What to watch tonight (WTWT) ?

A movie recommendation system

Karun Dawadi

1001660099 (kxd0099)

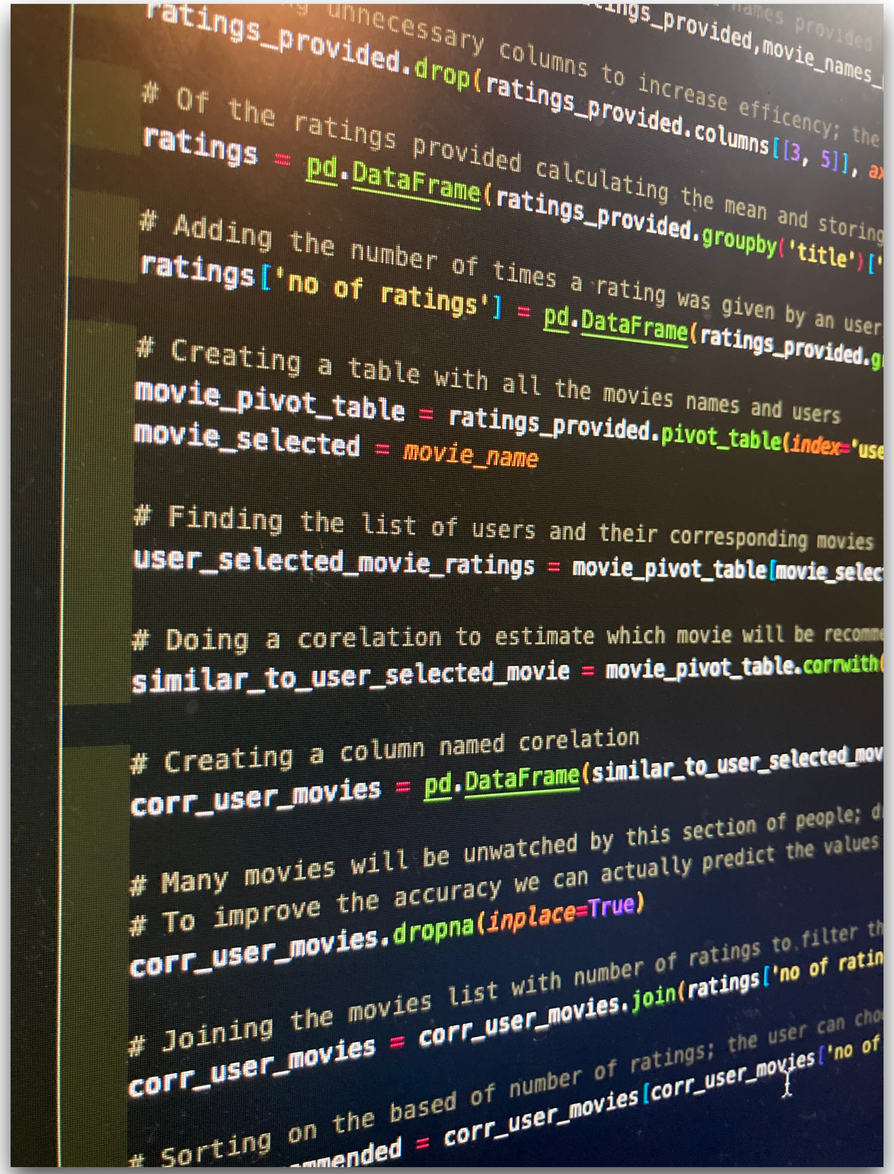
karun.dawadi@mavs.uta.edu

What is WTWT?

