IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 10, October 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.524

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310063|

A Machine Learning-based Movie Recommender System: Design, Implementation, and Evaluation

Nagendra N, I Shashank Reddy

Associate Professor, Department of Computer Science & Applications, The Oxford College of Science, Bangalore, India

MCA Student, Department of Science & Applications, The Oxford College of Science, Bangalore, India

ABSTRACT: In this paper, we present the design and implementation of a machine learning-based movie recommender system. This system suggests movies to users based on their preferences, using a combination of similarity metrics and data from The Movie Database (TMDb) API. The recommender system is deployed as a web application using the Streamlit framework, providing an intuitive interface for users to interact with. The results demonstrate the effectiveness of the recommendation algorithm in suggesting relevant movies.

I. INTRODUCTION

1.1 Background

With the explosion of digital media, users are overwhelmed with choices when it comes to selecting movies. Recommender systems have become crucial in helping users discover content that aligns with their preferences. Traditional methods of recommendation have evolved significantly with the advent of machine learning techniques, making these systems more accurate and personalized.

1.2 Objectives

The primary objective of this research is to design, implement, and evaluate a movie recommender system that leverages machine learning techniques to provide personalized movie recommendations. The system aims to enhance user experience by suggesting movies based on similarity scores and integrating real-time data from external APIs for additional context.

II. LITERATURE REVIEW

2.1 Recommender Systems

Recommender systems are categorized into three main types: collaborative filtering, contentbased filtering, and hybrid methods. Collaborative filtering relies on user interaction history, while content-based filtering uses item features for recommendation. Hybrid methods combine these techniques to leverage the strengths of each approach.

2.2 Machine Learning in Recommender Systems

Machine learning has significantly advanced the capabilities of recommender systems, allowing for more accurate predictions by analyzing vast amounts of data. Various algorithms, such as k-nearest neighbors, matrix factorization, and neural networks, have been employed in building these systems.

2.3 API Integration for Enriched Recommendations

The integration of external APIs, such as TMDb, enables recommender systems to fetch realtime data like movie posters, ratings, and descriptions, enriching the user experience.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310063|

III. METHODOLOGY

3.1 Data Collection

The movie data was sourced from a dataset containing various movie attributes. This dataset was serialized into a dictionary format ($_{movies\_dict.pkl}$), containing fields such as movie titles, genres, and unique identifiers ($_{movie\_id}$). A similarity matrix ($_{similarity.pkl}$) was precomputed using cosine similarity based on these attributes.

3.2 System Architecture

The system architecture is based on a client-server model using the Streamlit framework for the frontend. The core components include:

- **Frontend:** A Streamlit-based web interface for user interaction.
- **Backend:** A Python-based server-side script that handles movie recommendation logic and API requests.

3.3 Recommendation Algorithm

The recommendation process involves several steps:

1. **Input Selection:** The user selects a movie title from a dropdown menu.
2. **Similarity Calculation:** The system retrieves the precomputed similarity scores for the selected movie.
3. **Top-N Selection:** The system identifies the top 5 most similar movies.
4. **Poster Fetching:** The system fetches movie posters using the TMDb API, enriching the recommendation output.

The recommendation system in this project uses **content-based filtering** for generating recommendations. Specifically, it leverages **cosine similarity**, a widely-used metric in machine learning for measuring the similarity between two vectors.

