Introduction to Data Science

Movie Recommendations - Explore a World of Cinematic Brilliance

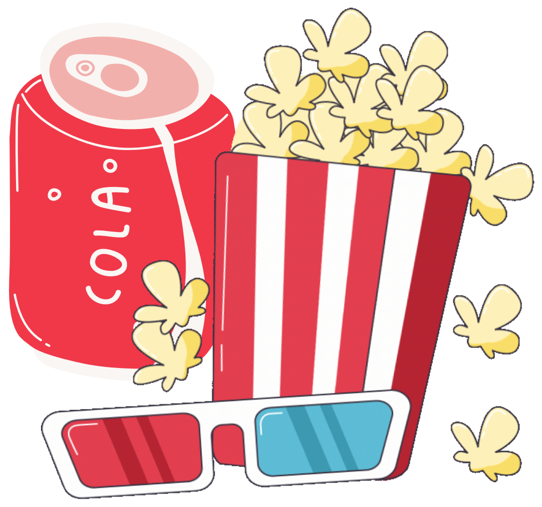
Group 1

Tran Huy Ban (21127012)

Tran Nguyen Huan (21127050)

Nguyen Minh Quan (21127143)

Le Anh Thu (21127175)



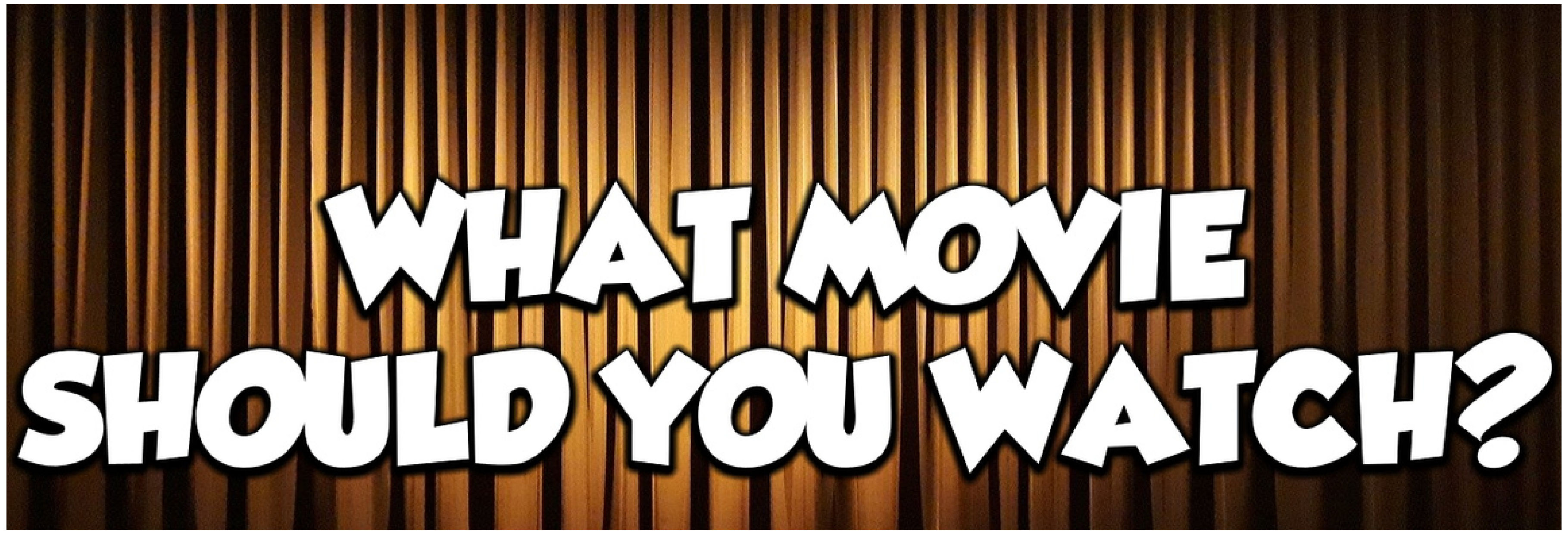
fit@hcmus

Contents

Github Trello API Deploy App

Overview

Looking for your next movie night delight? Our recommendation engine, powered by The Movie Database (TMDB) API, is here to guide you through a curated list of cinematic gems that promise both captivating overviews and significant impact.



Overview

Why the movies of our recommendations should be on your must-watch list?

Diverse Genres:

- Each recommended movie comes with a unique and compelling overview, providing a glimpse into the storyline.
- Our selection covers a diverse range of genres, from heartwarming dramas to spine-chilling thrillers.

Perfect Choices:

- Whether you're in the mood for a gripping narrative or a light-hearted adventure, our recommendations offer intriguing synopses to help you make the perfect choice.