- Movie recommendation system
- Designed to save time
- Platforms do not have all the contents available
- Recommend movies on basis of users choice

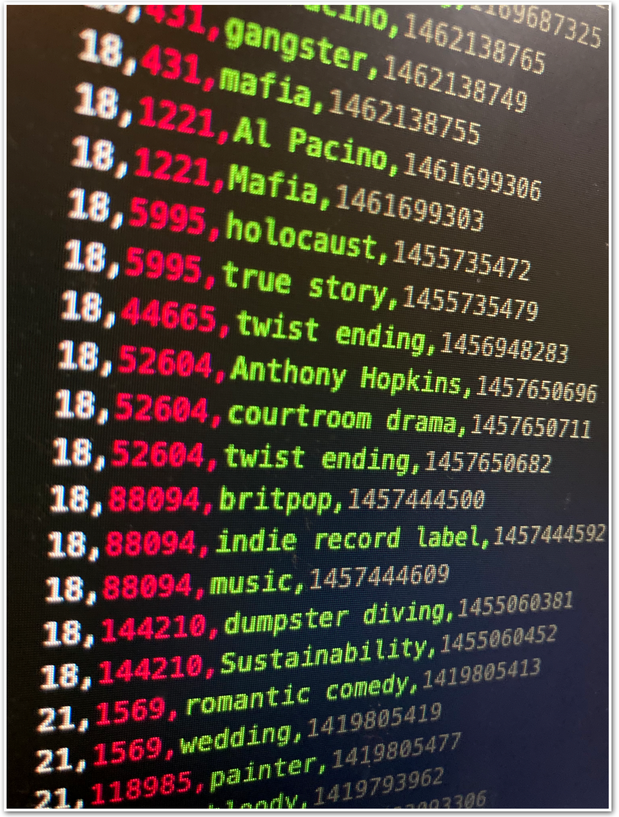
Tools used

- **Python**
  - Pandas , Flask, Flask-cors, Numpy
- **Node.js (npm)**
  - React, Material UI, Axios, Bootstrap
- **Dataset used**
  - GroupLens movie-lens ml-latest small
- **Version control**
  - GIT (github.com/karundawadi/WTWT)



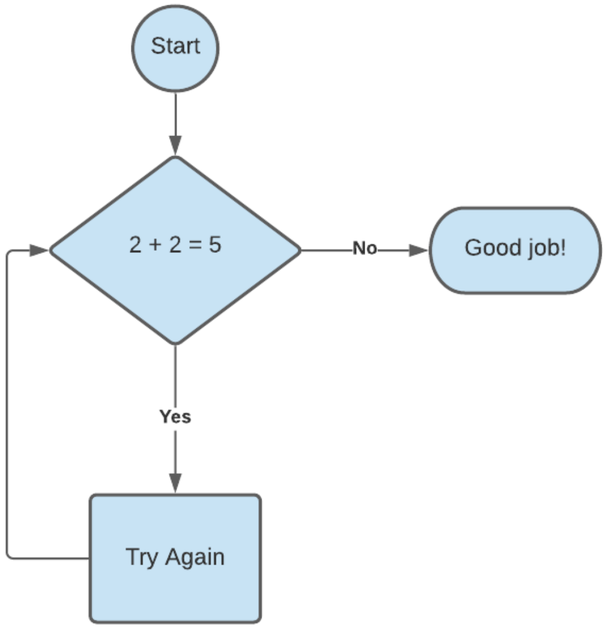
Background

- **Correlation** - Measure of the extent to which two variables are linearly related
- **Ratings to consider** - Number of people who have rated the suggested movie
- **<u>Recommender systems</u>**
  - Content based methods
    - Works with the data that is provided by the user
  - Collaborative filtering methods
    - Mimics user-to-user recommendation
  - Hybrid methods
    - Combine methods of both methods



How does WTWT work ?

- Uses collaborative filtering
  - Mimics user-user recommendation system
    - Summary
      - It finds correlation of movies
      - Filters on the basis of number of ratings



