Movies Popularity Predictor and Recommendation System

Part 3: Recommendation System

Table of Contents

Recommendation System

In addition to predicting movie popularity, we will also implement a recommendation system that uses movie overviews to create embeddings and then ranks movies using cosine similarity. This system will engage users and keep them watching movies they enjoy.

Importing Libraries

In [1]:

```
import pandas as pd
from tqdm.notebook import tqdm
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
from sentence_transformers import SentenceTransformer

tqdm.pandas()
```

```
/Users/nataliaedelson/opt/anaconda3/lib/python3.8/site-packages/scipy/__init__.p
y:138: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this ve
rsion of SciPy (detected version 1.23.0)
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion} is requ
ired for this version of "
```

In [2]:

```
# Uploading data
tmdb_5000_cred = pd.read_csv(r'tmdb_5000_credits.csv', index_col=False)
tmdb_5000_mov = pd.read_csv(r'tmdb_5000_movies.csv',index_col=False)
```

In [3]:

```
# Merging the data
tmdb_5000_cred.columns = ['id','tittle','cast','crew']
tmdb_5000_mov = tmdb_5000_mov.merge(tmdb_5000_cred,on='id')
```

In [4]:
```
# Creating a copy
df = tmdb_5000_mov.copy()
```

In [5]:
```
# Storing the necessary columns
imp_cols = ['genres','original_title','overview','popularity']
```

In [6]:
```
# Storing the columns in a new dataframe'
data = df[imp_cols]
```

In [7]:
```
# Viewing dataframe
data.head()
```

Out[7]:

	genres	original_title	overview	popularity
0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577
1	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615
2	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	Spectre	A cryptic message from Bond's past sends him o...	107.376788
3	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950
4	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	John Carter	John Carter is a war-weary, former military ca...	43.926995

In [8]:
```
# Extracting names from a list of dictionaries
def get_val(dictionary_list):
    val = [d['name'] for d in eval(dictionary_list)]
    return val
```

In [9]:
```
# Creating new column withe the names
data['genres'] = data['genres'].progress_apply(get_val)
```

```
<ipython-input-9-4a476894162d>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['genres'] = data['genres'].progress_apply(get_val)
```

In [10]:
```
# Viewing genres
data['genres']
```

Out[10]:
```
0        [Action, Adventure, Fantasy, Science Fiction]
1                         [Adventure, Fantasy, Action]
2                           [Action, Adventure, Crime]
3                     [Action, Crime, Drama, Thriller]
```

```
4                  [Action, Adventure, Science Fiction]
                                        ...
4798                           [Action, Crime, Thriller]
4799                                  [Comedy, Romance]
4800                [Comedy, Drama, Romance, TV Movie]
4801                                                 []
4802                                      [Documentary]
Name: genres, Length: 4803, dtype: object
```

In [11]:
```
# Viewing selected 10 overviews
data['overview'][10]
```

Out[11]:
```
'Superman returns to discover his 5-year absence has allowed Lex Luthor to walk
free, and that those he was closest too felt abandoned and have moved on. Luthor
plots his ultimate revenge that could see millions killed and change the face of
the planet forever, as well as ridding himself of the Man of Steel.'
```

Huggingface embedding

The "paraphrase-MiniLM-L6-v2" model is an embedding model that converts text into numerical representations. These embeddings capture the context of the input text, allowing for comparison like in our case - similarity measurement - in order to get closer mattch to another movie review.

In [12]:
```
# pip install -U sentence-transformers
```

In [13]:
```
# Storing the model
model = SentenceTransformer('paraphrase-MiniLM-L6-v2')
```