Overview

Why the movies of our recommendations should be on your must-watch list?

<u>Lasting Impact:</u>

- Beyond just entertaining, these movies have left a lasting impact on audiences worldwide.
- They have resonated with viewers, sparking discussions and leaving a mark on the world of cinema.

<u>Cultural Contribution:</u>

- Prepare to embark on an unforgettable journey as you explore films that have not only earned critical acclaim but have also contributed to the cultural tapestry of the film industry.

1. Data Collection

a. Crawl Movies data

Step 1: Set up

- Defining the API base URL, the specific endpoint for top-rated movies, the file path for data storage, and the initial API request URL.

Step 2: Data Retrieval

- For each page, there will be 50 movies on that page, ranked from high to low score. So, to get a complete dataset with 1000 rows of data, we need a minimum of 20 pages. This step will use a loop to get data for each page.

Step 3: API Request and Response Handling

- To get data, send an HTTP GET request to the TMDB API using the pre-initialized URL.
- On success, the response will be returned as JSON to be stored in the corresponding variable

Step 4: Storage data

- Data crawled will be saved to folder Data with the name ”movies.csv” after the crawled.

1. Data Collection

b. Crawl Casts data

We have data about the movies, including basic information such as name, release date, score, number of votes, etc. However, to make our dataset richer in information, we will get a list of the movie's actors.

Step 1:

- Come to credits page with **each** movie in dataset.

Step 2:

- Get all informations about the casts of the movie (similar to get movies data)

Step 3:

- Store casts data with the corresponding movie and save to csv file.

2. Data Pre-processing

Handle null value

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   adult              1000 non-null   bool
 1   backdrop_path      1000 non-null   object
 2   genre_ids          1000 non-null   object
 3   id                 1000 non-null   int64
 4   original_language  1000 non-null   object
 5   original_title     1000 non-null   object
 6   overview           1000 non-null   object
 7   popularity         1000 non-null   float64
 8   poster_path        1000 non-null   object
 9   release_date       1000 non-null   object
 10  title              1000 non-null   object
 11  video              1000 non-null   bool
 12  vote_average       1000 non-null   float64
 13  vote_count         1000 non-null   int64
 14  casts              1000 non-null   object
dtypes: bool(2), float64(2), int64(2), object(9)
memory usage: 103.6+ KB
```

Handle duplicates

```
duplicates = df[df.duplicated()]
print(f"\nNumber of duplicates: {len(duplicates)}")
```

```
Number of duplicates: 8
```

```
df = df.drop_duplicates()
df = df.reset_index()
```

```
duplicates = df[df.duplicated()]
print(f"\nCheck number of duplicates again: {len(duplicates)}")
```

```
Check number of duplicates again: 0
```

2. Data Pre-processing

Feature selection

- Original data has 15 columns, we don't need to parse the entire column. The important columns we select are:

1. genre_ids (id of movie genre) 2. id (id of movie) 3. overview (brief overview of the movie)
4. popularity (popularity score) 5. release_date (movie release time)
6. title (title of movie) 7. vote_average (the average voting score)
8. vote_count (the number of votes) 9. casts (details of the casts)

Arrange columns

- Rearrange the order of the columns to make them more accurate.

1. id 2. genre_ids 3. title 4. overview 5. popularity 6. release_date
7. vote_average 8. vote_count 9. casts

2. Data Pre-processing

Change value in columns

- In the release date column, we only extract the year, we do not need to get the full date.
- We will use the function of pandas to remove the day and month, keeping only the year in the release_date column. Then rename that column by release_year.

Extract all actors' names in the casts column

- The current casts column has a dictionary data type and contains a lot of information about the movie's cast (such as gender, adult, id,...) but we are only interested in the actor's name and do not need other information.
- Create a pattern and use the function findall of regex library to find all the actors' names. After extracting, only the names of the actors are retained in the column.

2. Data Pre-processing

Correct data type

- The data type of the genre_ids column is an object with a number (each number is the id corresponding to the movie genre), so we need to clarify what the movie genre is. We can get the category name from API:

28: 'Action', 12: 'Adventure', 16: 'Animation', 35: 'Comedy', 80: 'Crime',...

- We change all the numbers of genre id to movie genre string and then rename the column to genre_names.

3. Data Exploration

Basic level description

1. What is the meaning of each column?

#	Field Name	Description
1	`id`	A unique identifier for each movie.
2	`title`	Title of the movie.
3	`popularity`	A numeric quantity specifying the movie popularity.
4	`release_year`	The year on which it was released
5	`vote_average`	average ratings the movie recieved.
6	`vote_count`	the count of votes recieved.
7	`casts`	The name of lead and supporting actors.
8	`genre_names`	The genre of the movie, Action, Comedy ,Thriller etc.
9	`overview`	A brief description of the movie.

2. How many rows and columns are there in the data?

