Maham Azam

Movie Recommendation System

Abstract

This project develops a Movie Recommendation System that suggests movies based on user preferences. The system uses a content-based filtering approach, leveraging movie metadata to find similarities and provide personalized recommendations.

* **Objective**

The goal is to implement a system that:

1. Takes the user's favorite movie as input.
2. Finds similar movies based on specific features such as genres, keywords, tagline, cast, and director.
3. Recommends a list of up to 30 movies ranked by their similarity to the selected movie.

* **Steps in the Recommendation System**

1. **Data Preprocessing:**
   * + Loaded the movie dataset from a CSV file.
     + Selected five key features for recommendations:
       - Genres
       - Keywords
       - Tagline
       - Cast
       - Director
       - Handled missing values by replacing null entries with empty strings.
2. **Feature Engineering:**

* Combined the selected features into a single string for each movie, creating a unified representation for similarity calculations.
* "Action Adventure | hero save day | Courage is immortal | Robert Downey Jr. | Jon Favreau"

1. **Vectorization:**

* : Used TF-IDF Vectorizer to transform textual data into numerical feature vectors.
* This step converts the movie metadata into a machine-readable format, capturing the importance of terms within the combined features.

1. **Similarity Computation:**

* Calculated cosine similarity between feature vectors of movies to measure how closely related they are.
* Cosine similarity ensures that movies with similar metadata are given higher scores.

1. **User Input and Matching:**

* Allowed the user to input their favorite movie title.
* Used difflib to find the closest match to the user-provided title, accounting for potential spelling errors.

1. **Recommendation Generation:**

* Identified the index of the user-selected movie in the dataset.
* Retrieved a list of movies sorted by similarity scores.
* Displayed the top 30 similar movies to the user.
* **Core Functionalities**
* **Dynamic Recommendations:** Provides tailored suggestions based on user input.
* **Robust Matching**: Handles user input errors (e.g., typos) through close match detection.
* **Content-Based Filtering**: Focuses solely on metadata, ensuring relevant and personalized recommendations.
* **Technologies Used**
* Programming Language: Python
* Libraries:
* Pandas : For data manipulation.
* Numpy : For numerical operations.
* Sklearn: For TF-IDF vectorization and similarity calculations.
* Difflib : For finding close matches.
* **Sample Input and Output:**
* **Input:**

Enter your favorite movie name: "Iron Man"

* **Output:**

Movies suggested for you:

1. The Avengers

2. Iron Man 2

3. Iron Man 3

4. Thor

5. Captain America: The First Avenger

...

* **Conclusion**

The Movie Recommendation System demonstrates an effective approach to content-based filtering. By focusing on movie metadata, it provides relevant suggestions tailored to user preferences.

* **Future Enhancements:**
* Incorporate collaborative filtering for a hybrid approach.
* Integrate user feedback to improve recommendation quality.
* Add additional features like trailers, reviews, and ratings for an enhanced user experience.