Movie Recommendation System

Final Report

Sri Lanka Institute of Information Technology

Information Retrieval and Web Analytics - IT3041

Team DataNauts

IT21468360	Rizvi F.A
IT21924750	Senevirathna W.P.U.
IT21276200	Samarakoon S.M.R.S.B
IT21329760	Nanayakkara N.W.B.S

Declaration

This project report is our original work, and the content is not plagiarized from any other resource. References for all the content taken from external resources are correctly cited. To the best of our knowledge, this report does not contain any material published or written by third parties, except as acknowledged in the text.

Acknowledgement

We would like to convey our heartfelt gratitude to everyone who helped us to complete this project. We are especially grateful to our lecturers, Dr. Lakmini and Mr. Samadhi and our Lab Instructor, Mr. Derick, who have supported and directed us in carrying out the project and have been observant since the first stage of the project, providing stimulating recommendations and encouragement. We would also like to express our heartfelt gratitude to our parents and friends.

Table of Contents

Background

In the ever-expanding world of cinema, the demand for personalized movie recommendations is constantly growing. Both traditional viewers and users of streaming platforms are inundated with countless options, and making the right choice can often be an overwhelming task. To address this challenge, a content-based movie recommendation system was developed that harnesses the power of machine learning and natural language processing (NLP) techniques.

This recommendation system is designed to provide users with customized movie suggestions and allow them to explore movies that match their individual preferences. Users have the option to enter their choices based on movie titles or genres. The functionality of the system goes beyond just presenting random recommendations; it offers specific movie suggestions that resonate with each user's unique tastes.

The dataset that we used to build the system contains the top 10,000 TMDB movies up to July 26, 2022. It includes information such as movie ID, title, genre, original language, movie overview, popularity, release date, average vote, and number of votes.

The project workflow began with data preprocessing and cleaning to refine the dataset. This was followed by data analysis for a thorough understanding of the dataset. In Feature Engineering, we selected the movie title and genre for content-based filtering. The Similarity Computation step has successfully implemented the recommendation function. Finally, we deployed the system and made movie recommendations available to users.

To make the system accessible and user-friendly, it has been thoughtfully implemented using Streamlit, a Python library for web application development. This interface allows users to conveniently select movie presets from a drop-down menu or by entering movie titles and genres. Once the user's input is captured, an algorithm takes over and suggests a curated list of movies that closely match the user's preferences. To enhance the quality of the recommendations, these films are presented in order of IMDb rating, ensuring that the highest-rated options are highlighted. IMDb ratings offer a valuable way for movie enthusiasts to assess the quality and appeal of a film through collective audience opinions.

As the world of cinema continues to expand with an ever-growing library of movies, the potential for content-based recommendation systems remains huge. This system represents the fusion of technology and entertainment, offering users a more rewarding and satisfying movie-watching experience.

Methodology

1. Data Gathering
2. Data Pre-processing
3. Similarity Computation
4. Recommendation Function
5. Deployment

1. Data Gathering

We used a public data set to build our movie recommendation system.

Dataset Name – TMDB Movies Dataset
Provided by – Kaggle.
Link to the dataset - https://www.kaggle.com/datasets/ahsanaseer/top-rated-tmdb-movies-10k?fbclid=IwAR2MpWrWpcw2QNCv_FZg2l0sjBh9xAvhrqtnZBO9K-QS6PHI1aHkdB6qLa0

This is a data set of 10k top-rated TMDB movies till 26-July-2022. The Dataset contains the movie id, title, genre, original language, overview, popularity, release date, vote average, and vote count.

2. Data Preprocessing

Content – based Filtering (Based on the Movie Title)

2.1. Load the Dataset

- Loaded the movie dataset into Jupyter Notebook.

