TMDb Movie Data Analysis and Building a Movie Recommendation System

Part 3: Recommender System Through Content-Based Filtering

In this section, we will vectorize all relevant columns and create a cosine similarity matrix to use for our movie recommendation system.

- **id:** The ID of the movie (clear/unique identifier).
- **title:** The Official Title of the movie.
- **tagline:** The tagline of the movie.
- **release_date:** Theatrical Release Date of the movie.
- **genres:** Genres associated with the movie.
- **belongs_to_collection:** Gives information on the movie series/franchise the particular film belongs to.
- **original_language:** The language in which the movie was originally shot in.
- **budget_musd:** The budget of the movie in million dollars.
- **revenue_musd:** The total revenue of the movie in million dollars.
- **production_companies:** Production companies involved with the making of the movie.
- **production_countries:** Countries where the movie was shot/produced in.
- **vote_count:** The number of votes by users, as counted by TMDB.
- **vote_average:** The average rating of the movie.
- **popularity:** The Popularity Score assigned by TMDB.
- **runtime:** The runtime of the movie in minutes.
- **overview:** A brief blurb of the movie.
- **spoken_languages:** Spoken languages in the film.
- **poster_path:** The URL of the poster image.
- **cast:** (Main) Actors appearing in the movie.
- **cast_size:** number of Actors appearing in the movie.
- **director:** Director of the movie.
- **crew_size:** Size of the film crew (incl. director, excl. actors).

Loading the main libraries

In [1]:
```
import pandas as pd
import numpy as np
import pickle
from ast import literal_eval
```

```
pd.options.display.max_columns = 30
#pd.set_option('precision', 2)
```

Loading the Dataset

In [2]:
```
df = pd.read_csv('movies_complete.csv')
df.head()
```

Out[2]:

	belongs_to_collection	budget_musd	genres	id	original_language	overview	popula
0	Blondie Collection	NaN	Comedy	3924	en	Blondie and Dagwood are about to celebrate the...	2.
1	NaN	NaN	Adventure	6124	de	Der Mann ohne Namen is a German adventure movi...	0.
2	NaN	NaN	Drama|Romance	8773	fr	Love at Twenty unites five directors from five...	4.
3	New World Disorder	NaN	NaN	25449	en	Gee Atherton ripping the Worlds course the day...	1.
4	NaN	NaN	Family	31975	en	Elmo is making a very, very super special surp...	0.

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 578038 entries, 0 to 578037
Data columns (total 27 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   belongs_to_collection  15891 non-null   object
 1   budget_musd            24889 non-null   float64
 2   genres                 425057 non-null  object
 3   id                     578038 non-null  int64
 4   original_language      578038 non-null  object
 5   overview               496799 non-null  object
 6   popularity             578038 non-null  float64
 7   poster_path            420706 non-null  object
 8   production_companies   273625 non-null  object
 9   production_countries   377206 non-null  object
 10  release_date           556155 non-null  object
 11  revenue_musd           13603 non-null   float64
 12  runtime                474092 non-null  float64
 13  spoken_languages       370286 non-null  object
 14  tagline                90465 non-null   object
 15  title                  578038 non-null  object
 16  vote_average           256163 non-null  float64
 17  vote_count             578038 non-null  int64
 18  year                   556155 non-null  float64
 19  html                   420706 non-null  object
 20  cast_names             413840 non-null  object
 21  crew_names             502382 non-null  object
 22  director               494305 non-null  object
 23  profit_musd            8952 non-null    float64
 24  return_musd            8952 non-null    float64
 25  runtime_hours          474092 non-null  float64
 26  Franchise              578038 non-null  object
dtypes: float64(9), int64(2), object(16)
memory usage: 119.1+ MB
```

Dropping movies below a vote count threshold

In [4]:
```
min_votes = df.vote_count.quantile(0.95)
print(min_votes)
```

```
32.0
```

In [5]:
```
movies = df.copy().loc[:,['id', 'title', 'genres', 'cast_names', 'director', '
movies = movies.loc[df.vote_count >= min_votes].reset_index(drop=True)
print(f"Shape: {movies.shape}")
movies.head()
```

```
Shape: (29305, 9)
```

Out[5]:

	id	title	genres	cast_names	director	production_companies	overvie
0	8773	Love at Twenty	Drama|Romance	Jean-Pierre Léaud|Marie-France Pisier|Patrick ...	François Truffaut	Ulysse Productions|Unitec Films|Cinesecolo|Toh...	Love Twer unites fi directo fro five
1	2	Ariel	Drama|Crime|Comedy	Turo Pajala|Susanna Haavisto|Matti Pellonpää|E...	Aki Kaurismäki	Villealfa Filmproductions	Tais Kasurin is Finni co mir whose
2	3	Shadows in Paradise	Drama|Comedy	Matti Pellonpää|Kati Outinen|Sakari Kuosmanen|...	Aki Kaurismäki	Villealfa Filmproductions	episo in the l Nikand a garba
3	5	Four Rooms	Crime|Comedy	Tim Roth|Jennifer Beals|Antonio Banderas|Valer...	Allison Anders	Miramax|A Band Apart	It's T t Bellho first nig on t job
4	6	Judgment Night	Action|Thriller|Crime	Emilio Estevez|Cuba Gooding Jr.|Denis Leary|St...	Stephen Hopkins	Universal Pictures|Largo Entertainment|JVC	Wh racing a boxi mato Frar Mike,

In [6]:
```
movies.isnull().sum()
```

Out[6]:
```
id                        0
title                     0
genres                   86
cast_names              249
director                 90
production_companies    1796
overview                308
poster_path              57
html                     57
dtype: int64
```

Removing rows with null values

In [7]:
```
movies = movies.dropna().reset_index(drop=True)
print(f"Shape: {movies.shape}")
```