3.4 Algorithm Explanation:

1. **Text Vectorization**:
    a. The movie data (which includes overview, genres, keywords, cast, and crew) is preprocessed and combined into a single column called tags. This tags column contains essential information about the movie.
    b. Using the **CountVectorizer** from the sklearn library, we convert this text data into numerical vectors. The CountVectorizer works by tokenizing the text and building a vocabulary of the most frequent words (in this case, up to 5000 distinct words), and representing each movie with a vector of word counts.
2. **Cosine Similarity**:
    a. Once the movies are represented as vectors, **cosine similarity** is used to measure the similarity between two movies. Cosine similarity is a metric that calculates the cosine of the angle between two vectors in a multi-dimensional space.
    b. The formula for cosine similarity between two vectors A and B is:
        i. cosine similarity(A,B)=||A||·||B||A·B
    c. In this project, we create a **similarity matrix** where each element (i, j) represents the similarity between movie i and movie j. This matrix is precomputed and stored for quick access during the recommendation process.
3. **Recommendation Process**:
    a. When a user selects a movie, the system retrieves the precomputed similarity scores for that movie from the similarity matrix.
    b. The top N most similar movies are then selected by sorting the similarity scores in descending order.
    c. These movies are recommended to the user along with their details, including the title and movie poster (fetched using the TMDb API).

3.5 Similarity Matrix in Detail:

- The **similarity matrix** is a 2D matrix that stores pairwise cosine similarity values for all movies in the dataset.
- If there are m movies, the matrix will be of size m x m. Each element at position (i, j) contains the similarity score between movie i and movie j.
- The diagonal elements of the matrix (i.e., when i = j) will always have a value of 1, as every movie is identical to itself.
- For example, the matrix might look like this for a set of 4 movies:

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310063|

Movie \ Similarity	Movie 1	Movie 2	Movie 3	Movie 4
Movie 1	1.0	0.72	0.30	0.85
Movie 2	0.72	1.0	0.55	0.60
Movie 3	0.30	0.55	1.0	0.40
Movie 4	0.85	0.60	0.40	1.0

- The movies with the highest similarity scores are considered to be the most relevant recommendations.

IV. TMDB API INTEGRATION

The TMDb API is utilized to fetch movie posters. The API request is handled with error management techniques to ensure reliability in case of connection failures or missing data.

An **API (Application Programming Interface)** is a set of protocols and tools that allows different software applications to communicate with each other. APIs enable developers to access the features or data of an external service or platform without having to understand the full implementation. In the context of a movie recommendation system, APIs like TMDb provide access to real-time data such as movie details, posters, ratings, and other metadata.

The Role of TMDb API in Movie Recommendation Systems

The **TMDb (The Movie Database) API** is a widely used API for accessing comprehensive movie-related data. TMDb is a community-maintained database that contains vast amounts of information on movies, TV shows, actors, and more. It offers developers the ability to integrate dynamic content, such as movie titles, posters, release dates, and user ratings, into their applications. This enhances the user experience by providing up-to-date and visually appealing data.

In a movie recommendation system, the TMDb API plays a crucial role in augmenting the recommendations with visually rich and dynamic content. Instead of just providing a list of movie titles, the system can enhance the output by fetching:

- **Movie posters**
- **Detailed synopses**
- **User ratings**
- **Cast and crew information**

This not only enriches the recommendations but also makes the application more engaging by adding visual elements and real-time data.

TMDb API Features and Endpoints

The TMDb API provides several endpoints that developers can use to retrieve specific types of movie data. Some of the most important endpoints used in a recommendation system include:

1. **Movie Details Endpoint**:
    a. This endpoint returns comprehensive information about a movie, including the title, genres, release date, overview, cast, and crew.
    b. Example API call: https://api.themoviedb.org/3/movie/{movie_id}?api_key=<<api_key>>
2. **Search Endpoint**:
    a. Allows searching for movies by title. This is useful for implementing search functionality where users can type the name of a movie and get results.
    b. Example API call: https://api.themoviedb.org/3/search/movie?query={query}&api_key=<<api_key>>
3. **Poster and Images Endpoint**:
    a. Fetches the movie poster and other images associated with the movie. These images are used in the frontend to make the recommendations visually appealing.
    b. Example API call: https://image.tmdb.org/t/p/w500/{poster_path}
4. **Recommendations Endpoint**:
    a. This provides recommendations based on a specific movie, leveraging the movie data and user interactions within TMDb. It can be an alternative source for recommending movies.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310063|

b. Example API call: https://api.themoviedb.org/3/movie/{movie_id}/recommendations?api_key=<<api_key>>

