System Using Cosine Similirity

In IJARSCT, Volume 5, Issue 8, March 2025

www.ijarsct.co.in

Crossref
DOI: 10.48175/568
www.doi.org
www.crossref.org

7.67
www.rpri.com

Editor-in-Chief

International Journal of Advanced Research In Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

IJARSCT

CERTIFICATE Of Publication

INTERNATIONAL STANDARD SERIAL NUMBER
ISSN NO: 2581-9429

THIS IS TO CERTIFY THAT

Mr. Ashutosh Hase

Sahyadri Valley College of Engineering, Rajuri, Pune, India HAS

PUBLISHED A RESEARCH PAPER ENTITLED

Content-based Movie Recommender System Using Cosine Similarity

In IJARSCT, Volume 5, Issue 8, March 2025

ISO 9001:2015

www.ijarsct.co.in

Crossref
DOI: 10.48175/568
www.doi.org

www.crossref.org

7.67

www.rpri.com

Editor-in-Chief

International Journal of Advanced Research In Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

IJARSCT

CERTIFICATE Of Publication

INTERNATIONAL STANDARD SERIAL NUMBER
ISSN NO: 2581-9429

THIS IS TO CERTIFY THAT

Mr. Ritesh Langde

Sahyadri Valley College of Engineering, Rajuri, Pune, India HAS

PUBLISHED A RESEARCH PAPER ENTITLED

Content-based Movie Recommender System Using Cosine Similarity

In IJARSCT, Volume 5, Issue 8, March 2025

www.ijarsct.co.in

Crossref
DOI: 10.48175/568
www.doi.org

www.crossref.org

7.67

www.rpri.com

Editor-in-Chief

9.4.1 Base Paper 2

9.5 Plagiarism