```
Number of rows: 992
Number of columns: 9
```

3. What is the data type of each column?

```
id                 int64
title             object
overview          object
popularity       float64
release_year       int32
vote_average     float64
vote_count         int64
casts             object
genre_names       object
dtype: object
```

3. Data Exploration

Basic level description

4. What is the distribution of the numeric data in each column?

	id	popularity	release_year	vote_average	vote_count
count	9.920000e+02	992.000000	992.000000	992.000000	992.000000
mean	1.910366e+05	43.739479	1995.470766	7.897981	4124.322581
std	2.628580e+05	102.621345	25.566382	0.232909	5782.546609
min	1.100000e+01	0.600000	1902.000000	7.588000	300.000000
25%	1.116250e+03	14.841000	1977.000000	7.705000	598.750000
50%	2.096650e+04	23.201000	2004.000000	7.850000	1457.000000
75%	3.783735e+05	43.064750	2017.000000	8.044750	4993.500000
max	1.076364e+06	2287.202000	2023.000000	8.709000	34775.000000

3. Data Exploration

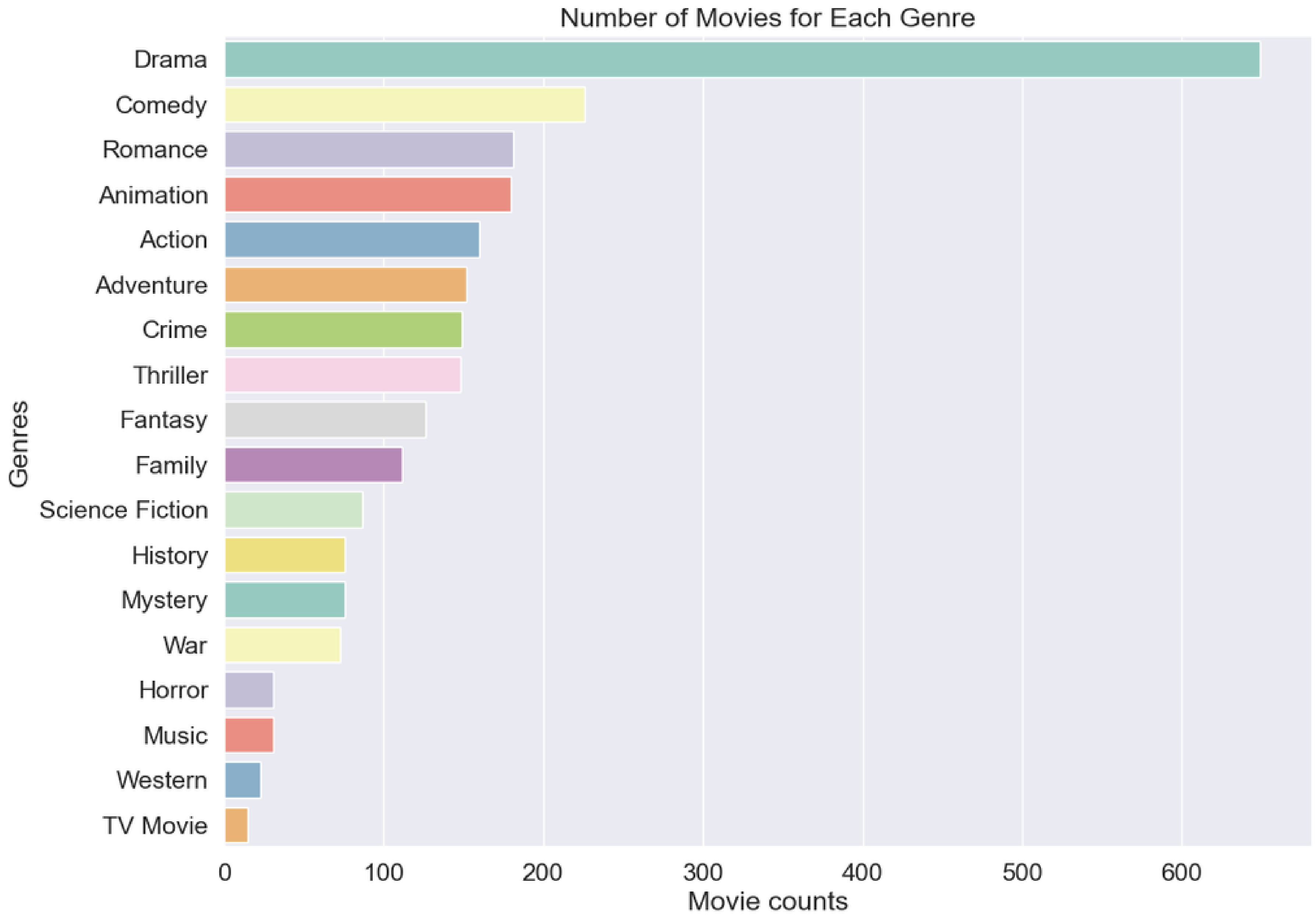
Further Exploration

In this section, we will ask 5 questions to dig deeper into the data.

- **Question 1:** How many movies based on their genres were produced? Which genres have the most movies produced?
- **Question 2:** What is the average vote distribution across different movie genres?
- **Question 3:** Who are the most influential actors across movies, and what is their distribution?
- **Question 4:** How do the number of votes and vote scores change over each period of year?
- **Question 5:** Is there a relationship between popularity and the number of ratings of movies? Does popularity affect the number of votes?

3. Data Exploration

Question 1: How many movies based on their genres were produced? Which genres has the most movies produced?

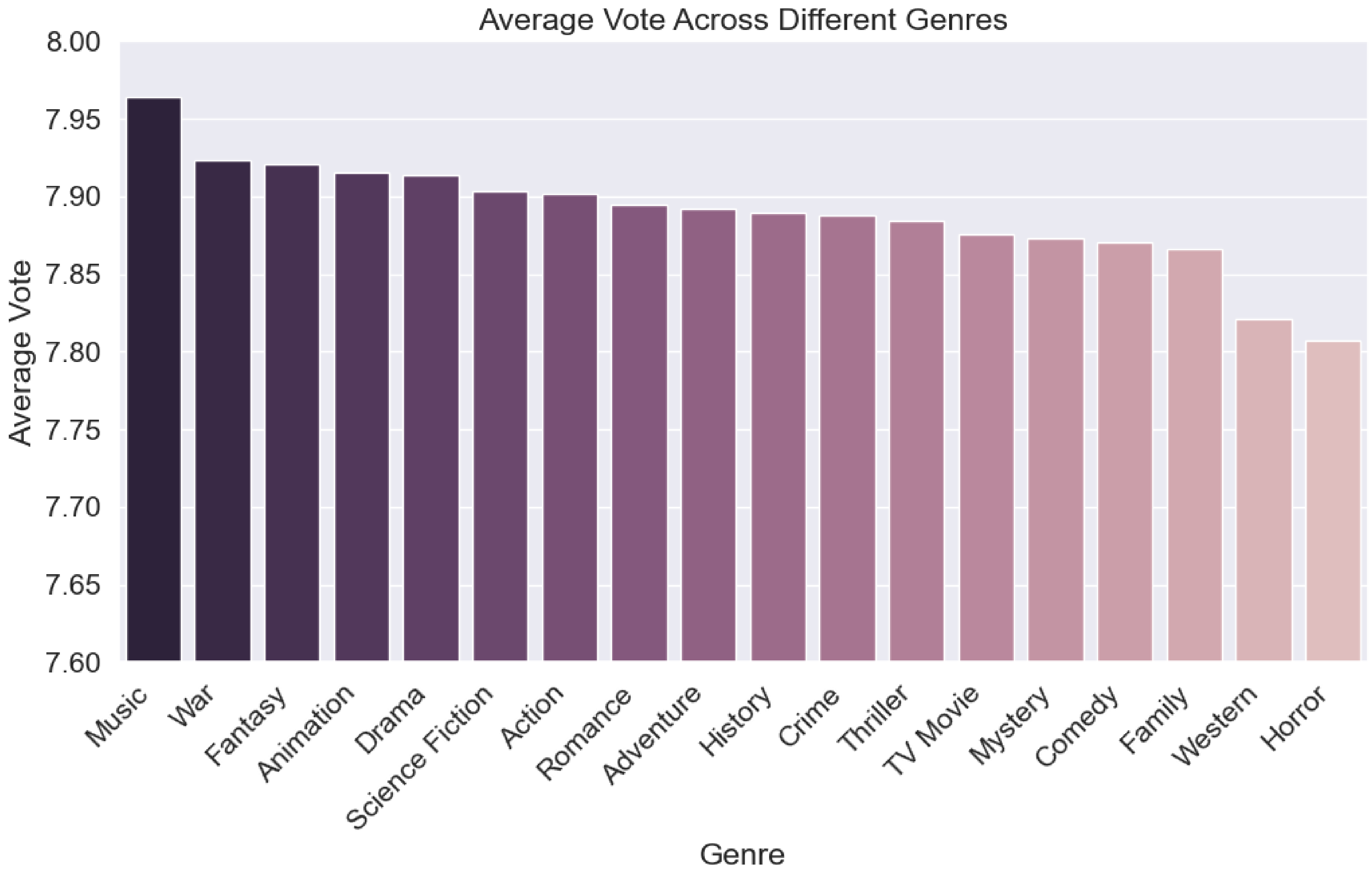


Observation:

- Drama is the most popular genre
- Horror, Music, Western and TVshow are the genres with the fewest movies