Import dependencies

In [5]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pickle
```

In [6]:
```
# Load the data set
movies = pd.read_csv('moviesData.csv')
movies
```

Out[6]:

	id	title	genre	original_language	overview	popularity	release_date	vote_average	vote_count
0	278	The Shawshank Redemption	Drama,Crime	en	Framed in the 1940s for the double murder of h...	94.075	1994-09-23	8.7	21862
1	19404	Dilwale Dulhania Le Jayenge	Comedy,Drama,Romance	hi	Raj is a rich, carefree, happy-go-lucky second...	25.408	1995-10-19	8.7	3731
2	238	The Godfather	Drama,Crime	en	Spanning the years 1945 to 1955, a chronicle o	90.585	1972-03-14	8.7	16280
3	424	Schindler's List	Drama,History,War	en	The true story of how businessman Oskar Schind...	44.761	1993-12-15	8.6	12959
4	240	The Godfather: Part II	Drama,Crime	en	In the continuing saga of the Corleone crime f...	57.749	1974-12-20	8.6	9811
...	...	...	...	...	...	...	...	...	...
9995	10196	The Last Airbender	Action,Adventure,Fantasy	en	The story follows the adventures of Aang, a yo...	98.322	2010-06-30	4.7	3347
9996	331446	Sharknado 3: Oh Hell No!	Action,TV Movie,Science Fiction,Comedy,Adventure	en	The sharks take bite out of the East Coast whe...	12.490	2015-07-22	4.7	417
9997	13995	Captain America	Action,Science Fiction,War	en	During World War II, a brave, patriotic Americ...	18.333	1990-12-14	4.6	332
9998	2312	In the Name of the King: A Dungeon Siege Tale	Adventure,Fantasy,Action,Drama	en	A man named Farmer sets out to rescue his kidn...	15.159	2007-11-29	4.7	668
9999	455957	Domino	Thriller,Action,Crime	en	Seeking justice for his partner's murder by an...	16.482	2019-05-31	4.6	221

2.2. Data Cleaning

- Checked for missing values.

Data Cleaning

In [7]:
```
# Check for null values
movies.isnull().sum()
```

Out[7]:
```
id                   0
title                0
genre                3
original_language    0
overview            13
popularity           0
release_date         0
vote_average         0
vote_count           0
dtype: int64
```

2.3. Data Exploration

- Analyzed the data set to identify its details well.

In [9]:
```
# Describe the data set
movies.describe()
```

Out[9]:

	id	popularity	vote_average	vote_count
count	10000.000000	10000.000000	10000.000000	10000.000000
mean	161243.505000	34.697267	6.621150	1547.309400
std	211422.046043	211.684175	0.766231	2648.295789
min	5.000000	0.600000	4.600000	200.000000
25%	10127.750000	9.154750	6.100000	315.000000
50%	30002.500000	13.637500	6.600000	583.500000
75%	310133.500000	25.651250	7.200000	1460.000000
max	934761.000000	10436.917000	8.700000	31917.000000

In [10]:
```
# Check movies info
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 10000 non-null  int64
 1   title              10000 non-null  object
 2   genre              9997 non-null   object
 3   original_language  10000 non-null  object
 4   overview           9987 non-null   object
 5   popularity         10000 non-null  float64
 6   release_date       10000 non-null  object
 7   vote_average       10000 non-null  float64
 8   vote_count         10000 non-null  int64
dtypes: float64(2), int64(2), object(5)
memory usage: 703.3+ KB
```

2.4. Feature Engineering

- Checked the column names of the data set.

Feature Engineering

In [11]:
```
# check columns
movies.columns
```

Out[11]:
```
Index(['id', 'title', 'genre', 'original_language', 'overview', 'popularity',
       'release_date', 'vote_average', 'vote_count'],
      dtype='object')
```

- Selected the features needed for content-based filtering, based on the movie title.

In [12]:
```
# select features needed for content-based filtering, based on the movie title.
movies = movies[['id', 'title',  'overview','genre']]
movies
```

Out[12]:

	id	title	overview	genre
0	278	The Shawshank Redemption	Framed in the 1940s for the double murder of h...	Drama,Crime
1	19404	Dilwale Dulhania Le Jayenge	Raj is a rich, carefree, happy-go-lucky second...	Comedy,Drama,Romance
2	238	The Godfather	Spanning the years 1945 to 1955, a chronicle o...	Drama,Crime
3	424	Schindler's List	The true story of how businessman Oskar Schind...	Drama,History,War
4	240	The Godfather: Part II	In the continuing saga of the Corleone crime f...	Drama,Crime
...	...	...	...	...
9995	10196	The Last Airbender	The story follows the adventures of Aang, a yo...	Action,Adventure,Fantasy
9996	331446	Sharknado 3: Oh Hell No!	The sharks take bite out of the East Coast whe...	Action,TV Movie,Science Fiction,Comedy,Adventure
9997	13995	Captain America	During World War II, a brave, patriotic Americ...	Action,Science Fiction,War
9998	2312	In the Name of the King: A Dungeon Siege Tale	A man named Farmer sets out to rescue his kidn...	Adventure,Fantasy,Action,Drama
9999	455957	Domino	Seeking justice for his partner's murder by an...	Thriller,Action,Crime

- Merged ‘genre’ and ‘overview’ columns and created a new column called ‘description.’

In [13]:
```
# merge 'overview' and 'genre' and create a new column 'description'
movies['description'] = movies['overview'] + movies['genre']
movies
```

Out[13]:

	id	title	overview	genre	description
0	278	The Shawshank Redemption	Framed in the 1940s for the double murder of h...	Drama,Crime	Framed in the 1940s for the double murder of h...
1	19404	Dilwale Dulhania Le Jayenge	Raj is a rich, carefree, happy-go-lucky second...	Comedy,Drama,Romance	Raj is a rich, carefree, happy-go-lucky second...
2	238	The Godfather	Spanning the years 1945 to 1955, a chronicle o...	Drama,Crime	Spanning the years 1945 to 1955, a chronicle o...
3	424	Schindler's List	The true story of how businessman Oskar Schind...	Drama,History,War	The true story of how businessman Oskar Schind...
4	240	The Godfather: Part II	In the continuing saga of the Corleone crime f...	Drama,Crime	In the continuing saga of the Corleone crime f...
...	...	...	...	...	...
9995	10196	The Last Airbender	The story follows the adventures of Aang, a yo...	Action,Adventure,Fantasy	The story follows the adventures of Aang, a yo...
9996	331446	Sharknado 3: Oh Hell No!	The sharks take bite out of the East Coast whe...	Action,TV Movie,Science Fiction,Comedy,Adventure	The sharks take bite out of the East Coast whe...
9997	13995	Captain America	During World War II, a brave, patriotic Americ...	Action,Science Fiction,War	During World War II, a brave, patriotic Americ...

- Dropped ‘genre’ and ‘overview’ columns from the dataset.

In [14]:
```
# Drop columns
newDf = movies.drop(columns=['overview','genre'])
newDf
```

Out[14]:

	id	title	description
0	278	The Shawshank Redemption	Framed in the 1940s for the double murder of h...
1	19404	Dilwale Dulhania Le Jayenge	Raj is a rich, carefree, happy-go-lucky second...
2	238	The Godfather	Spanning the years 1945 to 1955, a chronicle o...
3	424	Schindler's List	The true story of how businessman Oskar Schind...
4	240	The Godfather: Part II	In the continuing saga of the Corleone crime f...
...	...	...	...
9995	10196	The Last Airbender	The story follows the adventures of Aang, a yo...
9996	331446	Sharknado 3: Oh Hell No!	The sharks take bite out of the East Coast whe...
9997	13995	Captain America	During World War II, a brave, patriotic Americ...
9998	2312	In the Name of the King: A Dungeon Siege Tale	A man named Farmer sets out to rescue his kidn...

Content – based Filtering (Based on the Genre)

2.5. Load the Dataset

- Loaded the movie dataset into Jupyter Notebook.

Import Dependencies

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

In [2]:
```
# Load data set

