SPRING 22

Electrical & Computer Engineering & Computer Science (ECECS)

TECHNICAL REPORT

CONTENTS

Technical Report

Movie Recommendation

Submitted on: 04/24/2022

[Github Link 1]
[GitHub Link 2]

Team Members:
Name 1 Guanyu Zhou
Name 2 Chunwang Yuan

Questions?

Contact :

cyuan3@unh.newhaven.edu

gzhou2@unh.newhaven.edu

Movie Recommendation

Executive Summary

Movie Recommendation analysis would provide valuable business insights that drive decisions, as it is able to recommend real-time streamed movies and classify the movie based on user preference. A movie watcher would want to know what movies they like are similar, and if he/she will watch before making the next decision.

We were implementing machine learning models in order to compute similar movies based on user input. The outcome recommendation is very important and valuable in business decisions such as keeping daily active users or advertising on the webpage. Companies can estimate how well their movie recommendation performs and decide which business move they are taking next, cutting off a negative movie or promoting a popular movie.

We examined sentiment analysis on TMDB 5000 Movie Dataset from Kaggle.

Data Understanding

we are using 2 data files: tmdb_5000_movies.csv and tmdb_5000_credits.csv.
The tmdb_5000_movies.csv has four columns:movie_id, title, cast, crew

```
df_movies.head(2)
```

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...

The tmdb_5000_credits has 21 columns: title, genres, poster, status, runtime, release date, IMDB Rating, etc

```
df_credits.head(1)
```

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	production_countries	release_date	revenue	runtime	spoken_languages	status	tagline	title	vote_average	vote_count
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_3166_1": "US", "name": "United States o...	2009-12-10	2787965087	162.0	[{"iso_639_1": "en", "name": "English"}, {"iso...	Released	Enter the World of Pandora.	Avatar	7.2	11800

Data Preparation

Raw look:
As our data columns contain json objects, we need to convert them back to a list of literals. such as column “genres” etc.

```
df_movies.iloc[0].genres
```

```
'[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]'
```

Issues:
During cleaning, similar issues such as spaces between words and uppercase/lowercase letters have been solved by us using the NLTK library, which is a natural language machine library. And eventually, acquired the final data frame that we need.

	movie_id	title	tags
0	19995	Avatar	in the 22nd century, a parapleg marin is dispa...
1	285	Pirates of the Caribbean: At World's End	captain barbossa, long believ to be dead, ha c...
2	206647	Spectre	a cryptic messag from bond’ past send him on a...
3	49026	The Dark Knight Rises	follow the death of district attorney harvey d...
4	49529	John Carter	john carter is a war-weary, former militari ca...

Modeling

Cosine Similarity
Cosine similarly measures the similarity between two vectors of their inner product space. It simply means to measure if two vectors are in the same direction using the cosine of the angle. The score is between 0 and 1. Two movies can be similar based on their tags, name, or genres. We are using “tags” as our vectors and determine the similarity between movies.
A vector of the “Tag” look can be found in the above picture. and as the result we enumerate it with movie ID.

```
array([ 19995,    285, 206647, ..., 231617, 126186,  25975])
```

```
[60] sorted(list(enumerate(similarity[0])), reverse=True, key=lambda x:x[1])[1:6]
```

```
[(1216, 0.28676966733820225),
 (2409, 0.26901379342448517),
 (3730, 0.2605130246476754),
 (507, 0.255608593705383),
 (539, 0.25038669783359574)]
```

As we have ready-to-go data frames and we can now query our data frames to recommend movies. “input movie”: A list of movies of sorted cosine similarity between it and other movies “Tags”.

```
recommend('Batman Begins')
```

```
[(65, 0.40218090755486674), (1364, 0.35434169344615046), (1362, 0.3340765523905305), (3, 0.3177444546511212),
The Dark Knight
Batman
Batman
The Dark Knight Rises
10th & Wolf
Rockaway
Batman v Superman: Dawn of Justice
Synecdoche, New York
Defendor
Sexy Beast
City By The Sea
American Psycho
Harsh Times
Batman & Robin
```

Libraries	Processing
NumPy	Cleaning and rearranging arrays
nltk	Cleaning and extracting characters
	Cleaning and removing spaces
	Cleaning and changing lower/upper letters
Pandas	Cleaning and processing data frame
Stream it	Similar to the flask, making the model to the web for later evaluation

