Hello, my name is Ricardo Avelar, and my project focuses on developing a Netflix movie recommendation system. The project incorporates Natural Language Processing (NLP) techniques, primarily for text data processing. To clean and prepare the text, I utilized NLTK for stop word removal and lemmatization, alongside regex-based cleaning with the re library to ensure the data was properly formatted. For feature extraction, I implemented TF-IDF(Term frequency-inverse document frequency) to identify the most significant words in the dataset. Using this, I generated a TF-IDF matrix to represent the text numerically, enabling the computation of similarity. Finally, I employed cosine similarity to compare the TF-IDF vectors, allowing me to find related content and make accurate recommendations. This combination of NLP and feature engineering forms the core of my recommendation system, providing users with personalized suggestions based on textual data.

In [1]:
```
#import required libraries
import pandas as pd
import numpy as np

#Import data
#Dataset downloaded from https://www.kaggle.com/datasets/shivamb/netflix-shows
df = pd.read_csv('netflix_titles.csv')

#Print the head of the DataFrame
df.head()
```

Out[1]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	du
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	S
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	S

In [2]:
```
#Print the tail of the DataFame
df.tail()
```

Out[2]:

	show_id	type	title	director	cast	country	date_added	release_year	ratir
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2007	
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2018	TV-Y
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2009	
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	2006	P
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	2015	TV-'

In [3]:
```
#check for missing values
print(df.isnull().sum())
```

```
show_id          0
type             0
title            0
director      2634
cast           825
country        831
date_added      10
release_year     0
rating           4
duration         3
listed_in        0
description      0
dtype: int64
```

In [4]:
```
#Drop columns with excessive missing values
df = df.drop(columns=['director'])
```

In [5]:
```
#Replace NaN with an empty string or with unknown
df['cast'] = df['cast'].fillna('Unknown')
df['country'] = df['country'].fillna('Unknown')
df['date_added'] = df['date_added'].fillna('Unknown')
df['rating'] = df['rating'].fillna('Unknown')
df['duration'] = df['duration'].fillna('Unknown')
```

In [6]:
```
#Text Preprocessing, import required libraries
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

In [7]:
```
#download necessary NLTK Data
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ravelar2\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ravelar2\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[7]:
```
True
```

In [8]:
```
#Initialize stopwords and lemmatizer
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
```

In [9]:
```
#Create a function to prepare raw text data for further analysis
def preprocess_text(text):
    text = text.lower()  # Lowercase to make text case-insensitive
    text = re.sub(r'[^a-z\s]', '', text)  # Remove punctuation and numbers
    tokens = text.split()  # Splits text into individual words(tokens)
    tokens = [word for word in tokens if word not in stop_words]  # Remove sto
    tokens = [lemmatizer.lemmatize(word) for word in tokens]  # Lemmatize, con
    return ' '.join(tokens)
```

In [10]:
```
#apply preprocessing to description column
df['cleaned_description'] = df['description'].apply(preprocess_text)
```

In [11]:
```
#Import TfIdfVectorizer from the scikit-learn library
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [12]:
```
#Define a TF-IDF Vectorizer Object. Remove all english stopwords
tfidf = TfidfVectorizer(stop_words='english')
```

In [13]:
```
#Drop old description column
df = df.drop(columns=['description'])
```

In [14]:
```
#Print the new cleaned DataFrame
df.head()
```

Out[14]:

	show_id	type	title	cast	country	date_added	release_year	rating	duration
0	s1	Movie	Dick Johnson Is Dead	Unknown	United States	September 25, 2021	2020	PG-13	90 min
1	s2	TV Show	Blood & Water	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons
2	s3	TV Show	Ganglands	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	Unknown	September 24, 2021	2021	TV-MA	1 Season
3	s4	TV Show	Jailbirds New Orleans	Unknown	Unknown	September 24, 2021	2021	TV-MA	1 Season
4	s5	TV Show	Kota Factory	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons

In [15]:
```
#Print the last 5 shows
df.tail()
```

Out[15]:

	show_id	type	title	cast	country	date_added	release_year	rating	durati
8802	s8803	Movie	Zodiac	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2007	R	158 r
8803	s8804	TV Show	Zombie Dumb	Unknown	Unknown	July 1, 2019	2018	TV-Y7	Seasc
8804	s8805	Movie	Zombieland	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2009	R	88 r
8805	s8806	Movie	Zoom	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	2006	PG	88 r
8806	s8807	Movie	Zubaan	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	2015	TV-14	111 r

In [16]:
```
#Verify no missing values
print(df.isnull().sum())
```

