**Movie Recommendation System**

**Team Members**

|  |
| --- |
| **MennaTallah Srour Mohamed** |
| **Yassmin yousery imam Muhammad** |
| **Hassan Elsayed Abdelfattah** |
| **Khalid Omar Sedik** |
| **Noha Yahia Ibrahim Elsaadawy** |

1. **General Overview**

**Project Name:** MovieRecommendation System  
**Type:** Web-based Application  
**Technologies:** Python, Flask, Scikit-learn, Pandas, NumPy

This project delivers a movie recommendation system that uses both Content-Based Filtering and Collaborative Filtering techniques. It is designed as a lightweight Flask web application that provides users with personalized movie suggestions based on either the attributes of movies they like or based on the preferences of similar users.

**Features**

* **Search by Title or Tagline:** Users can input a full movie title or descriptive tags like “dystopian future” or “romantic thriller” to retrieve similar titles.
* **Content-Based Recommendations:** Matches based on genres, plot keywords, cast, and director.
* **Ranked by IMDb Score:** Ensures quality by filtering and ranking suggestions using IMDb scores.
* **Simple UI:** A Flask web interface allows for an easy-to-use experience.
* **Collaborative Filtering Extension:** Recommends movies using user-user similarity based on ratings.

**Tech Stack**

* **Languages:** Python
* **Libraries:** Flask, Pandas, NumPy, Scikit-learn
* **Web Framework:** Flask
* **Recommendation Algorithms:** TF-IDF, Cosine Similarity, KNN

**Datasets Used**

* **Movie Recommendations Dataset (**[**Movie Recommendations**](https://www.kaggle.com/datasets/sreenathkk/movie-recommendations)**):** Detailed info including genres, cast, plot, and taglines
* **Movie Metadata Dataset (**[**movie\_metadata.csv**](https://www.kaggle.com/datasets/brtej1/movie-metadata-csv)): Rich metadata for content-based filtering
* **MovieLens Dataset (**[**MovieLens | GroupLens**](https://grouplens.org/datasets/movielens/)**):** Ratings and titles used for collaborative filtering

1. **Metadata Preprocessing Phase**

Before building the recommendation engines, the project conducts thorough data cleaning and enrichment to enhance recommendation quality.

**Data Cleaning Goals**

* Eliminate irrelevant columns (e.g., social media likes, budget).
* Normalize textual data (e.g., genres, keywords).
* Handle and fill missing values through:
  + Mean/Mode Imputation for numeric and categorical columns.
  + OMDb API for critical fields (director, year, plot, etc.).
  + Manual cleanup of non-standard characters and string formats.

**Steps Taken**

1. **Initial Exploration:**
   * Loaded movie\_metadata.csv using Pandas.
   * Removed noisy columns.
2. **Text Cleanup:**
   * Fixed malformed strings in titles.
   * Converted multi-value text fields into lists (e.g., genres).
3. **Handling Missing Values:**
   * Used OMDb API to fill important missing fields.
   * Added request delay to avoid hitting API rate limits.
4. **Final Output:**
   * Saved enriched dataset as updated\_movies.csv.

**Tools Used**

* **Libraries:** Pandas, requests, csv, time
* **API:** OMDb (for director, plot, language, year, etc.)

1. **Content-Based Filtering Phase**

Using the cleaned dataset, this phase applies Content-Based Filtering via TF-IDF Vectorization and Cosine Similarity.

**How It Works**

1. **User Input:**Users input a movie title (e.g., Pirates of the Caribbean: At World's End") or a tagline keyword (e.g., "At the end of the world, the adventure begins.")
2. **Recommendation Strategies:**Users can select recommendation types:
   * By Genre
   * By Actor Name
   * By Director Name
   * By Plot Keywords
   * Or simply return the Top 5 highest IMDb-rated related movies
3. **Algorithm Details:**
   * The chosen feature is transformed using TF-IDF.
   * Cosine Similarity is calculated between vectors.
   * Top 15 similar movies are retrieved and filtered using a score threshold (IMDb ≥ 7.0).
4. **Evaluation:**
   * Conducted on test examples using: Precision ,Recall ,and F1 Score

**Tools Used**

* Python
* Pandas, NumPy
* Scikit-learn (TF-IDF, cosine similarity, metrics)

1. **Collaborative Filtering Phase**

This phase introduces User-Based Collaborative Filtering using the K-Nearest Neighbors (KNN) algorithm to recommend movies based on similar users’ preferences.

**How It Works**

1. **Data Preparation:**
   * Loaded ratings.csv and movies.csv.
   * Merged movie titles with rating records.
   * Split into train/test sets.
2. **Matrix Construction:**
   * Mapped user IDs and movie IDs to indices.
   * Created a sparse user-item rating matrix.
3. **Model Training:**
   * Used Nearest Neighbors with cosine similarity.
   * Found similar users using KNN (adjustable k-values).
4. **Recommendation Logic:**

For a given user:

* + Find k most similar users.
  + Recommend highly rated movies the target user hasn’t seen.

1. **Evaluation:**
   * Measured using Precision@5 for different k-values.
   * Reported top-N recommendations per user.
2. **Future Directions**

* **Incorporate Item-Based Collaborative Filtering**
* **Explore Hybrid Approaches blending both filtering methods:** Merging content-based and collaborative outputs could improve robustness.
* **Migrate to scalable frameworks like LightFM or Surprise**
* **Cold Start Solutions:** Consider integrating user registration and preference capturing at first interaction.
* **Performance Scaling:** Move to more scalable libraries or GPU-based models for real-time recommendations.
* **User Feedback Loop:** Include feedback features to retrain models with evolving user preferences.

1. **Conclusion**

This Movie Recommendation System successfully combines:

* **Preprocessing for Clean Input Data**
* **Content-Based Recommendations using NLP.**
* **Collaborative Filtering via KNN**
* **Evaluation of standard metrics**

The current system is functional, modular, and easily extensible. With further enhancements in scalability, hybrid modeling, and user interaction, it can evolve into a fully-fledged intelligent recommendation engine.