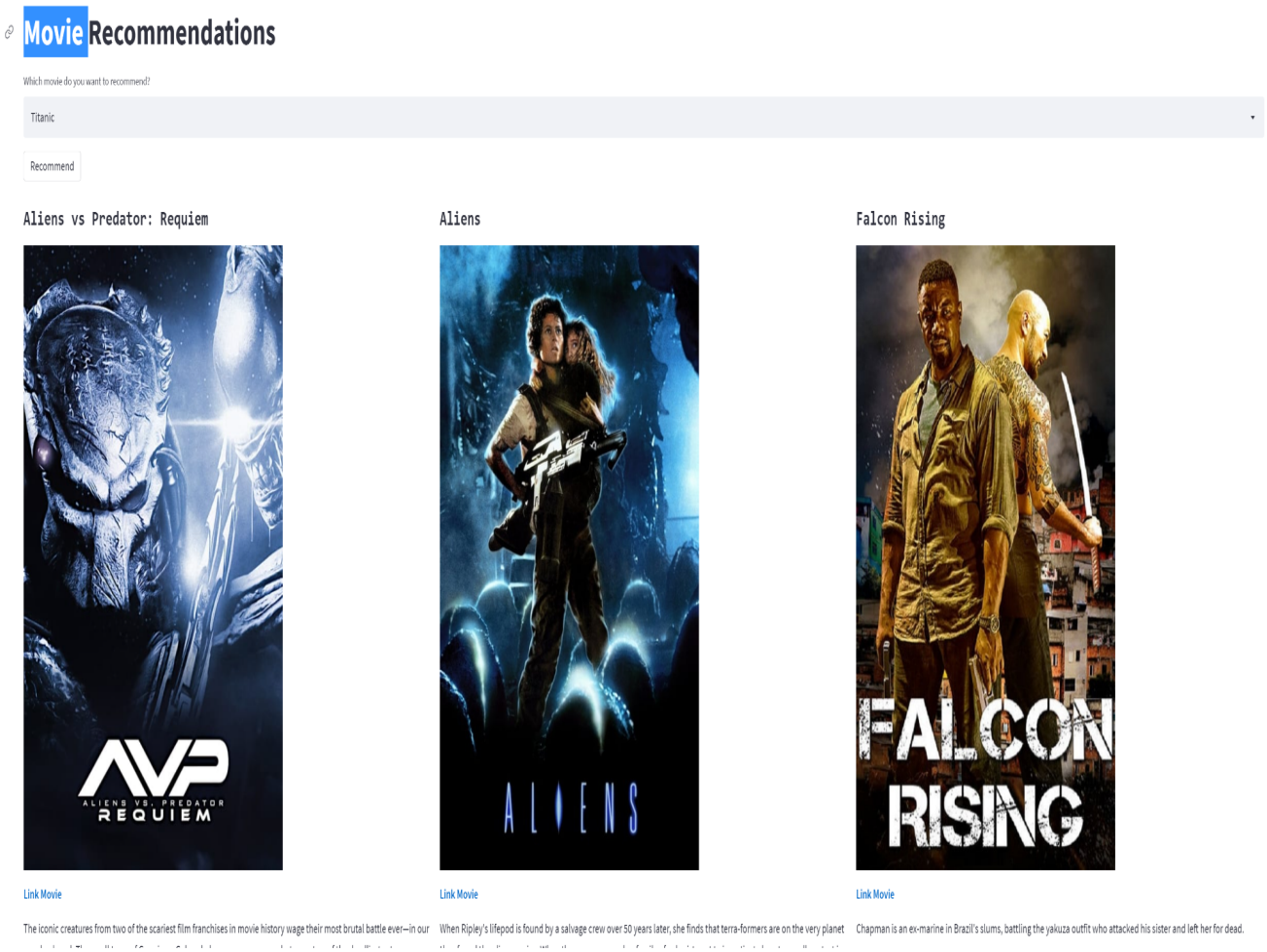
Evaluation

We are pickling our final model data for web building later use. As we are using content-based recommendations, the similarity of tags is a very simple recommendation model. The function makes it easy to show other recommendation movies and to improve this function, we can add a lot more features such as “actors”,” status”,” runtime” etc columns. But to prevent it from overfitting with too many features and simply test our cosine similarity method, we are using the simple recommendation model here.

Deployment

The Streamlight is an open-source library for machine learning and data science to turn data scripts into sharable web apps in minutes with its powerful and easy-to-use APIs compared to the flask.

We are making a webpage with a container that can take “user input”, and since it connects to the data frame files that we pickled and provided. It runs through our data frame and gets all of the closed “cosine similarity” movies back. Also since all of the movie's ID is listed in the IMDB database, we are using the API to get images based on the “moive_id”. The website is as shown below:



Conclusion

In our task of movie recommendation, our cosine similarity method finding out the similarity between vectors, in the words, the relationship between movies, can help movie companies to decide the next move in the business. For example, if a movie is poorly searched or watched, they can delete it from the database to save resources. Similarly, if a movie is very popular, they can redirect a lot of customers to this movie and insert the advertisements in between the watching for profit.

The difficulty that we faced was a feature that we used in cosine similarity is hard to decide which feature should be put in. A simple recommendation model means a simple calculation between two vectors, it could also be underfitting. However, content-based recommendations could also be easily overfitting by adding extra features, since most of them can be completely irrelevant to the input movie.

Our approach is still in its developing stage and our models will need further improvement by adding more information. In the future, we will be exploring more linguistic analysis and using other methods of machine learning to predict and improve our model.