movie_list = pd.read_csv('moviesData.csv')
movie_list
```

Out[2]:

	id	title	genre	original_language	overview	popularity	release_date	vote_average	vote_count
0	278	The Shawshank Redemption	Drama,Crime	en	Framed in the 1940s for the double murder of h...	94.075	1994-09-23	8.7	21862
1	19404	Dilwale Dulhania Le Jayenge	Comedy,Drama,Romance	hi	Raj is a rich, carefree, happy-go-lucky second...	25.408	1995-10-19	8.7	3731
2	238	The Godfather	Drama,Crime	en	Spanning the years 1945 to 1955, a chronicle o...	90.585	1972-03-14	8.7	16280
3	424	Schindler's List	Drama,History,War	en	The true story of how businessman Oskar Schind...	44.761	1993-12-15	8.6	12959
4	240	The Godfather: Part II	Drama,Crime	en	In the continuing saga of the Corleone crime f...	57.749	1974-12-20	8.6	9811
...	...	...	...	...	...	...	...	...	...
9995	10196	The Last Airbender	Action,Adventure,Fantasy	en	The story follows the adventures of Aang, a yo...	98.322	2010-06-30	4.7	3347
9996	331446	Sharknado 3: Oh Hell No!	Action,TV Movie,Science Fiction,Comedy,Adventure	en	The sharks take bite out of the East Coast whe...	12.490	2015-07-22	4.7	417
9997	13995	Captain America	Action,Science Fiction,War	en	During World War II, a brave, patriotic Americ...	18.333	1990-12-14	4.6	332
9998	2312	In the Name of the King: A Dungeon Siege Tale	Adventure,Fantasy,Action,Drama	en	A man named Farmer sets out to rescue his kidn...	15.159	2007-11-29	4.7	668
9999	455957	Domino	Thriller,Action,Crime	en	Seeking justice for his partner's murder by an...	16.482	2019-05-31	4.6	221

2.6. Data Cleaning

- Checked for missing values.

Data Cleaning

In [4]:
```
# Check for null values
movie_list.isnull().sum()
```

Out[4]:
```
id                  0
title               0
genre               3
original_language   0
overview           13
popularity          0
release_date        0
vote_average        0
vote_count          0
dtype: int64
```

- Filled the missing ‘genre’ values with ‘0’ and converted it into a string.

In [5]:
```
# Fill NaN values in 'Genre' with 0
movie_list['genre'] = movie_list['genre'].fillna(0)
# Convert 'genre' column to strings
movie_list['genre'] = movie_list['genre'].astype(str)
movie_list
```

Out[5]:

	id	title	genre	original_language	overview	popularity	release_date	vote_average	vote_count
0	278	The Shawshank Redemption	Drama,Crime	en	Framed in the 1940s for the double murder of h...	94.075	1994-09-23	8.7	21862
1	19404	Dilwale Dulhania Le Jayenge	Comedy,Drama,Romance	hi	Raj is a rich, carefree, happy-go-lucky second...	25.408	1995-10-19	8.7	3731
2	238	The Godfather	Drama,Crime	en	Spanning the years 1945 to 1955, a chronicle o...	90.585	1972-03-14	8.7	16280
3	424	Schindler's List	Drama,History,War	en	The true story of how businessman Oskar Schind...	44.761	1993-12-15	8.6	12959
4	240	The Godfather: Part	Drama,Crime	en	In the continuing saga of the Corleone	57.749	1974-12-20	8.6	9811

2.7. Data Exploration

- Analyzed the data set to identify its details well.

Data Exploration

In [3]:
```
# Check movies info
movie_list.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 10000 non-null  int64
 1   title              10000 non-null  object
 2   genre              9997 non-null   object
 3   original_language  10000 non-null  object
 4   overview           9987 non-null   object
 5   popularity         10000 non-null  float64
 6   release_date       10000 non-null  object
 7   vote_average       10000 non-null  float64
 8   vote_count         10000 non-null  int64
dtypes: float64(2), int64(2), object(5)
memory usage: 703.3+ KB
```

2.8.Feature Engineering

- Checked the column names of the data set and selected the features needed for content-based filtering, based on the movie genre.

Feature Engineering

In [6]:
```
# check columns
movie_list.columns
```

Out[6]:
```
Index(['id', 'title', 'genre', 'original_language', 'overview', 'popularity',
       'release_date', 'vote_average', 'vote_count'],
      dtype='object')
```

In [7]:
```
# select features
movie_list = movie_list[['id', 'title',  'genre', 'vote_average']]
movie_list
```

Out[7]:

	id	title	genre	vote_average
0	278	The Shawshank Redemption	Drama,Crime	8.7
1	19404	Dilwale Dulhania Le Jayenge	Comedy,Drama,Romance	8.7
2	238	The Godfather	Drama,Crime	8.7
3	424	Schindler's List	Drama,History,War	8.6
4	240	The Godfather: Part II	Drama,Crime	8.6
...	...	...	...	...
9995	10196	The Last Airbender	Action,Adventure,Fantasy	4.7
9996	331446	Sharknado 3: Oh Hell No!	Action,TV Movie,Science Fiction,Comedy,Adventure	4.7
9997	13995	Captain America	Action,Science Fiction,War	4.6
9998	2312	In the Name of the King: A Dungeon Siege Tale	Adventure,Fantasy,Action,Drama	4.7
9999	455957	Domino	Thriller,Action,Crime	4.6

3. Similarity Computation

Content – based Filtering (Based on the Movie Title)

- Initialized a count vectorizer with a maximum of 10 000 features and English stop words.

In [15]:
```
#Initialized a count vectorizer with a maximum of 10 000 features and English stop words.
countVect = CountVectorizer(max_features = 10000, stop_words='english')
countVect
```

Out[15]:
```
CountVectorizer
CountVectorizer(max_features=10000, stop_words='english')
```

- Applied the count vectorizer to the ‘description’ column, converting the text data into a matrix of token counts (represented as a dense array).

In [17]:
```
# Convert 'description' column into vectors
vector = countVect.fit_transform(newDf['description'].values.astype('U')).toarray()
vector.shape
```

Out[17]:
```
(10000, 10000)
```

- Computed the cosine similarity between movies based on the features derived from count vectorizer. The resulting 'similarity' matrix indicates how similar each pair of movies is.

