Mo - a movie recommendation system

Zhendong Cao(zhendong.cao@yale.edu)
Shiying Xu(shiying.xu@yale.edu)

How to run

Demo

visit http://104.154.252.35:3000/ to view live demo

Directories

dir	description	package required
`/api_server`	API server responsible for calculating and return recommendations according to the list of movies received(Python Flask)	numpy, scipy, pandas, flask
`/app`	web server(Node.js) and web interface(React.js)	npm, node, React.js, redux, redux-saga, lodash, etc
`/process.ipynb`	python notebook experimentally dealing with the data	Jupyter notebook
`/movie_metadata.csv`	raw data	

Commands required to run

The package installing maybe painfully complicated and time-wasting. Again, please visit http://104.154.252.35:3000/ for a live demo

api_server

make sure all required packages(numpy, scipy, pandas, flask) have been installed, and the api server is on, then run:

1. `$ cd ./api_server`
2. `$ python server.py`

app

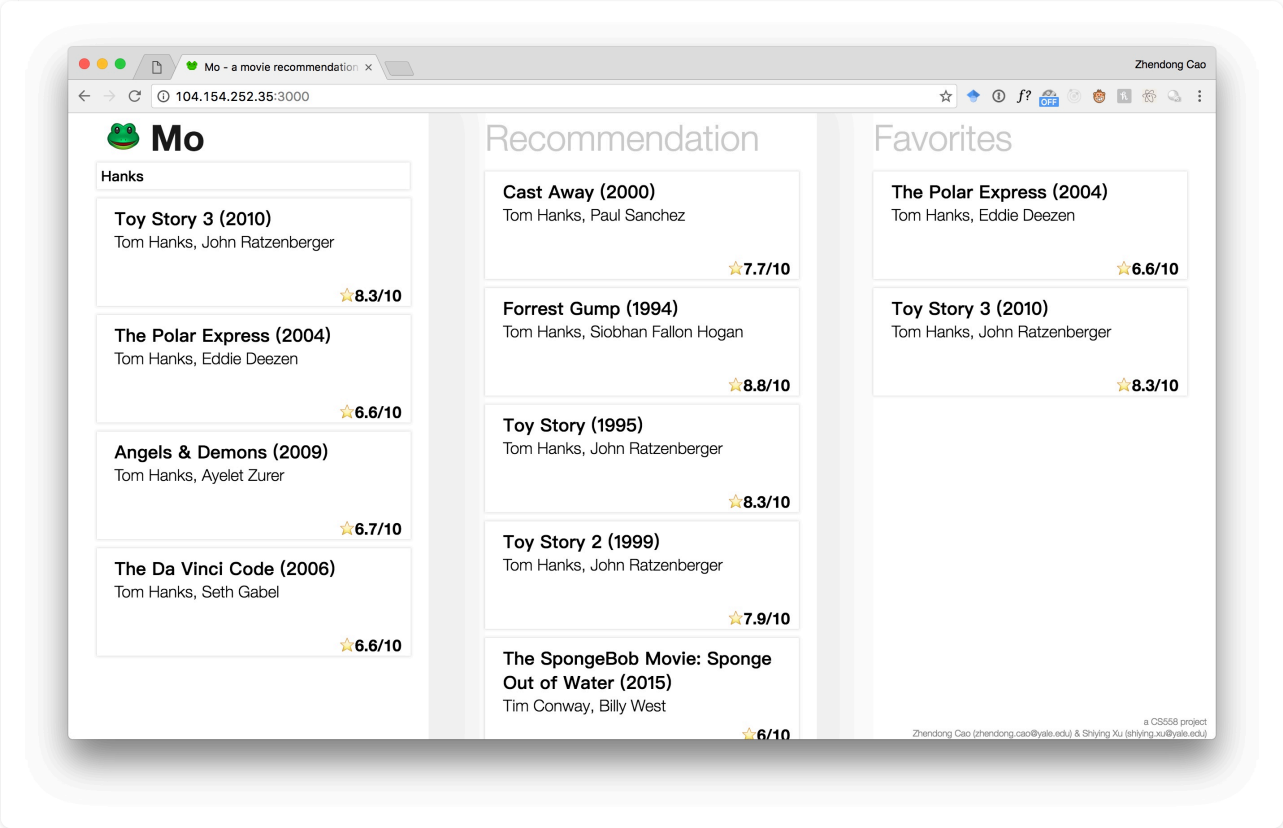
open another terminal, make sure `npm` and `nodejs` have been installed, and the api server is on, then run:

1. `$ cd ./app`
2. `$ npm install`
3. `$ npm npm start -- --release`

What it does

You can type search query in the search box to search for movies and the suggestions will appear instantly below the search box:

Then click on the movie you like to add to favorites, which are listed in the right column, and click once more to remove it from favorites, and the recommendation results will display in the middle column.



All cards are clickable, which means you can add/remove a movie to/from the favorites by clicking on all the cards visible.

what is interesting or important

The most interesting part is how to calculate the distance between different movies, or how to define the similarity, what weights should be given on different factors. A slight change can result in a complete different result.

Here is my similarity defination:

```
similarity(m1, m2):
    return director_weight * (m1_director == m2_director)
           + color_weight * (m1_color == m2_color)
           + actor_weight * len(set(m1_actor) & set(m2_actor))
           + genres_weight * len(set(m1_genre) & set(m2_genre))
           + plot_weight * len(set(m1_plot) & set(m2_plot))
           + language_weight * (m1_lenguage == m2_language)
           + country_weight * (m1_country == m2_country)
           + metrics_weight *
             (1 + cosine_similarity(m1_metrics, m2_metrics))
```

`m_metrics` includes `num_critic_for_reviews`, `duration`, `gross`, `num_voted_users`, `num_user_for_reviews`, `title_year`, `imdb_score`, and they are normalized.

The weights are:

```
color_weight = 0.1
director_weight = 3
actor_weight = 3
genre_weight = 0.2
plot_weight = 0.2
language_weight = 0.1
country_weight = 0.1
metrics_weight = 2
```

After the process of trying and tuning the weights, I find that giving director and actor more weights are more resonable and indeed results better results.

The results are sorted based on the similarity value with each items in the list of favorites.