- The difference in drama genre is quite significant and shows that this is the favorite genre of viewers

3. Data Exploration

Question 2: What is the average vote distribution across different movie genres?



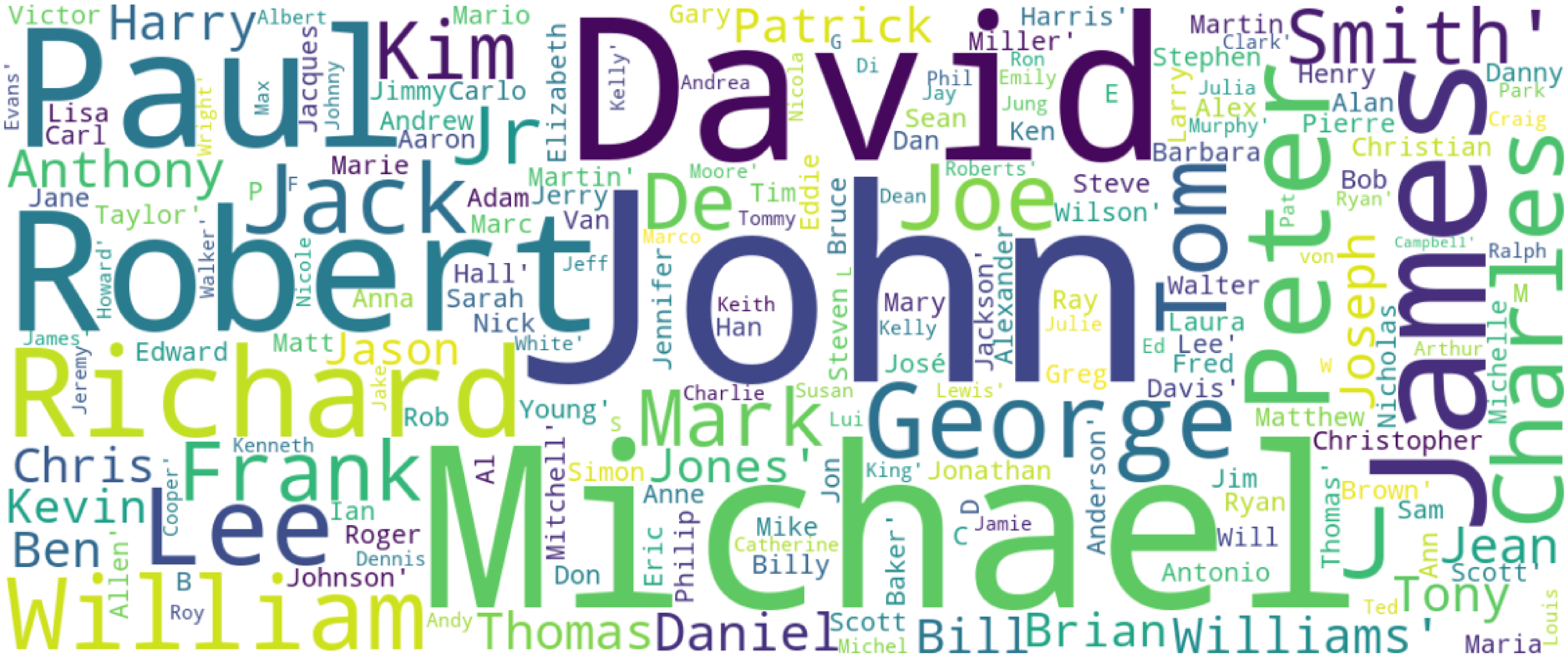
Observation:

- Minimal variation in audience scores across genres, ranging from 7.8 to 8.
- Music genre stands out with the highest ratings, indicating strong audience satisfaction.
- War and Fantasy genres closely follow, suggesting significant audience interest in these themes.
- Lower scores for Western and Horror genres imply potential difficulty in appealing to a broader audience.

3. Data Exploration

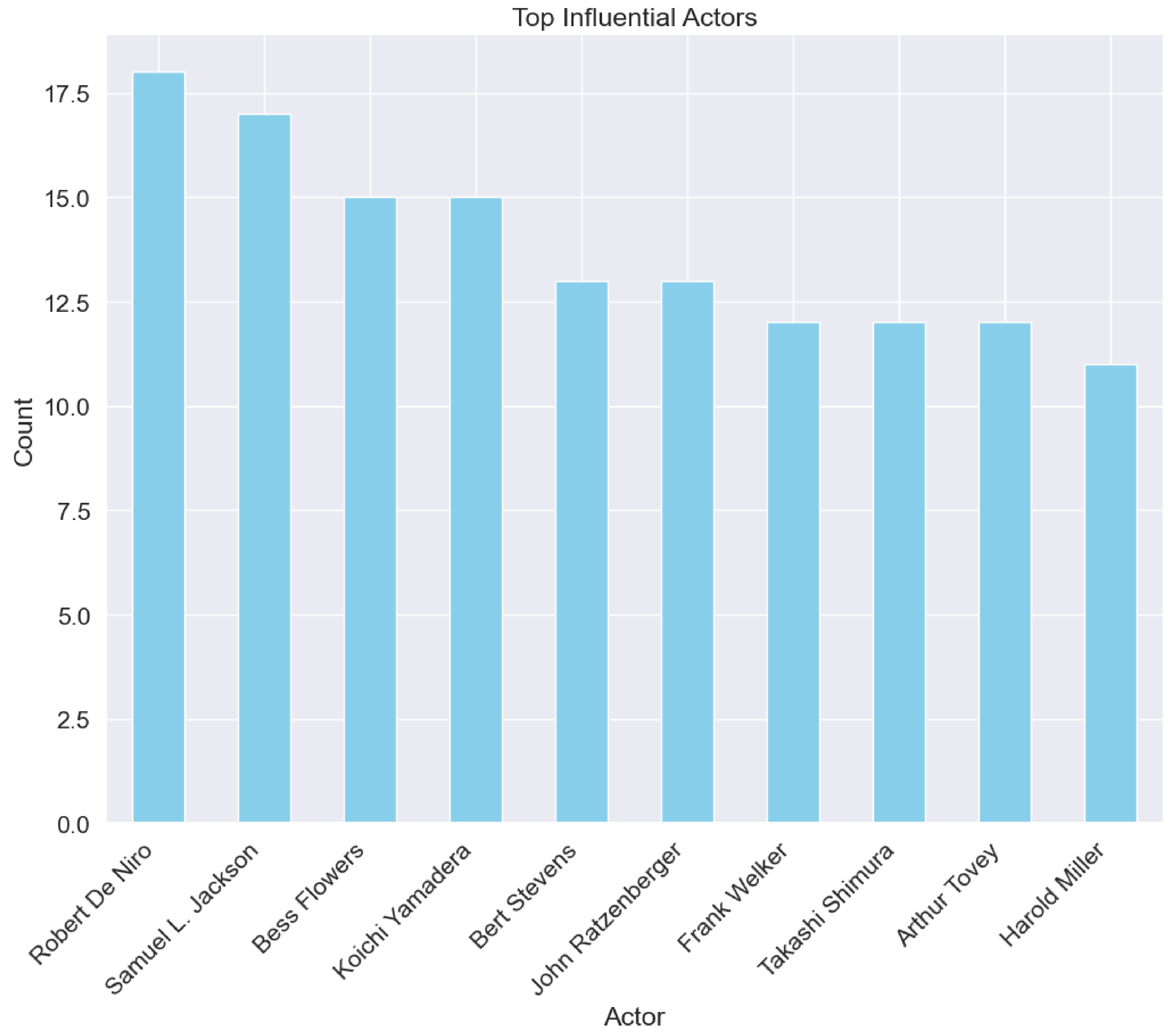
Question 3: Who are the most influential actors across movies, and what is their distribution?

Word Cloud of Most Influential Actors



3. Data Exploration

Question 3: Who are the most influential actors across movies, and what is their distribution?

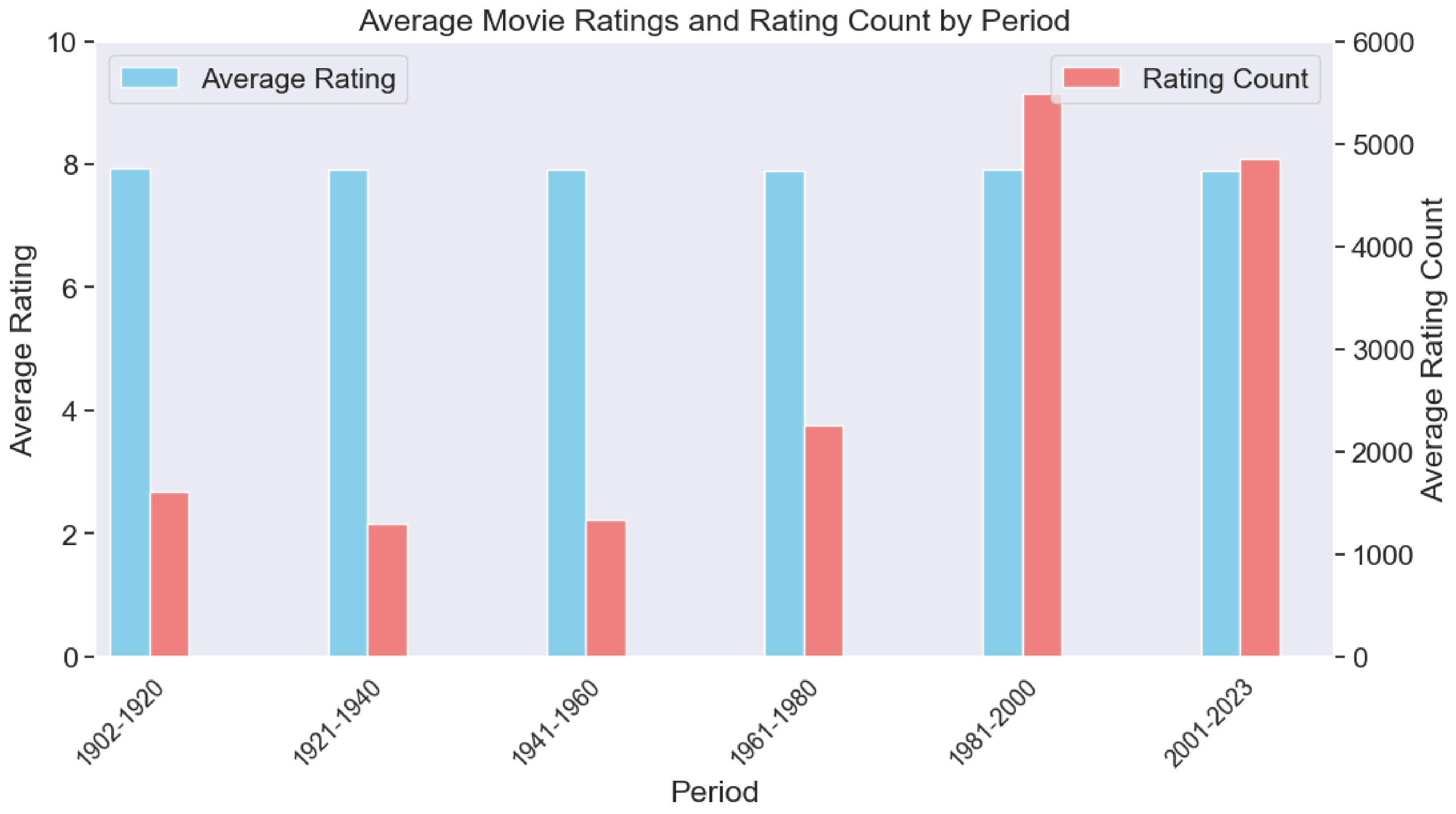


