Movie Recommendation System Using Machine Learning

Name: Nikita Bisht

Email: nikitab1709@gmail.com

Date: [23-09-2024]

Course/Institute: [Lovely Professional University]

2. Introduction

Purpose of the Project

The goal of this project is to develop an intelligent movie recommendation system that suggest movies similar to the one selected by the user. It leverage machine learning concepts and precomputed similarity values to make fast and effective recommendations. The application uses Streamlit to provide a user-friendly web interface where users can select a movie and instantly get similar movie recommendations with poster preview.

Problem Statement

In today's world, users are flooded with entertainment choices.

With so many movies released across genres and platforms, it becomes difficult for users to decide what to watch next. Traditional search methods are either too generic or not personalized. Hence, users often waste time browsing instead of enjoying their time watching movies.

Solution Overview

To tackle this issue, we created a recommendation system that utilize precomputed cosine similarity values based on movie features (like titles, genres, tags, etc.). When a user selects a movie, the system fetches and displays the top 5 similar movies, along with their posters using the TMDB API, to enhance visual appeal and user engagement.

3. Tools and Technologies Used

The following technologies and tools were used to implement the project:

Languages & Libraries:

- Python: Core language used for implementation
- Pickle: To load precomputed data (movies and similarity matrix)
- Requests: For making API calls to TMDB

Web Framework:

- Streamlit: Used to build the front-end and user interface

Other Tools:

- VS Code: Code editor
- GitHub: For version control
- SetupTools: For packaging the application

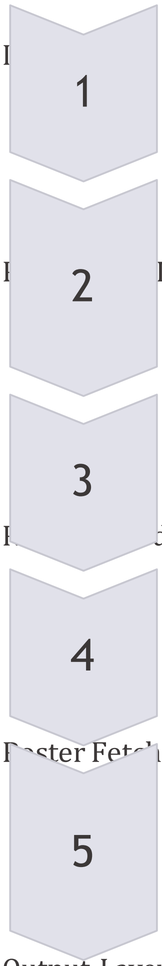
External Services:

- TMDB API: Used to fetch movie posters dynamically based on movie ID

These tools were chosen for their ease of integration, performance, and community support.

4. System Design / Architecture

The recommendation system follows a modular architecture:



I

User selects a movie from the dropdown list in the Streamlit interface.

F Layer

The system finds the index of the selected movie and retrieves its similarity values from a precomputed similarity matrix.

F dule

It sorts the most similar movies using cosine similarity and filters the top 5.

P ster Fet her

It uses the TMDB API to fetch the posters for each recommended movie.

Output Layer

Displays the recommended movie titles with posters in a clean and visually appealing layout using Streamlit columns.

Diagram (Textual Representation):

> User Input ⋆ Similarity Matrix Lookup ⋆ Top Matches ⋆ TMDB API Call ⋆ UI Display

This architecture ensures a fast, user-centric experience.

5. Implementation Detail’s

Frontend Streamlit UI3:

- A header displays the title of the app.
- A dropdown (st.selectbox) allows the user to pick a movie.
- A button (st.button) triggers the recommendation function.
- Five columns display each recommended movie's name and poster.

Backend:

- recommend() function takes the selected movie title, finds its index, sorts similarity scores, and returns the top 5 movies.
- fetch_poster() uses the TMDB API to get the poster path and construct the full image URL.

File Structure:

- app.py: Main logic of the app
- movie_list.pkl: Stores the movie dataframe
- similarity.pkl: Stores the similarity matrix
- setup.py: Setup configuration
- README.md, requirements.txt: Documentation and dependencies

6. Machine Learning and Data Logic

The recommendation engine relies on a precomputed similarity matrix. Here9s how the ML aspect works:

- Dataset Used:

  A movie dataset containing movie titles and metadata such as genres, tags, and overviews.

- Feature Extraction:

  Textual information is vectorized using techniques like TF-IDF or Count Vectorizer.

- Similarity Calculation:

  Cosine similarity is calculated between all movies and stored in a matrix.

- Precomputations:

  The similarity matrix is saved using pickle for faster access during runtime.

Even though there is no real-time training during app use, the preprocessing and similarity computations use ML techniques.

7. TMDB API Integration

About TMDB API:

The Movie Database (TMDB) provides detailed movie information via a free API. It is a valuable tool for developers building movie-based apps.