```
show_id              0
type                 0
title                0
cast                 0
country              0
date_added           0
release_year         0
rating               0
duration             0
listed_in            0
cleaned_description  0
dtype: int64
```

In [17]:
```
#Construct the required TF-IDF matrix by applying the fit_transform method on
tfidf_matrix = tfidf.fit_transform(df['cleaned_description'])
```

In [18]:
```
#Output the shape of tfidf_matrix
tfidf_matrix.shape
```

Out[18]: (8807, 17884)

In [19]:
```
# Import linear_kernel to compute the similiraity between two vectors
from sklearn.metrics.pairwise import linear_kernel

#This code computes the cosine similarity between all pairs of documents repre
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

In [20]:
```
#Creates a mapping where each movie title in the 'title' column maps to its co
indices = pd.Series(df.index, index=df['title']).drop_duplicates()
```

In [21]:
```
# Function that takes in movie title as input and gives recommendations
def content_recommender(title, cosine_sim=cosine_sim, df=df, indices=indices):
    # Obtain the index of the movie that matches the title
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    # And convert it into a list of tuples as described above
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the cosine similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies. Ignore the first movie, be
    sim_scores = sim_scores[1:11]

    # Get the movie indices using a for loop
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return df['title'].iloc[movie_indices]
```

In [26]:
```
#Verify cosine_sim, The shpae of the cosine-sim matrix indicates that it is co
print(cosine_sim.shape)
```

(8807, 8807)

In [22]:
```
#Get recommendations for The Walking Dead
#this step will take time
content_recommender('The Walking Dead')
```

Out[22]:
```
7087                          Into the Forest
696                              Black Summer
6353                                    Bokeh
6961                              Here Alone
7286                                  Legion
5921    Bill Burr: I'm Sorry You Feel That Way
1841                  The Last Kids on Earth
4816                  Expelled from Paradise
2774                                  7SEEDS
3281                              The Stranded
Name: title, dtype: object
```

In [29]:
```
#Second example
content_recommender('2012')
```

Out[29]:
```
728         The Devil Below
7610    NOVA: Killer Floods
3111                  Agent
6998          Horror Homes
2489     Sayed the Servant
8308             The Force
4292     Maps to the Stars
7529        Mutant Busters
2298     Southern Survival
3144        Potato Potahto
Name: title, dtype: object
```

In [23]:
```
#we need to drop na
#since it doesn't know how to deal with na's
df.dropna(subset=['title'], inplace=True)
```

In [24]:
```
#In order to find all titles that start with the letter B
#filter df
filtered_df=df[df['title'].str.startswith('B')]
```

In [25]:
```
#print the titles that starts with the letter B
print (filtered_df['title'])
```

```
1                                  Blood & Water
11                               Bangkok Breaking
38                             Birth of the Dragon
88      Blood Brothers: Malcolm X & Muhammad Ali
106                                        Bunk'd
                           ...
6391       Burlesque: Heart of the Glitter Tribe
6392                                     Burning
6393                                  Burnistoun
6394                                    Bushwick
6395                                 Butterfield 8
Name: title, Length: 576, dtype: object
```

In this project, I developed a Netflix recommendation system using Natural Language Processing (NLP) techniques. By preprocessing text data, extracting features with TF-IDF vectorization, and computing similarity scores, the system effectively identifies relevant content to recommend based on user preferences. This project highlights the power of NLP in transforming unstructured text data into actionable insights. Techniques such as stop word removal, lemmatization, and vectorization were integral in preparing and analyzing the data. The resulting recommendation system provides a user-friendly platform for discovering new content and establishes a foundation for future enhancements, including the integration of user ratings, sentiment analysis, or advanced topic modeling.