Observation:

- Notable names like Michael, Robert, ... dominate the word cloud, indicating their frequent association with movies.
- Robert De Niro emerges as the leading actor, consistently appearing in numerous films, making him a reliable choice for filmmakers.
- Samuel L. Jackson holds a notable position as the second most influential actor, although not surpassing Robert De Niro.
- Actors such as Bess Flowers and Koichi Yamadera have a lower movie count, suggesting a less frequent but still noteworthy presence in films.

3. Data Exploration

Question 4: How do the number of votes and vote scores change over each period of year?



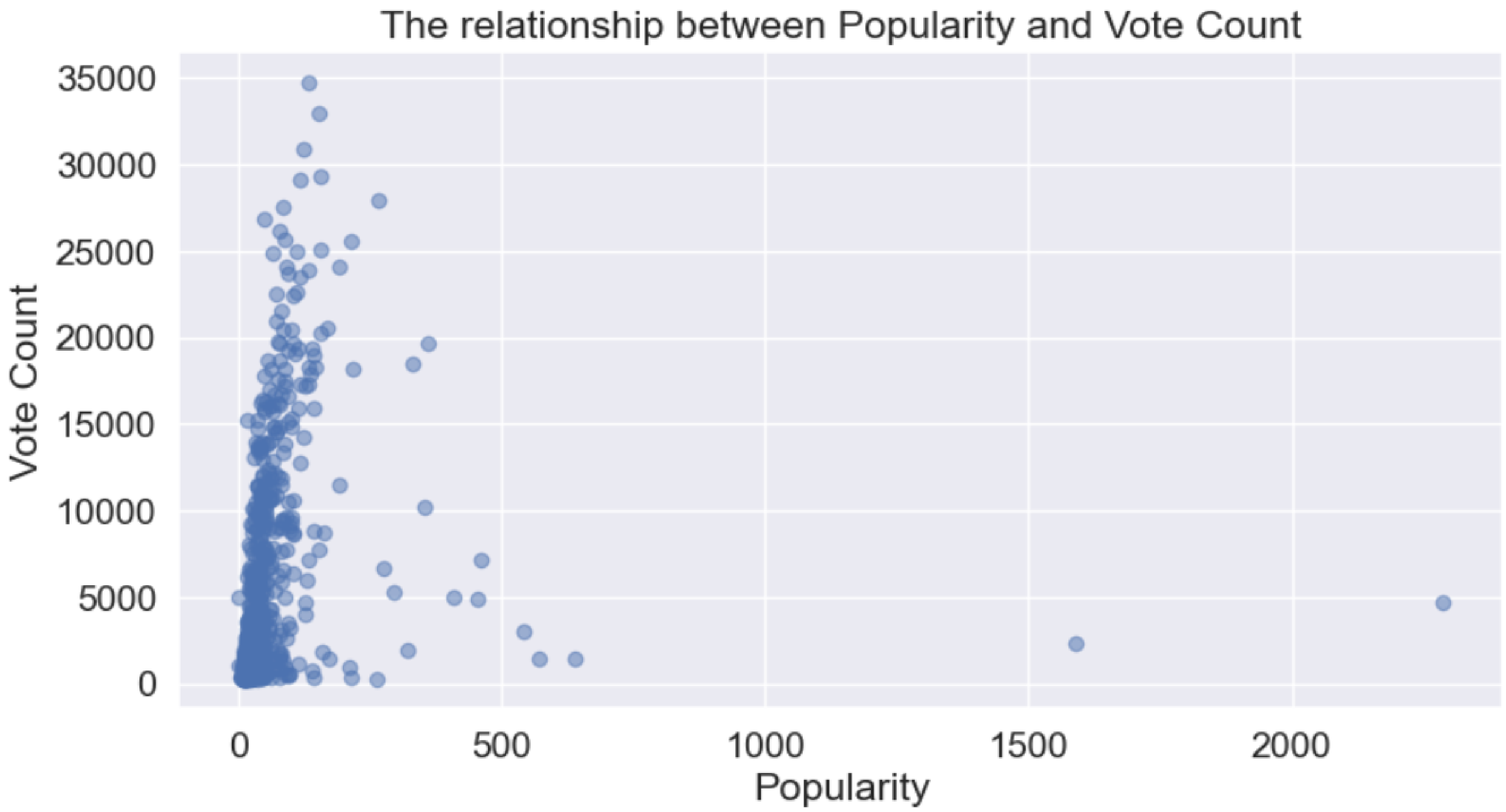
Observation

- Slight variations across periods; 1961-1980 has the highest, 2001-2023 the lowest.
- Significant increase over time, with peaks in 1981-2000 and 2001-2023, indicating growing audience participation.
- Stable quality ratings, but varying vote counts suggest changing audience engagement and accessibility.