```
In [18]: # Check the similarity between movies
         cosSimilarity = cosine_similarity(vector)
         cosSimilarity

Out[18]: array([[1.        , 0.05634362, 0.12888482, ..., 0.07559289, 0.11065667,
                 0.06388766],
                [0.05634362, 1.        , 0.07624929, ..., 0.        , 0.03636965,
                 0.        ],
                [0.12888482, 0.07624929, 1.        , ..., 0.02273314, 0.06655583,
                 0.08645856],
                ...,
                [0.07559289, 0.        , 0.02273314, ..., 1.        , 0.03253   ,
                 0.02817181],
                [0.11065667, 0.03636965, 0.06655583, ..., 0.03253   , 1.        ,
                 0.0412393 ],
                [0.06388766, 0.        , 0.08645856, ..., 0.02817181, 0.0412393 ,
                 1.        ]])
```

Content – based Filtering (Based on the Genre)

- Initialized a TfidfVectorizer to convert the features into a matrix of TF-IDF features.
- Fitted the vectorizer to the 'genre' column to transform it into a sparse matrix of TF-IDF features.
- Calculated the cosine similarity between each pair of movies based on their TF-IDF feature vectors.
- 'tfidf_matrix' contains the TF-IDF features for each movie's genre.

Similarity Computation

