Movie Recommendation System: Project Report

Introduction

In the vast landscape of digital content, recommendation systems have become indispensable tools for personalizing user experiences and combating information overload. This report details the development of a Movie Recommendation System designed to suggest films to users based on their expressed preferences. The system employs a hybrid approach, combining the strengths of collaborative filtering and content-based filtering, and is presented through an interactive web application. The objective is to provide relevant and diverse movie suggestions, enhancing user engagement with available cinematic content.

Abstract

This project focuses on building a hybrid movie recommendation system using the MovieLens ml-latest-small dataset. The system integrates two primary recommendation techniques: item-item collaborative filtering, which identifies movies liked by users with similar tastes, and content-based filtering, which recommends movies based on genre similarities. Data preprocessing involves cleaning movie titles, parsing genres, and constructing a user-item rating matrix. The core recommendation logic combines scores from both methods with an adjustable weight, allowing for flexible prioritization. The entire system is encapsulated within an intuitive web application developed using Streamlit, enabling users to interactively select a movie and receive personalized top-5 recommendations. The report outlines the architecture, implementation steps, and key outcomes of this recommendation engine.

Tools Used

The development of this Movie Recommendation System leveraged a suite of powerful Python libraries and frameworks:

- **Python 3.x:** The foundational programming language for all logic and application development.
- **Pandas:** Utilized extensively for data loading, cleaning, manipulation, and the creation of the user-item matrix.
- **NumPy:** Provides essential numerical computing capabilities, particularly for handling arrays and matrix operations involved in similarity calculations.
- **Scikit-learn (sklearn):** A comprehensive machine learning library used for:
    - `TfidfVectorizer`: To convert movie genres into numerical feature vectors for content-based filtering.
    - `cosine_similarity`: To compute similarity scores between movies based on both genre features and user ratings.
- **Streamlit:** An open-source app framework used to rapidly build and deploy the interactive web-based user interface, allowing for seamless user interaction with the recommendation engine.

Steps Involved in Building the Project

The project was developed following a structured approach, broken down into distinct phases:

1. **Environment Setup and Dependency Installation:**
    - A Python virtual environment was created, and essential libraries (`pandas`, `numpy`, `scikit-learn`, `streamlit`) were installed.
2. **Data Acquisition and Initial Exploration:**
    - The `MovieLens ml-latest-small` dataset (`movies.csv`, `ratings.csv`) was downloaded and loaded into Pandas DataFrames.
    - A Jupyter Notebook facilitated initial data inspection and experimentation.
3. **Data Preprocessing and Feature Engineering:**
    - Movie titles were cleaned, and genres were transformed for vectorization.
    - A sparse user-item matrix was created from ratings, with missing values filled.
4. **Model Development (Similarity Calculation):**
    - **Content-Based Filtering:** TF-IDF and cosine similarity were used to calculate movie similarities based on genre.
    - **Collaborative Filtering:** Cosine similarity was applied to a transposed user-item matrix to find movie similarities based on user ratings.
5. **Hybrid Recommendation Logic Implementation:**
    - A `get_hybrid_recommendations` function was developed to combine content-based and collaborative scores using an adjustable `genre_weight` parameter.
    - The function returns the top 5 unique recommended movie titles.
6. **Streamlit User Interface Development:**
    - The `streamlit_app.py` script was created, utilizing `@st.cache_resource` for efficient data loading.
    - Streamlit widgets (`st.selectbox`, `st.slider`, `st.button`) were implemented for user interaction and to display recommendations.
7. **Testing and Documentation:**
    - The Streamlit application was thoroughly tested locally.
    - A `requirements.txt` file was generated, and a comprehensive `README.md` was prepared for project documentation.

Conclusion

This project successfully delivered a functional hybrid movie recommendation system, demonstrating the effective integration of machine learning techniques with an interactive user interface. By combining collaborative and content-based filtering, the system offers robust and diverse movie suggestions, addressing the common challenge of information overload in digital media. The modular design, implemented using Python, Pandas, Scikit-learn, and Streamlit, ensures maintainability and provides a solid foundation for future enhancements. While the current implementation provides a strong proof-of-concept, future work could explore more advanced collaborative filtering algorithms (e.g., matrix factorization), incorporate sentiment analysis from movie reviews, and optimize for scalability with larger datasets. This project serves as a practical example of applying data science principles to create a personalized user experience.