BI-VWM

Collaborative filtering

Authors: Aydar Mannanov, Alexander Matveenko

About project

Our application will recommend you what movie to watch tonight. This application can use three types of algorithms for collaborative filtering (Pearson correlation, Kendall correlation and Spearman correlation). To get relevant movies we using user based filtration. All datasets we taking from open source files(MovieLens).

Our goals

Goals of our project is creating system that will automate recommendations of movies for user. Recommendations will be based on preferences of other users.

Inputs

Data set of real users, movies, user's reviews and links to movies. All data from MovieLens.

Outputs

List of relevant movies, links to themoviedb.org and posters of those movies.

Implementation:

Docker:

We decided to use docker because of portability(my coworker using linux, but I prefer Windows), and efficiency. We using two containers for backend/frontend and worker. To start our application you just need to clone project from github and run docker-compose up command. Backend/frontend is working on port 40001. Worker container depends on backend/frontend container.

Backend/frontend:

For our small application we decided to use Python Flask Framework for backend and frontend. To get REST API requests from our application I use Python Flask-WTF module, but for getting requests from another APIs I using default Python requests module. All html templates are locating in templates directory. To share data between worker container we using container volumes. So we can easy store text result of some container inside this volume. In root page('/') application will reset all arrays and files. In movies page('/movies') you need to add your favorite movies by adding its name from (movies.csv) (Note: you need to add at least 6 movies then you can send it to worker container), to send data to worker you need to type correlation method(pearson, spearman, kendall), then click get recommendations button. In recommendations page(/recomendations) we get 5 relevant movies in ids, then we making get request to api.themoviedb.org to get poster original name and link and then we render it.

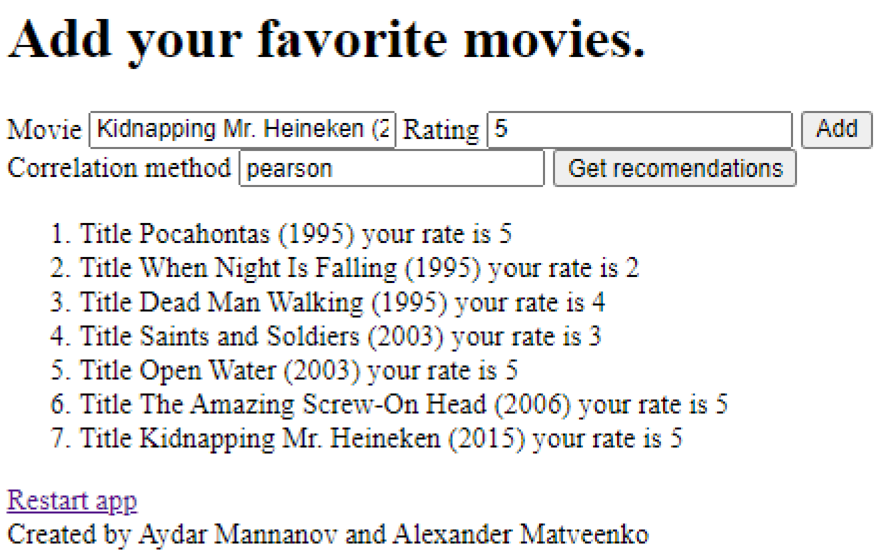
Worker:

We using NumPy, Pandas to analyze datasets. On the beginning we reading text movies from volume docker path, then we create new user in dataset of user's reviews. Then we calling function create_corr() to fin relevant movies, creating pandas table with rows of userID and columns of moviesID. We making correlation of user's reviews based on correlation method that we defined on backend. Then we multiplicate rows of user's reviews on users correlation (ref. line 67 in worker's main). Then we sum every column of every movie, sorting from highest to lowest values and return first fifty movies. And then put it two docker volume then backend takes these movies.

Experiment

We decided to compare three correlation algorithms, because of our small data sets we get identical outputs, but we have some difference with run time of algorithms.

Our input for all algorithms:



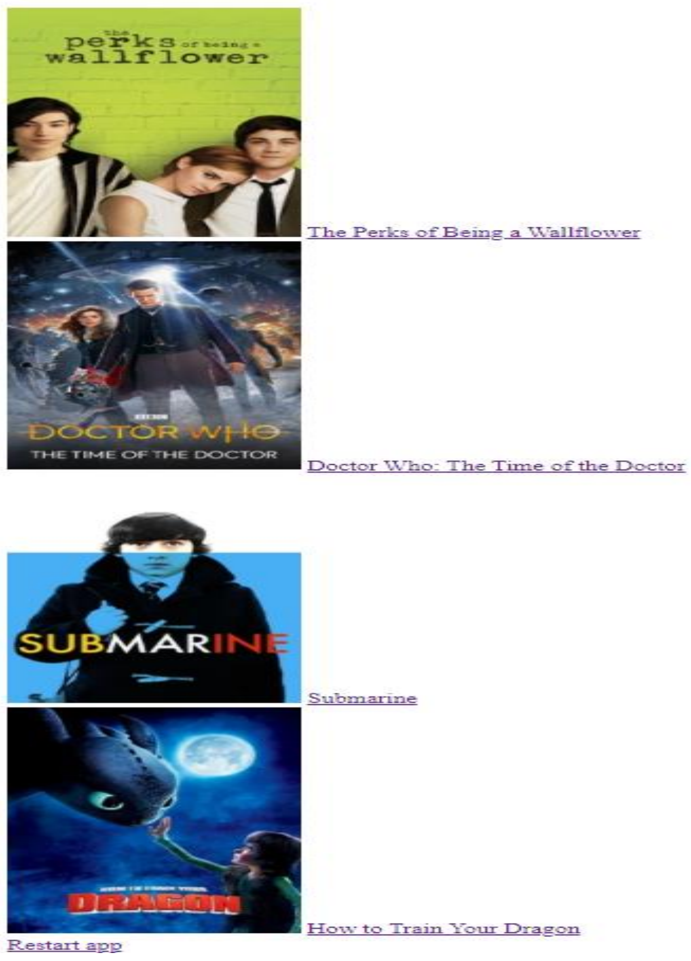
Add your favorite movies.

Movie [Kidnapping Mr. Heineken (2] Rating [5] [Add]
Correlation method [pearson] [Get recomendations]

1. Title Pocahontas (1995) your rate is 5
2. Title When Night Is Falling (1995) your rate is 2
3. Title Dead Man Walking (1995) your rate is 4
4. Title Saints and Soldiers (2003) your rate is 3
5. Title Open Water (2003) your rate is 5
6. Title The Amazing Screw-On Head (2006) your rate is 5
7. Title Kidnapping Mr. Heineken (2015) your rate is 5

Restart app
Created by Aydar Mannanov and Alexander Matveenko

Our output for all algorithms:



The Perks of Being a Wallflower
Doctor Who: The Time of the Doctor
Submarine
How to Train Your Dragon
Restart app

Runtimes:

Spearman:	59.18887 seconds
Pearson:	8.77232 seconds
Kendall:	179.68774 seconds

Unfortunately my PC can’t work with larger datasets, even so We have some results. Here we can see best algorithm to analyze large dataset.

Aydar’s conclusion:

In this project I got first experience with working with science oriented python modules. And now I also understand some basic web algorithms. I decided to use all correlation algorithms because, this python module Pandas easy to use and I was interested to compare them.

Alexander’s conclusion:

As a student from Knowledge engineering, It was very interesting for me to applicate my knowledge from BI-VZD subject on this project, and I really happy with it, because I always happy to get knowledge. In summer time, I will try to code item-based filtration. During coding this project I found many interesting movies.