```
In [8]: from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.metrics.pairwise import linear_kernel

In [9]: # TF-IDF Vectorizer
        tfidf_vectorizer = TfidfVectorizer(stop_words='english')
        tfidf_matrix = tfidf_vectorizer.fit_transform(movie_list['genre'])

In [10]: # Compute the cosine similarity
         genre_cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

4. Recommendation Function

Content – based Filtering (Based on the Movie Title)

In [28]:
```
# Build a recommendation function to recommend movies

def recommend(movies):
    # Get the each index and access the title of each movie
    index = newDf[newDf['title'] == movies].index[0]

    # create a list of similarity
    distance = sorted(list(enumerate(cosSimilarity[index])), reverse=True, key=lambda vector:vector[1])

    for i in distance[0:5]:
        print(newDf.iloc[i[0]].title)
```

In [30]:
```
recommend("The Avengers")
```

```
The Avengers
The Humanity Bureau
Allegiant
The Matrix Resurrections
Kingsman: The Golden Circle
```

In [31]:
```
recommend("Dilwale Dulhania Le Jayenge")
```

```
Dilwale Dulhania Le Jayenge
A Passage to India
The Manual of Love
The Cameraman
The Graduate
```

Content – based Filtering (Based on the Genre)

Recommendation

In [11]:
```
# Funciton to Recommend movies based on the genre

def recommend_movies_genres(genre_string, genre_cosine_sim=genre_cosine_sim):
    # Convert the comma-separated string of genres to a list
    genres = [genre.strip() for genre in genre_string.split(',')]

    # Get the indices of movies that have all of the selected genres
    genre_indices = movie_list[movie_list['genre'].apply(lambda x: all(genre in x for genre in genres))].index

    # Sort movies based on 'vote_average' in descending order
    sorted_indices = movie_list.loc[genre_indices].sort_values(by='vote_average', ascending=False).index

    # Return the top 5 highest-rated movies that fall into all selected genres
    return movie_list.loc[sorted_indices[:5], ['id', 'title', 'vote_average']]
```

In [12]:
```
# Checking the built model
recommend_movies_genres('Crime')
```

Out[12]:

	id	title	vote_average
0	278	The Shawshank Redemption	8.7
2	238	The Godfather	8.7
4	240	The Godfather: Part II	8.6
13	497	The Green Mile	8.5
15	155	The Dark Knight	8.5

```
In [13]: recommend_movies_genres('Action,Science')
```

Out[13]:

	id	title	vote_average
34	283566	Evangelion: 3.0+1.0 Thrice Upon a Time	8.4
41	18491	Neon Genesis Evangelion: The End of Evangelion	8.4
42	1891	The Empire Strikes Back	8.4
54	27205	Inception	8.4
35	324857	Spider-Man: Into the Spider-Verse	8.4

```
In [14]: recommend_movies_genres('Comedy,Drama,Romance')
```

Out[14]:

	id	title	vote_average
1	19404	Dilwale Dulhania Le Jayenge	8.7
20	13	Forrest Gump	8.5
57	901	City Lights	8.4
85	572154	Rascal Does Not Dream of a Dreaming Girl	8.3
92	522924	The Art of Racing in the Rain	8.3

```
In [15]: recommend_movies_genres('Action,TV Movie,Science Fiction,Comedy,Adventure')
```

Out[15]:

	id	title	vote_average
9905	438970	Sharknado 5: Global Swarming	4.9
9996	331446	Sharknado 3: Oh Hell No!	4.7

5. Deployment

- Saved the models.
  - For content-based filtering based on the title.

Save the model

```
In [32]: #movies file
         pickle.dump(newDf, open('movies_list.pkl', 'wb'))

         #similarity file
         pickle.dump(cosSimilarity, open('similarity.pkl', 'wb'))
```

  - For content-based filtering based on the genre.

Save the model

```
In [16]: # Save the model
         import pickle

         #movie_list
         pickle.dump(movie_list, open('genre_movie_list.pkl', 'wb'))

         #similarities
         pickle.dump(genre_cosine_sim, open('genre_cosine_sim.pkl', 'wb'))
```