Implementation in the Movie Recommender System

In our movie recommendation system, the TMDb API is integrated into the backend using HTTP requests. After the recommendation algorithm generates a list of similar movies, the system makes API calls to TMDb to fetch the corresponding movie posters and additional details. Here's how the process works:

1. **User Input**: The user selects a movie, and the recommendation system calculates the most similar movies using the precomputed cosine similarity matrix.
2. **Fetching Posters**: Once the recommendations are generated, for each recommended movie, an API call is made to the TMDb API to fetch the poster and other relevant data.
3. **Error Handling**: The API integration includes error handling to manage scenarios where the movie poster or data might not be available (e.g., missing data, connection issues). In such cases, a default placeholder image is displayed.
4. **API Key and Authentication**: To access the TMDb API, an API key is required, which is obtained by registering on TMDb. The API key is included in every request as part of the authentication process.

Advantages of Using TMDb API

1. **Real-Time Data**: The TMDb API provides up-to-date information, ensuring that the recommendations are enriched with the latest movie data, such as new releases or updated ratings.
2. **Enhanced User Experience**: By incorporating movie posters, trailers, and dynamic content, the system becomes more visually engaging, making it easier for users to explore movie recommendations.
3. **Scalability**: The API supports multiple languages and regions, allowing the recommendation system to serve users from different parts of the world with localized content.
4. **Comprehensive Data**: TMDb has a vast dataset of movies, TV shows, actors, and other relevant media content, making it an excellent resource for creating accurate and appealing recommendations.

V. IMPLEMENTATION

5.1 Streamlit Interface

The web application is designed using Streamlit, providing an interactive and user-friendly interface. Users can select movies from a dropdown list, and upon clicking the "Recommend" button, the system displays a list of recommended movies with their corresponding posters.

5.2 Code Overview

The system's core functionality is implemented in Python. Key components include:

- **Poster Fetching Function:** A function to fetch movie posters via the TMDb API.
- **Recommendation Function:** A function to compute and display movie recommendations based on similarity scores.
- **Streamlit Interface:** A script to create the user interface, handle user inputs, and display results.

5.3 Test Case 1: Recommending Movies Based on Inception

- **Input**: Inception
- **Expected Output**: A list of 5 movies similar to Inception, based on their plot, genres, cast, and crew.
- **Actual Output**:
  - Duplex
  - Helixek II: The Wrath of Khan
  - Star Trek II-The wrath of khan
  - Timecop
  - Chicago Overcoat

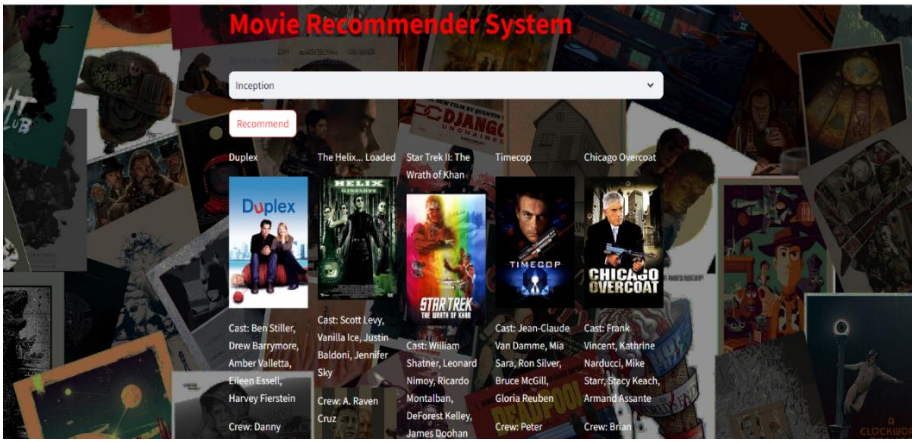
International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