```
Shape: (27062, 9)
```

Creating a "tags" column by concatenating relevant columns

In [8]:
```
movies['tags'] = movies.genres + '|' + movies.cast_names + '|' + movies.direct
movies['tags'] = movies.tags.apply(lambda x: x.replace(' ', ''))
```

Vectorizing the "tags" column

In [9]:
```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
#cv = CountVectorizer(min_df=10)
cv = TfidfVectorizer(min_df=10)
cv_matrix = cv.fit_transform(movies.tags)
print(f"cv_matrix.shape: {cv_matrix.shape}")
```

```
cv_matrix.shape: (27062, 12246)
```

Creating a cosine similarity matrix

In [10]:
```
from sklearn.metrics.pairwise import cosine_similarity
cs_matrix = cosine_similarity(cv_matrix, dense_output=False)#.astype('int16')
print(f"cs_matrix.shape: {cs_matrix.shape}")
```

```
cs_matrix.shape: (27062, 27062)
```

Creating a function that returns the most similar movies based on the title of a movie

In [11]:
```
from IPython.display import HTML
# Function that takes in movie title as input and outputs most similar movies
def recommendations(title, cosine_sim=cs_matrix):
    # Get the index of the movie that matches the title
    idx = movies.loc[movies.title == title].index[0]
    # Get the pairwise similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx].toarray()[0]))
    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]
    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    results = movies[['html', 'title']].iloc[movie_indices].set_index(np.arang
    return HTML(results.to_html(escape=False))
```

In [12]:
```
recommendations('Star Wars')
```

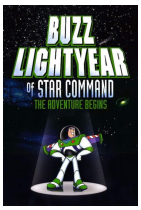
Out[12]:

		Top 10
1		The Empire Strikes Back
2		The Star Wars Holiday Special
3		Empire of Dreams: The Story of the Star Wars Trilogy
4		Return of the Jedi
5		Elstree 1976
6		Secrets of the Force Awakens: A Cinematic Journey
7		Star Wars: Episode III - Revenge of the Sith
8		The Skywalker Legacy

		Top 10
9		Electronic Labyrinth: THX 1138 4EB
10		Willow

In [13]:
```
recommendations('Toy Story')
```

Out[13]:

		Top 10
1		Toy Story 2
2		A Bug's Life
3		Tin Toy
4		Toy Story 3
5		The Incredibles
6		Cars
7		Buzz Lightyear of Star Command: The Adventure Begins
8		Monsters, Inc.

		Top 10
9		The Pixar Story
10		Hawaiian Vacation

In [14]:
```
recommendations('Akira')
```

Out[14]:

	Top 10	
1		Appleseed
2		City Hunter: Shinjuku Private Eyes
3		Doraemon: Nobita's Dinosaur
4		Lupin the Third: The Fuma Conspiracy
5		Ghost in the Shell Arise - Border 3: Ghost Tears
6		Ghost in the Shell Arise - Border 4: Ghost Stands Alone
7		Ghost in the Shell Arise - Border 1: Ghost Pain
8		Attack on Titan

	Top 10	
9		Attack on Titan II: End of the World
10		Battle Angel

Saving the "movies" DataFrame to a csv file

In [114…
```
movies.to_csv('movies_streamlit.csv', index=False)
```

Saving the cosine similarity matrix to a pkl file

In [ ]:
```
with open('cs_matrix.pkl', 'wb') as f:
    pickle.dump(cs_matrix, f)
```

Saving the cast and crew data of the final movies dataframe

In [2]:
```
credits = pd.read_csv('credits.csv')
```

In [24]:
```
credits['0'] = credits['0'].apply(lambda x: literal_eval(x) if type(x) == str
```

In [28]:
```
credits = pd.json_normalize(credits['0'])
```

In [33]:
```
credits = credits[credits.id.isin(movies.id)]
print(credits.shape)
```

```
(26961, 3)
```

In [34]:
```
credits.to_json('credits_streamlit.json', orient='records')
```

In [29]:
```
credits.head()
```

Out[29]:

	id	cast	crew
0	8773	[{'adult': False, 'gender': 2, 'id': 1653, 'kn...	[{'adult': False, 'gender': 2, 'id': 1650, 'kn...
1	2	[{'adult': False, 'gender': 2, 'id': 54768, 'k...	[{'adult': False, 'gender': 2, 'id': 16767, 'k...
2	3	[{'adult': False, 'gender': 2, 'id': 4826, 'kn...	[{'adult': False, 'gender': 2, 'id': 16767, 'k...
3	5	[{'adult': False, 'gender': 2, 'id': 3129, 'kn...	[{'adult': False, 'gender': 1, 'id': 3110, 'kn...
4	6	[{'adult': False, 'gender': 2, 'id': 2880, 'kn...	[{'adult': False, 'gender': 2, 'id': 2042, 'kn...

Creating a function to retrieve cast information

In [90]:

```
def get_cast(movie_id):
    cast = credits.loc[credits.id == movie_id, 'cast'].values
    names = []
    characters = []
    profile_paths = []
    base_url = 'https://image.tmdb.org/t/p/w500'

    for val in cast[0]:
        if hasattr(val, 'get'):
            names.append(val.get('name'))
            characters.append(val.get('character'))
            profile_paths.append(base_url + val.get('profile_path') if type(va
    return names, characters, profile_paths


names, characters, profile_paths = get_cast(55)
```

Saving the new credits dataframe to a pkl file

In [106…

```
with open('credits_streamlit.pkl', 'wb') as f:
    pickle.dump(credits, f)
```

In [2]:

```
y = pickle.load(open('credits_streamlit.pkl', 'rb'))
```

We have everything we need to create our app and deploy the movie recommendation system.