How It's Used:

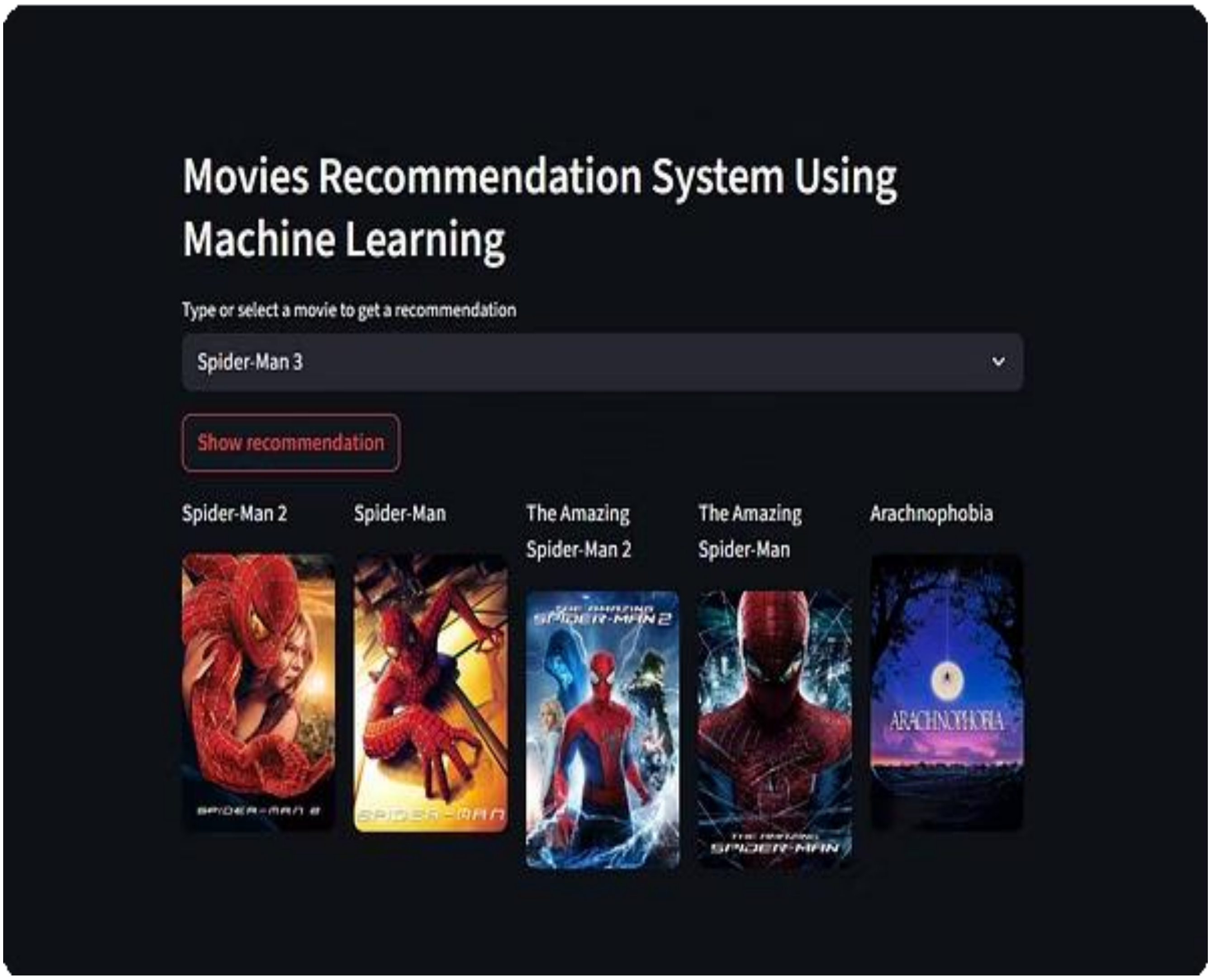
- After identifying similar movies, their unique movie_id is extracted.
- The TMDB endpoint https://api.themoviedb.org/3/movie/{movie_id} is called with an API key. • The returned JSON contains the poster path.
- The full poster URL is built and displayed using Streamlit's st.image().

This integration significantly improves the user experience by providing a visual element.

8. UI of the Recommendation System:

Stream lit App UI: Showing dropdown, button, and poster grid

These visuals help demonstrate the app9s functionality and clean interface.



Movies Recommendation System Using Machine Learning

Type or select a movie to get a recommendation

Spider-Man 3

9. Testing and Output Examples

Testing Procedure:

- Select a known movie like "Spider-Man 3"
- Click on the "Show recommendation" button
- Observe the 5 recommended movies and verify if they are related

Example Output:

Input:	Spider-Man 3
Output:	Spider-Man 2, Spider-Man, The Amazing Spider- Man, The Amazing Spider-Man 2, Arachnophobia

This confirms that the recommendation engine is working logically and returning genre-relevant results.

10. Conclusion and Future Scope

Conclusion:

The Movie Recommendation System successfully suggests relevant movie titles based on similarity measures. With a clean UI and real-time poster fetching, it enhances the decision-making experience for users. It combines machine learning logic with API-based interactivity.

Future Enhancements:

- User login and profile-based recommendations
- Rating and feedback loop to improve suggestions
- Genre filters or mood-based recommendations
- Integration with streaming platform availability

This project lays a solid foundation for intelligent recommendation systems in the entertainment domain.