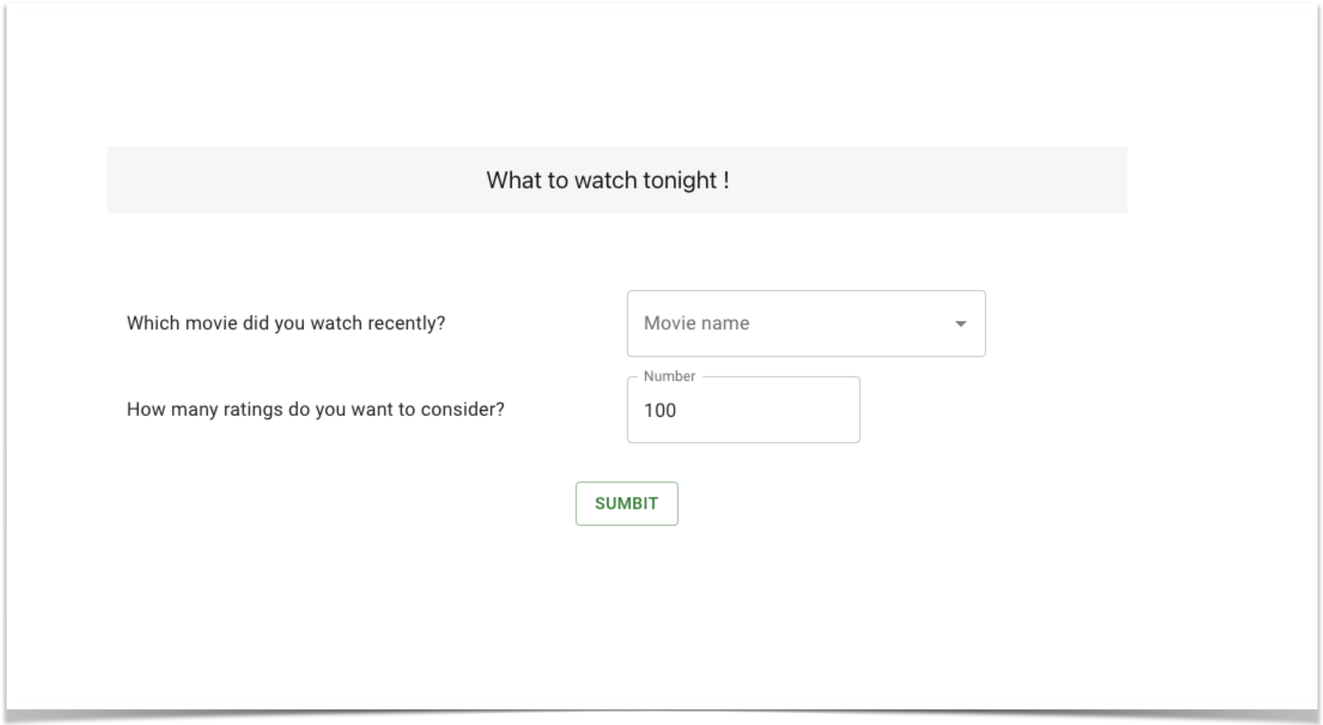
Algorithm followed by WTWT

1. Get input from front end system
2. Read the dataset of ratings and movie names
3. Merge the two dataset together
4. Create a pivot table
5. Correlate user’s movie choice with other movies
6. Filter on the basis of correlation values
7. Filter on the basis of number of ratings
8. Format data
9. Send to front end system

```python
import pandas as pd

def recommend_movies(movie_name,user_filter):
    ratings_provided = pd.read_csv("training/ratings.csv")
    movie_names_provided = pd.read_csv("training/movies.csv")

    # Merging both ratings provided and movie names provided such that we have access to the titles
    ratings_provided = pd.merge(ratings_provided,movie_names_provided,on='movieId')

    # Dropping unnecessary columns to increase efficency; the more data means more time taken
    ratings_provided.drop(ratings_provided.columns[[3, 5]], axis = 1, inplace = True)

    # Of the ratings provided calculating the mean and storing it in ratings
    ratings = pd.DataFrame(ratings_provided.groupby('title')['rating'].mean())

    # Adding the number of times a rating was given by an user; will later be used
    ratings['no of ratings'] = pd.DataFrame(ratings_provided.groupby('title')['rating'].count())

    # Creating a table with all the movies names and users
    movie_pivot_table = ratings_provided.pivot_table(index='userId',columns="title",values='rating')
    movie_selected = movie_name

    # Finding the list of users and their corresponding movies ratings
    user_selected_movie_ratings = movie_pivot_table[movie_selected]

    # Doing a corelation to estimate which movie will be recommended by a group who has watched similar movies
    similar_to_user_selected_movie = movie_pivot_table.corrwith(user_selected_movie_ratings)

    # Creating a column named corelation
    corr_user_movies = pd.DataFrame(similar_to_user_selected_movie,columns=['Corelation'])

    # Many movies will be unwatched by this section of people; dropping them
    # To improve the accuracy we can actually predict the values
    corr_user_movies.dropna(inplace=True)

    # Joining the movies list with number of ratings to filter them
    corr_user_movies = corr_user_movies.join(ratings['no of ratings'])

    # Sorting on the based of number of ratings; the user can choose the no of users ratings
    movies_recommended = corr_user_movies[corr_user_movies['no of ratings']>user_filter].sort_values('Corelation',ascending=False)

    # Data formatting
    values = movies_recommended.head(10)
    return_values = []
    for i in range(1,6):
        return_values.append(values.iloc[i].name) # Changing to array format

    # Will send top 5 movies
    return return_values
```

