Movie Recommendation System

Tarun Gangadhar Vadaparthi

Abstract

This project showcases an interactive movie recommendation system that was constructed with Streamlit. The system uses a content-based filtering technique in which cosine similarity is used to calculate similarities and feature embeddings are used to represent movies. Preprocessing cleans the metadata and creates a similarity matrix from the dataset, which is then saved as pre-computed pickle files for quick retrieval. Users can enter a movie title into the deployed application to get the top-N recommended films with comparable features. The system can be deployed on platforms like Heroku or Streamlit Cloud and is user-friendly and lightweight. In order to enhance the quality of recommendations, future extensions will incorporate hybrid models, collaborative filtering, and enriched metadata like cast, crew, and genres.

1 Introduction

Because they allow for customized suggestions and improve user engagement, recommendation systems have emerged as a crucial part of contemporary digital platforms. By customizing content to each user's preferences, recommender systems enhance the user experience across all platforms, including streaming and e-commerce. In the domain of movies, platforms such as Netflix and Amazon Prime rely heavily on recommender systems to surface relevant titles and retain user attention.

2 Motivation

Users find it more difficult to manually find pertinent titles as the amount of movie content available online keeps increasing. To capture user-specific interests, conventional browsing or genre-based searches are insufficient. The creation of a content-based movie recommendation system that uses similarity metrics between film attributes to recommend appropriate titles was spurred by this. Implementing a lightweight, interpretable solution that strikes a balance between usability and efficiency is the main goal of this project. It also lays the groundwork for future extensions like collaborative and hybrid filtering techniques.

3 Dataset Description and Preprocessing

3.1 Dataset

The Movie Database (TMDB) is the source of the dataset used in this project. It includes metadata about thousands of films, such as release year, genres, titles, and overview text. Because it records both textual and categorical data pertinent to user preferences, TMDB offers a wealth of features that can be used to create content-based recommendation systems.

3.2 Preprocessing

The raw dataset underwent multiple preprocessing steps to prepare it for recommendation:

- **Cleaning:** To guarantee data consistency, duplicate or missing records were eliminated.
- **Feature Selection:** Relevant features were extracted, including *title*, *genres*, *keywords*, and *overview*.
- **P**rocessing of Text: Overview text and keywords were converted to lowercase, tokenized, and stripped of stop words and punctuation.
- **Vectorization:** To numerically encode textual features, a Bag-of-Words (BoW) or Term Frequency–Inverse Document Frequency (TF–IDF) representation was developed.

- **T**he Matrix of Similarities: For effective retrieval during runtime, cosine similarity between movie feature vectors was calculated and the resulting similarity matrix was saved in pre-computed pickle files (`movie_list.pkl` and `similarity.pkl`).

These preprocessing procedures guarantee that the recommendation system has a structured representation of films that can be quickly retrieved using similarity.

4 Methodology and System Design

4.1 System Overview

Movies are represented by feature vectors that are extracted from their metadata in the content-based filtering method used by the Movie Recommendation System. Three phases make up the core system: interactive user interface, similarity computation, and data preprocessing. The system can produce recommendations quickly and in real time thanks to preprocessed data and a precomputed similarity matrix.

4.2 Recommendation Algorithm

Cosine similarity serves as the foundation for the recommendation process:

1. The user-selected movie is mapped to the similarity matrix's index. The corresponding row of similarity scores, which represent similarity values with all other movies in the dataset, is extracted.
2. The top-N films with the highest similarity are returned as recommendations after the scores are sorted in descending order.

By using this approach, recommendations are guaranteed to be comprehensible and directly connected to feature overlaps like genre, keywords, and text summary.

4.3 Interactive Application

Streamlit is used to implement the system as a web application. Through the interface, users can:

- Enter the title of a movie or choose one.
- See a ranking of suggested films.
- Engage with a straightforward and lightweight user interface that necessitates little client-side processing.

4.4 Deployment

The system is made to be deployed on cloud platforms like Heroku or Streamlit Cloud in order to guarantee accessibility. The runtime environment is defined by configuration files like `Procfile`, `setup.sh`, and `requirements.txt`, which allow for scalability and smooth deployment.

5 Implementation Details

The project is implemented as a Python-based application that supports preprocessing scripts and has a Streamlit front end. The following are the main elements of the implementation:

5.1 Application Code

`app.py` contains the main application logic. A Streamlit interface that takes user input in the form of a movie title is initialized by this script. It retrieves and displays the top-N suggested films using cosine similarity after loading the pre-computed datasets from pickle files. The functions in the modular code are devoted to:

- Preprocessed data is being loaded from `movie_list.pkl` and `similarity.pkl`.
- Finding the user-selected movie's index in the dataset.
- Getting and classifying similarity scores.
- Streamlit interface displaying suggestions.

5.2 Preprocessing

The preprocessing logic is contained in the Jupyter Notebook `DataPreprocessing.ipynb`. This notebook:

- Cleans and prepares the TMDB dataset.
- Extracts features such as title, genres, keywords, and overview.
- Vectorizes textual features into numerical representations.
- Computes cosine similarity between all movies.
- Stores results in two pickle files: `movie_list.pkl` (movie metadata) and `similarity.pkl` (cosine similarity matrix).

