REC-IT RALPH
Movie Recommender System

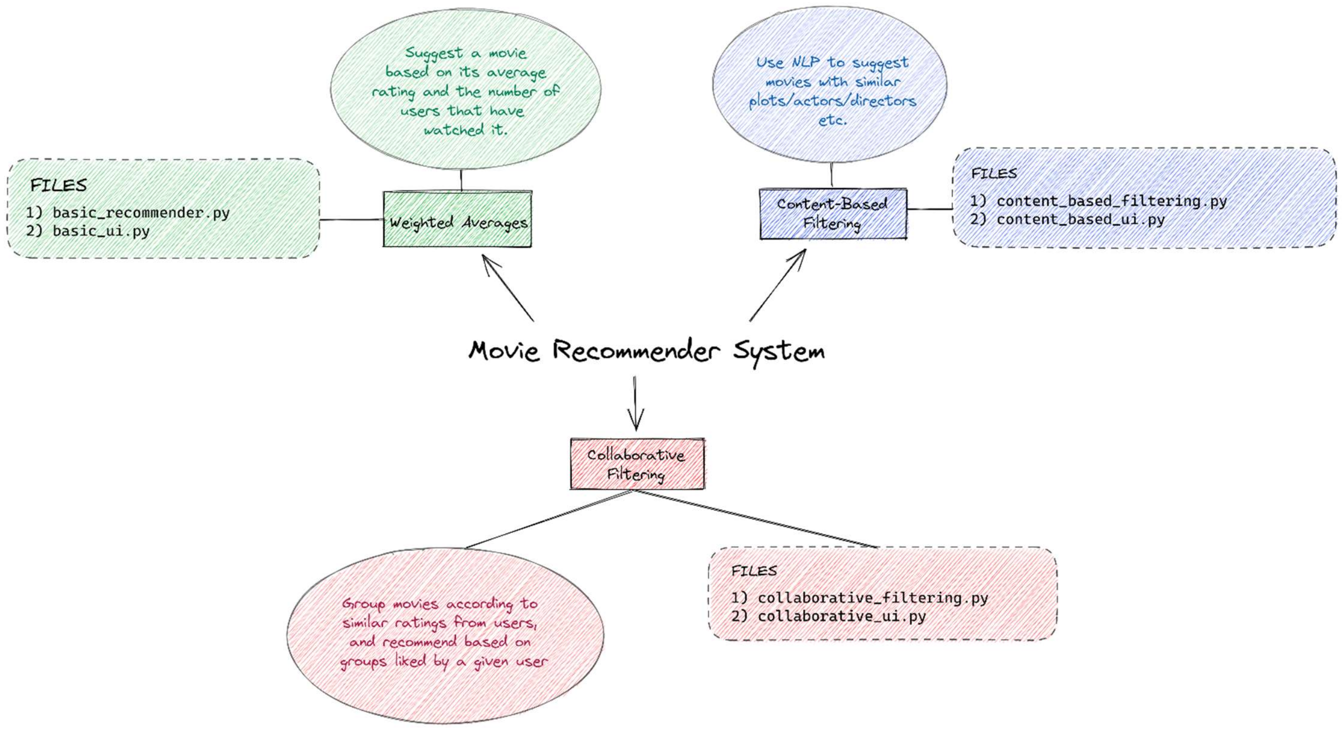
Overview

This project aims to demonstrate various movie recommendation algorithms used by major platforms like Netflix and Spotify.
For educational purposes, the project shall begin with the most basic algorithms or techniques used in a recommendation, and progressively build upon them to implement more advanced and sophisticated techniques.

Finally, the project will enable a user to get movie recommendations through different methods. For example, the user can view which movies are the most popular and highly rated in general or they can input their preferred genres or favorite movies or choose a set of movies and rate them, and can get personalized recommendations based on these inputs.

Problem statement:

To demonstrate the different techniques through which a user can be recommended a movie based on their preference for different genres, actors, directors etc.



Suggest a movie based on its average rating and the number of users that have watched it.
Use NLP to suggest movies with similar plots/actors/directors etc.
FILES
1) basic_recommender.py
2) basic_ui.py
Weighted Averages
Content-Based Filtering
FILES
1) content_based_filtering.py
2) content_based_ui.py
Movie Recommender System
Collaborative Filtering
Group movies according to similar ratings from users, and recommend based on groups liked by a given user
FILES
1) collaborative_filtering.py
2) collaborative_ui.py

How will it work?

The UI is meant to be as self-descriptive as possible. The main webpage will be divided into multiple segments, with each segment corresponding to a particular algorithm for movie recommendation, and each segment having different input fields.
The end-user may choose to pick from a general selection of popular movies to watch, or they may enter some information about their preferred genres or favorite movie(s) to get personalized recommendations.

Algorithms Used

- Sorting algorithms for average ratings for 10k movies.
- Natural Language Processing using TF-IDF and Pearson similarity for keyword searching.
- Statistical analysis to identify similar movies according to users' ratings.

Folder Structure

- `Datasets` - contains all the required data in csv format.
  Web scraping and data processing has been used to produce required information from existing datasets on Kaggle.
  - *average_ratings.csv*
  - *links.csv*
  - *movies_features.csv*
  - *ratings_sorted_movies.csv*
  - *ratings.csv*

- `JupyterNotebooks` - contains Python code in Jupyter notebooks format.
  These notebooks are not required for the app to run, but are made available for reference as all the data processing and web scraping was done using Jupyter notebook environment.
  - BasicRecommender
  - CollaborativeFiltering
  - ContentBasedFiltering

- `Posters` - contains posters for all movies, along with website favicon and other required images.
  Movies' posters have been web scraped from IMDB using Python.

- `Screenshots` - contains screenshots and demo of the deployed project webpage.

- `basic_recommender.py` `collaborative_filtering.py` `content_based_filtering.py` - contain Python classes for different recommendation algorithms.
- `basic_ui.py` `collaborative_ui.py` `content_based_ui.py` - contain Python classes for frontend elements for different recommendation algorithms.
- `main.py` - driver script to run the application.
- `posters_printer.py` - contains Python class to render posters for a given list of movies, along with their IMDB links.
- `watchlist.py` - contains Python class to maintain a watchlist of the user's chosen movies.

Development Timeline

The development plan of the project is as follows:

- Week 1: Ideation and Documentation
- Week 2: Getting the script(backend) ready and working
- Week 3: Integration with frontend + addition of new algorithms
- Week 4: Final touches and submission

How do we measure success?

The final application will be tested with multiple users via a survey form circulated in my university. The project's success will depend upon reviews from more than 2000 users.