In [14]:
```
# Viewing
data
```

Out[14]:

	genres	original_title	overview	popularity
0	[Action, Adventure, Fantasy, Science Fiction]	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577
1	[Adventure, Fantasy, Action]	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615
2	[Action, Adventure, Crime]	Spectre	A cryptic message from Bond's past sends him o...	107.376788
3	[Action, Crime, Drama, Thriller]	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950
4	[Action, Adventure, Science Fiction]	John Carter	John Carter is a war-weary, former military ca...	43.926995
...	...	...	...	...
4798	[Action, Crime, Thriller]	El Mariachi	El Mariachi just wants to play his guitar and ...	14.269792
4799	[Comedy, Romance]	Newlyweds	A newlywed couple's honeymoon is upended by	0.642552

	genres	original_title	overview	popularity
			th...	
4800	[Comedy, Drama, Romance, TV Movie]	Signed, Sealed, Delivered	"Signed, Sealed, Delivered" introduces a dedic...	1.444476
4801	[]	Shanghai Calling	When ambitious New York attorney Sam is sent t...	0.857008
4802	[Documentary]	My Date with Drew	Ever since the second grade when he first saw ...	1.929883

4803 rows × 4 columns

Taking an overview text as input and return embedding using hugging face pre-train model.

In [15]:
```
# Getting the embedding
def get_embedding_sent(overview):

    #Sentences we want to encode.
    sentence = [overview]

    #Sentences are encoded by calling model.encode()
    embedding = model.encode(sentence)
    return embedding[0].tolist()
```

In [16]:
```
# Viewing data
data
```

Out[16]:

	genres	original_title	overview	popularity
0	[Action, Adventure, Fantasy, Science Fiction]	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577
1	[Adventure, Fantasy, Action]	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615
2	[Action, Adventure, Crime]	Spectre	A cryptic message from Bond’s past sends him o...	107.376788
3	[Action, Crime, Drama, Thriller]	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950
4	[Action, Adventure, Science Fiction]	John Carter	John Carter is a war-weary, former military ca...	43.926995
...	...	...	...	...
4798	[Action, Crime, Thriller]	El Mariachi	El Mariachi just wants to play his guitar and ...	14.269792
4799	[Comedy, Romance]	Newlyweds	A newlywed couple's honeymoon is upended by th...	0.642552
4800	[Comedy, Drama, Romance, TV Movie]	Signed, Sealed, Delivered	"Signed, Sealed, Delivered" introduces a dedic...	1.444476
4801	[]	Shanghai Calling	When ambitious New York attorney Sam is sent t...	0.857008

	genres	original_title	overview	popularity
4802	[Documentary]	My Date with Drew	Ever since the second grade when he first saw ...	1.929883

4803 rows × 4 columns

In [17]:
```
# Dropping null values
data.dropna(inplace=True)
```

```
<ipython-input-17-961934b8d315>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy
  data.dropna(inplace=True)
```

We are adding a new column with the embedding for each movie

In [25]:
```
# Calculating sentence embeddings for movie overviews

data['Sent_Embedding'] = ( data['overview'].progress_apply
                           (get_embedding_sent)
                          )
# data['Embedding'] = data['overview'].progress_apply(lambda x: get_embedding(x)
data.head()
```

```
<ipython-input-25-dfebc2c1b2c2>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['Sent_Embedding'] = ( data['overview'].progress_apply
```

Out[25]:

	genres	original_title	overview	popularity	Sent_Embedding
0	[Action, Adventure, Fantasy, Science Fiction]	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[0.13959655165672302, 0.2724362313747406, 0.45...
1	[Adventure, Fantasy, Action]	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[0.1408797800540924, -0.13821646571159363, -0....
2	[Action, Adventure, Crime]	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[-0.6094507575035095, -0.00885468814522028, -0...
3	[Action, Crime, Drama, Thriller]	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950	[-0.10923074185848236, -0.023745622485876083, ...
4	[Action, Adventure, Science Fiction]	John Carter	John Carter is a war-weary, former military ca...	43.926995	[-0.05899054929614067, -0.03969033807516098, -...

We convert the input text into numerical representations using an embedding function. By calculating the cosine similarity between the input text embedding and the embeddings of all movies, we determine their similarity. The dataframe is then sorted by this similarity, and the top 10 movies with the highest similarity are selected.

The selected movies are returned with their titles, similarity values, genres, and popularity, providing a list of recommended movies that closely match the type pf gerne that was inputed.