5.3 Configuration Files

Deployment to cloud platforms requires additional configuration:

- `requirements.txt`: specifies Python package dependencies such as `streamlit`, `pandas`, and `scikit-learn`.
- `Procfile`: defines the startup command for Heroku to run the Streamlit application.
- `setup.sh`: contains environment setup commands for deployment.

5.4 Pickle Files

Two serialized objects support fast runtime recommendations:

- `movie_list.pkl`: stores processed movie metadata used to map titles to indices.
- `similarity.pkl`: stores the cosine similarity matrix for efficient retrieval of recommendations.

6 Results and Discussion

6.1 System Output

The Movie Recommendation System successfully generates recommendations based on user input. When a movie title is entered into the Streamlit interface, the system retrieves the corresponding index from the `movie_list.pkl` file and uses the precomputed `similarity.pkl` matrix to return a ranked list of top-N similar movies. Recommendations are displayed instantly on the user interface.

6.2 Example

Figure 1 illustrates the output of the Movie Recommendation System. In this case, the user selects a movie title, and the system instantly returns a ranked list of similar movies through the Streamlit interface.

These results illustrate the ability of cosine similarity to surface movies with overlapping genres, themes, and narrative styles.

6.3 Performance

The system operates in real time, with response times under one second for typical queries. This is due to the precomputation of the similarity matrix, which eliminates the need for repeated vectorization or recalculation during runtime.

6.4 Discussion

While the system is efficient and produces reasonable recommendations, its performance is constrained by the simplicity of the content-based approach. Recommendations are limited to metadata similarity and may not fully capture nuanced user preferences. For example, movies with different narratives but shared genres may be incorrectly recommended as ”similar.” Despite these limitations, the system provides a solid baseline and can be extended with collaborative filtering or hybrid techniques for improved personalization.

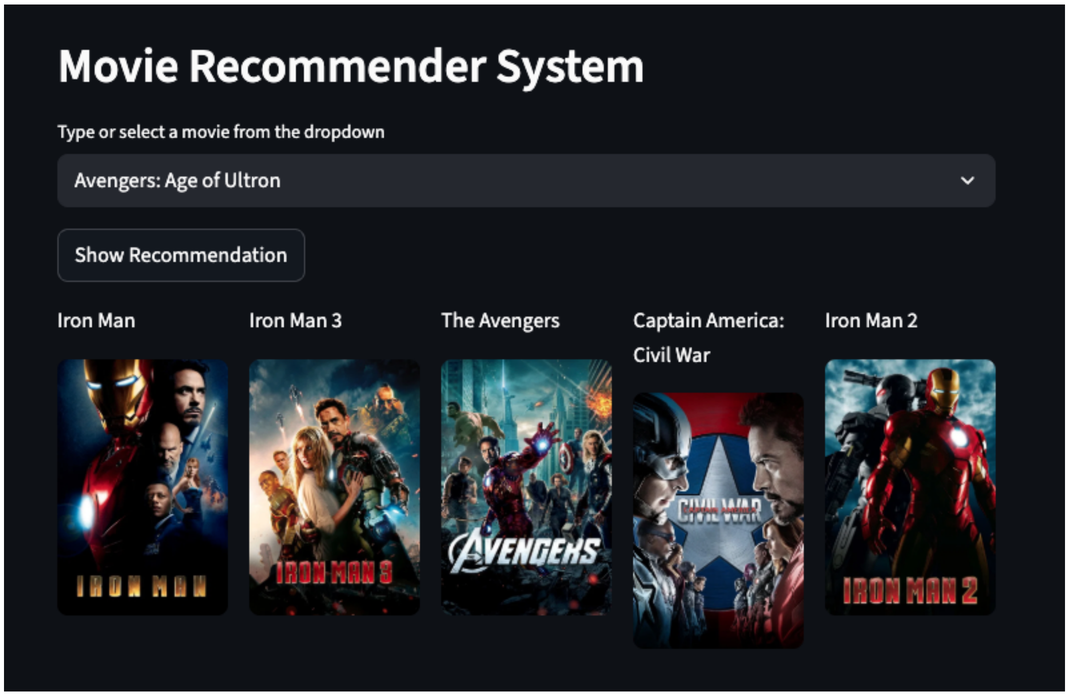

Figure 1: Sample recommendation results for a selected movie.

7 Limitations and Future Work

While the system produces relevant recommendations, it is limited to a content-based approach. Recommendations depend entirely on metadata similarity and do not incorporate actual user behavior. As a result, personalization is limited. Future work will focus on:

- Incorporating collaborative filtering to learn from user ratings.
- Designing a hybrid system combining content-based and collaborative methods.
- Enriching the dataset with additional metadata such as cast, crew, and user reviews.

8 Conclusion

This project demonstrates the implementation of a content-based movie recommendation system using the TMDB dataset. The system preprocesses metadata, computes cosine similarity, and deploys a lightweight, interactive interface with Streamlit. The application generates real-time recommendations and can be easily deployed to cloud platforms such as Heroku. Although limited by its reliance on metadata alone, the system provides a strong baseline and offers multiple pathways for future enhancement.