- Loaded the saved models and implemented the frontend.

```
import streamlit as st
import pickle
import requests
import streamlit.components.v1 as components

# Load the models

#title - content based
movies = pickle.load(open("C:/Users/Azmarah Rizvi/Desktop/IRWA Project/movies_list.pkl", 'rb'))
similarity = pickle.load(open("C:/Users/Azmarah Rizvi/Desktop/IRWA Project/similarity.pkl", 'rb'))

#genre - content based
genre_sim = pickle.load(open("C:/Users/Azmarah Rizvi/Desktop/IRWA Project 3/genre_cosine_sim.pkl", 'rb'))
genre_movies = pickle.load(open("C:/Users/Azmarah Rizvi/Desktop/IRWA Project 3/genre_movie_list.pkl", 'rb'))

movies_list = movies['title'].values
```

```python
# fetch the posters
def fetch_poster(movie_id):
    url = "https://api.themoviedb.org/3/movie/{}?api_key=80321b186ae2ff767c6ef9499a9bae85&language=en-US".format(movie_id)
    data = requests.get(url)

    data=data.json()
    poster_path = data['poster_path']
    full_path = "https://image.tmdb.org/t/p/w500/"+poster_path
    return full_path
```

```python
# Title - Content based - recommendation function
def recommend(movie):
    # Get the each index and access the title of each movie
    index = movies[movies['title'] == movie].index[0]

    # create a list of similarity
    distance = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda vector:vector[1])

    # store the recommended movies inside a list
    recommend_movie = []

    # store the movie posters inside a list
    recommend_poster = []

    for i in distance[1:6]:

        #for posters
        movies_id = movies.iloc[i[0]].id
        recommend_poster.append(fetch_poster(movies_id))

        recommend_movie.append(movies.iloc[i[0]].title)

    return recommend_movie, recommend_poster
```

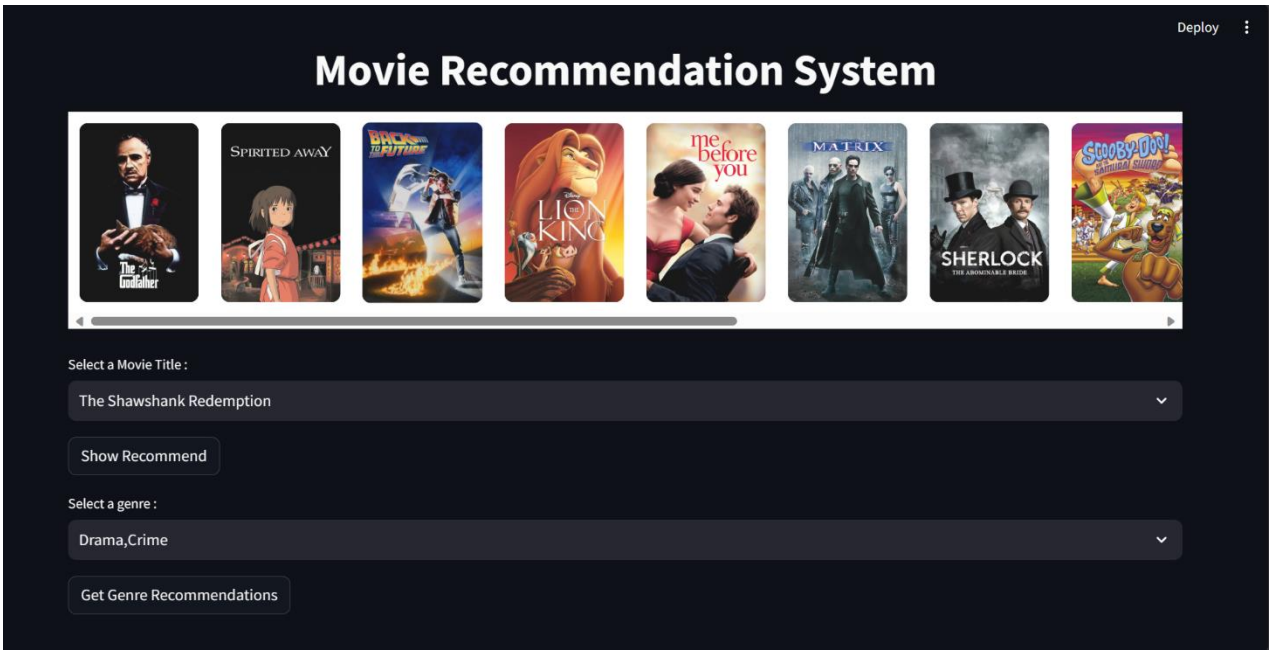
```python
# Genre - content based
def recommend_movies_genres(genre_string, genre_sim=genre_sim):
    # Convert the comma-separated string of genres to a list
    genres = [genre.strip() for genre in genre_string.split(',')]

    # Get the indices of movies that have all of the selected genres
    genre_indices = genre_movies[genre_movies['genre'].apply(lambda x: all(genre in x for genre in genres))].index

    # Sort movies based on 'vote_average' in descending order
    sorted_indices = genre_movies.loc[genre_indices].sort_values(by='vote_average', ascending=False).index

    # Return the top 5 highest-rated movies that fall into all selected genres
    return genre_movies.loc[sorted_indices[:5], ['id', 'title', 'vote_average']]
```