In [32]:
```
def get_recommendation(text,embd_fn,embd_col):

    # Creating a copy
    temp = data.copy()

    # Getting the embedding representations of all movies

    y = embd_fn(text)

    # Preparing for calculation by converting to array

    x_embed = np.array([i for i in temp[embd_col]])


    # Assignning  the cosine similarity values to
    # the 'similarity' column

    cs = cosine_similarity(x_embed,np.array(y).reshape(1,-1))

    # Sorting the dataframe by similarity
    temp['similarity'] = cs


    # Selecting the columns of interest for the
    # recommendations and get the top 10 results

    temp = temp.sort_values('similarity',ascending=False)

    # Selecting the columns of interest for the
    # recommendations and get the top 10 results

    temp = temp[['original_title','similarity',
                 'genres','popularity']].head(10)

    return temp

#https://docs.pinecone.io/docs/movie-recommender
```

In [33]:
```
# Store Example
ex = get_recommendation('horror movie with action',get_embedding_sent,'Sent_Embe
```

In [34]:
```
# Resetting the index
ex.reset_index(drop=1)
```

Out[34]:

	original_title	similarity	genres	popularity
0	Scream	0.602784	[Crime, Horror, Mystery]	45.996110
1	The Horror Network Vol. 1	0.602098	[Horror]	0.392658
2	Krampus	0.590564	[Horror, Comedy, Fantasy]	31.565117
3	Super 8	0.585009	[Thriller, Science Fiction, Mystery]	37.069253
4	Extreme Movie	0.579823	[Comedy]	8.148187
5	Grindhouse	0.579100	[Thriller, Action, Horror]	16.637642
6	Disaster Movie	0.572808	[Action, Comedy]	16.238961
7	Seed of Chucky	0.571745	[Drama, Horror, Comedy]	12.653831
8	Superhero Movie	0.566473	[Action, Comedy, Science Fiction]	19.088655
9	Jagal	0.560564	[Documentary]	8.887411

In [35]:

```
# Looking into the data before we store it
data
```

Out[35]:

	genres	original_title	overview	popularity	Sent_Embedding
0	[Action, Adventure, Fantasy, Science Fiction]	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[0.13959655165672302, 0.2724362313747406, 0.45...
1	[Adventure, Fantasy, Action]	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[0.1408797800540924, -0.13821646571159363, -0....
2	[Action, Adventure, Crime]	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[-0.6094507575035095, -0.00885468814522028, -0...
3	[Action, Crime, Drama, Thriller]	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950	[-0.10923074185848236, -0.023745622485876083, ...
4	[Action, Adventure, Science Fiction]	John Carter	John Carter is a war-weary, former military ca...	43.926995	[-0.05899054929614067, -0.03969033807516098, -...
...	...	...	...	...	...
4798	[Action, Crime, Thriller]	El Mariachi	El Mariachi just wants to play his guitar and ...	14.269792	[0.22522863745689392, 0.32955336570739746, -0....
4799	[Comedy, Romance]	Newlyweds	A newlywed couple's honeymoon is upended by th...	0.642552	[-0.18524450063705444, 0.11452585458755493, 0....

	genres	original_title	overview	popularity	Sent_Embedding
4800	[Comedy, Drama, Romance, TV Movie]	Signed, Sealed, Delivered	"Signed, Sealed, Delivered" introduces a dedic...	1.444476	[-0.47836053371429443, -0.14381982386112213, -...
4801	[]	Shanghai Calling	When ambitious New York attorney Sam is sent t...	0.857008	[0.05273731052875519, 0.20882274210453033, 0.1...
4802	[Documentary]	My Date with Drew	Ever since the second grade when he first saw ...	1.929883	[0.06735286861658096, -0.002224662574008107, 0...

4800 rows × 5 columns

In [36]:
```
# Exporting the data that is needed
data.to_csv('Embedding_chkpoint_1.csv',index=False)
```

Conclusion

Our recommendation system utilizes BERT embeddings, a language model, to build a personalized movie recommendation system based on movie overviews. By analyzing the content and context of movies, we can suggest similar movies within your preferred genre. By providing personalized recommendations, this system aims to keep customers engaged and encourage them to explore movies tailored to their preferences