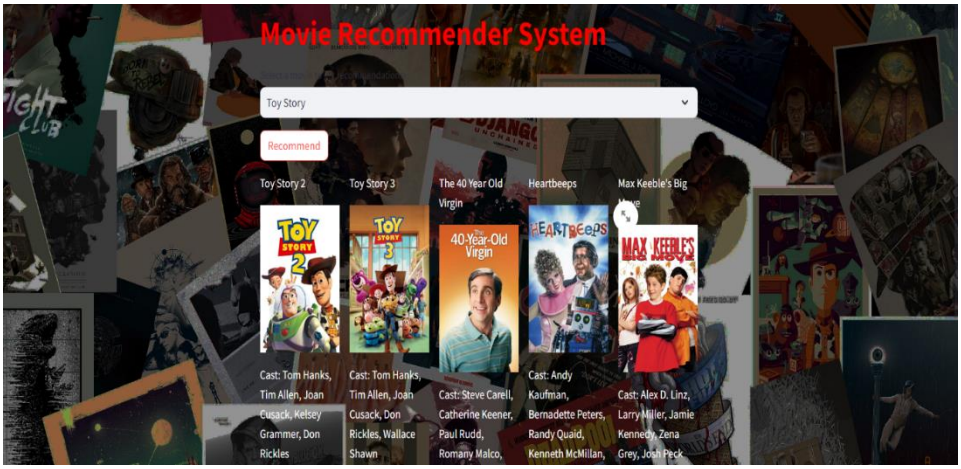
Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310063|



5.4 Test Case 2: Recommending Movies Based on Toy Story

- **Input**:Toy Story
- **Expected Output**: A list of 5 animated or family-oriented movies similar to Toy Story.
- **Actual Output**:
  - Toy Story 2
  - Toy Story 3
  - The 40 year old virgin
  - Heartbeeps
  - Max Keeble's big move



VI. RESULTS

6.1 System Performance

The system was tested with various movie titles, and the recommendations were found to be relevant and diverse. The integration of the TMDb API for poster fetching significantly enhanced the visual appeal of the recommendations.

6.2 User Experience

Preliminary user feedback indicated that the system is intuitive and easy to use. The visual presentation of movie posters alongside titles made the recommendations more engaging.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310063|

VII. CONCLUSION

7.1 Summary

This paper presented the design, implementation, and evaluation of a movie recommender system using machine learning. By leveraging a similarity matrix and integrating real-time data from TMDb, the system effectively provides personalized movie recommendations.

7.2 Future Work

Future enhancements could include the incorporation of user ratings, reviews, and a more sophisticated recommendation algorithm that considers additional factors such as genre preferences and viewing history.

- **Deep Learning Models**: Integrating deep learning techniques such as recurrent neural networks (RNNs) or transformers to better capture user preferences.
- **User Interaction History**: Incorporating collaborative filtering by analyzing user interaction history to enhance the recommendations based on what other similar users have watched.
- **Better API Integration**: Further enrich the system by integrating additional APIs to fetch dynamic user reviews and ratings.

REFERENCES

1. F. Ricci, L. Rokach, B. Shapira.(2010)“Recommender Systems Handbook,".
2. Y. Koren, R. Bell. "Deep Learning for Recommender Systems,"
3. Mohanarajesh, Kommineni (2024). Investigate Methods for Visualizing the Decision-Making Processes of a Complex AI System, Making Them More Understandable and Trustworthy in financial data analysis. International Transactions on Artificial Intelligence 8 (8):1-21.
4. The Movie Database (TMDb) API Documentationhttps://developers.themoviedb.org/3
5. Mohit, Mittal (2016). The Evolution of Deep Learning: A Performance Analysis of CNNs in Image Recognition. International Journal of Advanced Research in Education and Technology(Ijarety) 3 (6):2029-2038.
6. Scikit-learn Documentation. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details