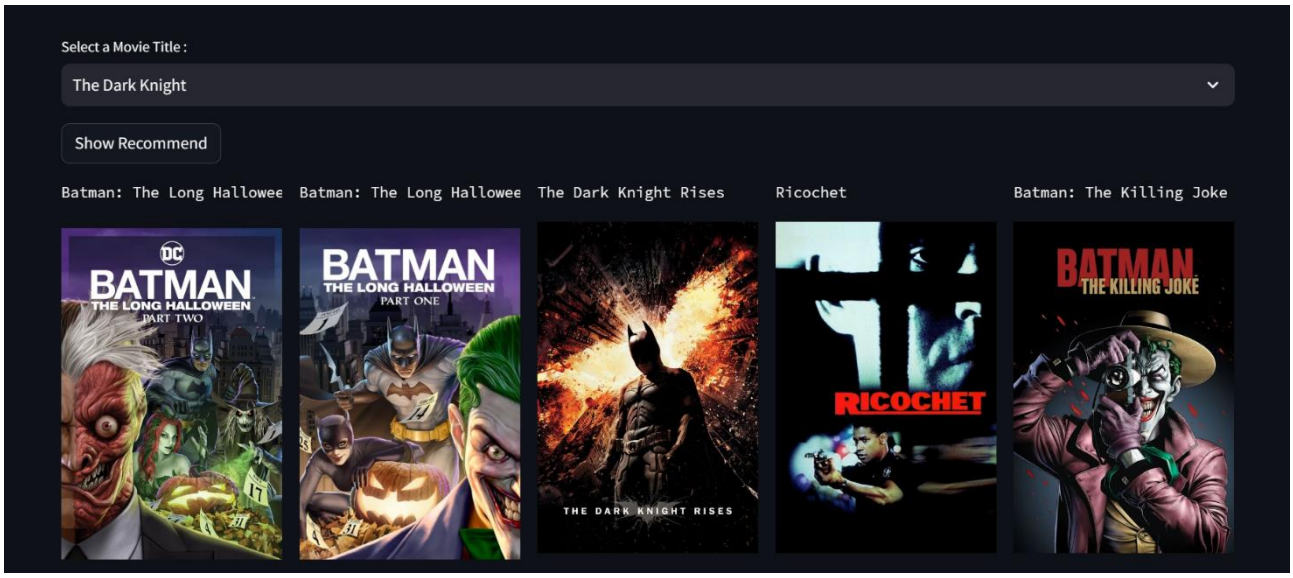
```python
st.markdown("<h1 style='text-align: center;'>Movie Recommendation System</h1>", unsafe_allow_html=True)

imageCarouselComponent = components.declare_component("image-carousel-component", path="C:/Users/Azmarah Rizvi/Desktop/IRWA Project/frontend/public")


imageUrls = [
    fetch_poster(1652),
    fetch_poster(299536),
    fetch_poster(17455),
    fetch_poster(2830),
    fetch_poster(429422),
    fetch_poster(9722),
    fetch_poster(13972),
    fetch_poster(240),
    fetch_poster(155),
    fetch_poster(598),
    fetch_poster(914),
    fetch_poster(255709),
    fetch_poster(572154)

    ]


imageCarouselComponent(imageUrls=imageUrls, height=200)
```