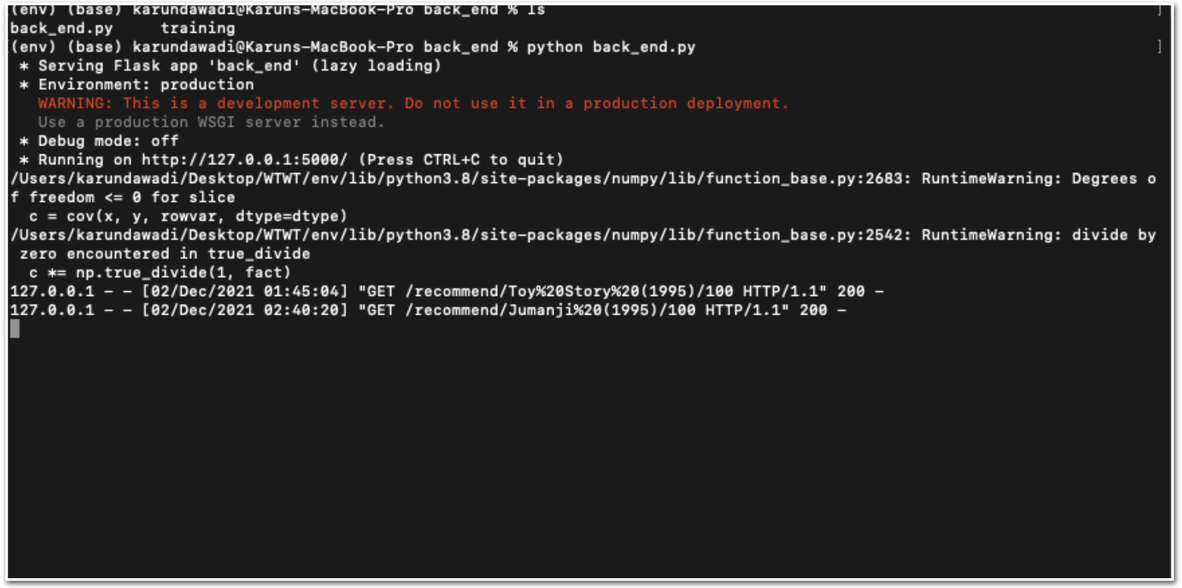
Front end system

- Created using React
  - Style using bootstrap, Material UI
  - API calls made using Axios
- Responsive



Back end system

- API calls handled by Flask
  - Requires Flask-CORS
  - API calls routed using routers



Demo

Improvements

- **Accuracy** - Limited processing power; light data set
  - Other dataset might be used to get higher accuracy (MovieLens 25M Dataset)
  - Implementation of another machine learning model to predict a rating user would provide to a movie that is not rated
  - Using test data to improve the recommendation
  - Implementation of better movie recommendation algorithm

References

Baptiste Rocca. “Introduction to Recommender Systems.” *Medium*, Towards Data Science, 2 June 2019, towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada.

Harper, F. Maxwell, and Joseph A. Konstan. “The MovieLens Datasets.” *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, 22 Dec. 2015, pp. 1–19, 10.1145/2827872.

Questions ?

karun.dawadi@mavs.uta.edu