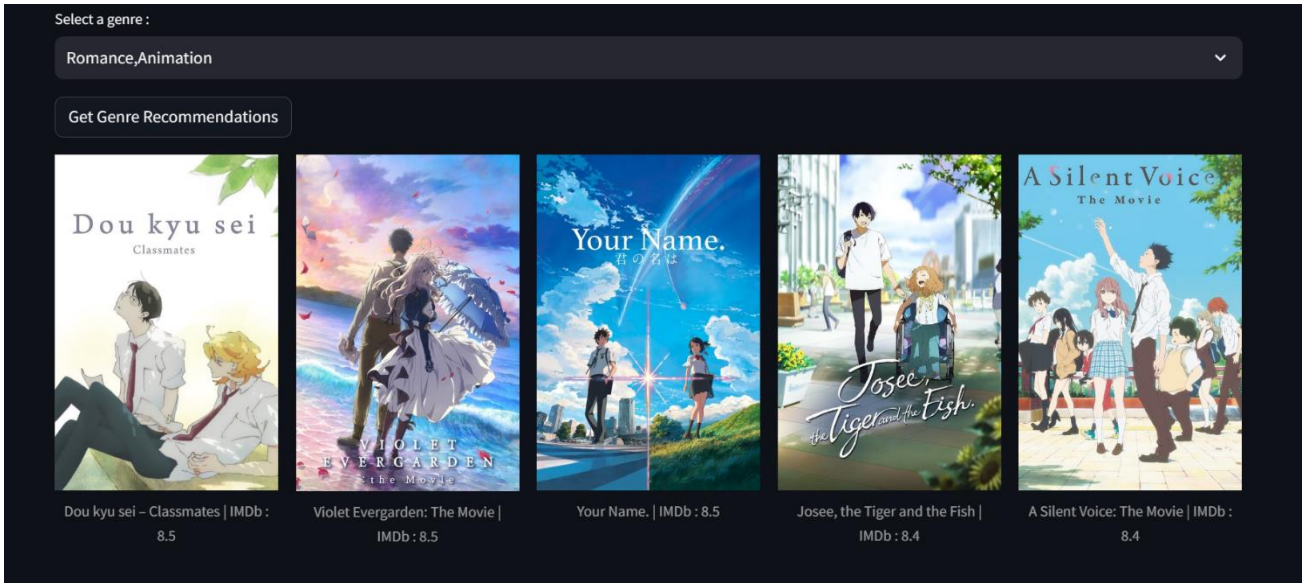
```python
def main():
    # Title - content based
    selectvalue = st.selectbox("Select a Movie Title :", movies_list)

    if st.button("Show Recommend"):
        movie_name, movie_poster = recommend(selectvalue)
        #display
        col1, col2, col3, col4, col5 = st.columns(5)
        with col1:
            st.text(movie_name[0])
            if movie_poster[0] is not None:
                st.image(movie_poster[0])
            else:
                st.warning("Poster not available")
        with col2:
            st.text(movie_name[1])
            if movie_poster[1] is not None:
                st.image(movie_poster[1])
            else:
                st.warning("Poster not available")
        with col3:
            st.text(movie_name[2])
            if movie_poster[2] is not None:
                st.image(movie_poster[2])
            else:
                st.warning("Poster not available")
        with col4:
            st.text(movie_name[3])
            if movie_poster[3] is not None:
                st.image(movie_poster[3])
            else:
                st.warning("Poster not available")
        with col5:
            st.text(movie_name[4])
            if movie_poster[4] is not None:
                st.image(movie_poster[4])
            else:
                st.warning("Poster not available")
```

```python
    # genre-based recommendation

    # selecting genres
    selected_genre = st.selectbox('Select a genre :', genre_movies['genre'].str.split(', ').explode().unique())

    # Button for genre-based recommendation
    if st.button("Get Genre Recommendations"):
        # Call the recommend_movies_genres function with all required arguments
        filtered_movies_genre = recommend_movies_genres(selected_genre)

        if not filtered_movies_genre.empty:
            # Display posters for recommended movies
            row_posters = st.columns(5)

            for idx, (_, movie_row) in enumerate(filtered_movies_genre.head(5).iterrows(), start=1):
                movie_title = movie_row['title']
                vote_average = movie_row['vote_average']

                # Fetch poster path using the movie ID
                poster_path = fetch_poster(movie_row['id'])

                # Display posters, title, and vote average in a single row
                with row_posters[idx - 1]:
                    st.image(poster_path, caption=f"{movie_title} | IMDb : {vote_average}", use_column_width=True)
        else:
            st.warning('No movies found for the selected genre.')


if __name__ == "__main__":
    main()
```

5.1. Web View


Deploy
Movie Recommendation System
Select a Movie Title :
The Shawshank Redemption
Show Recommend
Select a genre :
Drama,Crime
Get Genre Recommendations


Select a Movie Title :
The Dark Knight
Show Recommend
Batman: The Long Hallowee
Batman: The Long Hallowee
The Dark Knight Rises
Ricochet
Batman: The Killing Joke


Select a genre :
Romance,Animation
Get Genre Recommendations
Dou kyu sei – Classmates | IMDb : 8.5
Violet Evergarden: The Movie | IMDb : 8.5
Your Name. | IMDb : 8.5
Josee, the Tiger and the Fish | IMDb : 8.4
A Silent Voice: The Movie | IMDb : 8.4

Conclusion

We developed a content-based movie recommendation system that depends on the supplied movie name and genre through the analysis of movie data sets. Users can utilize the genres or movie names in this recommendation system. It offers specific movie recommendations, illuminating the possibility of individualized film choices.

Our algorithm effectively matches user choices with comparable movies by examining the elements and features of movies, such as genre, cast, director, overviews, and IMDb rating.

We utilized the strength of machine learning and natural language processing in this endeavor. The future of personalized movie discovery that caters to a variety of likes and inclinations in the ever-expanding world of cinema holds immense potential for content-based recommendation systems.

We successfully implemented it in Stream lit (using Python language) in a user-friendly way, allowing users to choose a movie using a drop-down menu and having it display recommended movies in order of IMDb ratings.

In the end, our content-based movie recommendation engine has the potential to completely transform how people find and enjoy movies by streamlining and improving the process.

Appendix

GitHub Link: https://github.com/IT21468360/Movie-Recommendation-System

Link to the dataset: https://www.kaggle.com/datasets/ahsanaseer/top-rated-tmdb-movies-10k?fbclid=IwAR2MpWrWpcw2QNCv_FZg2l0sjBh9xAvhrqtnZBO9K-QS6PHI1aHkdB6qLa0