3. Data Exploration

Question 5: Is there a relationship between popularity and the number of ratings of movies?

Does popularity affect the number of votes?



Observation

- Positive correlation between the number of votes and popularity, but not strong.
- Some less popular films get many votes, indicating a dedicated fan base, some highly popular movies do not have many reviews.
- Showing that people do not only review movies based on their popularity. They share their feelings and connections, making movie ratings more personal and subjective.

4. Data Modeling

Introduction

- The recommendation system is a hot cake in data science now. A true recommendation system can change the outcome of any business policy and catch the market easily.

Types of recommendation system

- **Content Based:** Content based filtering actually finds similarities between two movies based on the movie's actor, producer, genre, content, and so on.
- **Collaborative Based:** Collaborative filtering basically finds the similarity between two users.

*We will focus on a **Content-Based Recommendation System** utilizing metadata such as genre_names, casts, title, and overview to understand user preferences and recommend movies or TV shows accordingly.*

4. Data Modeling

Plot description based Recommender

- We'll calculate similarity scores between movies using their plot descriptions found in the **overview** feature of our dataset. Then, we'll provide movie recommendations based on these similarity scores.

Technical steps:

- Create a **TF-IDF** (Term Frequency-Inverse Document Frequency) **matrix** for the overview column.
- Calculate the **Cosine similarity** - a numeric quantity that denotes the similarity between two movies.

4. Data Modeling

Plot description based Recommender

- Enter the movie title and get recommendations:

```
get_overview_based_recommendations('The Dark Knight')
```

```
The Dark Knight Rises
Batman: Under the Red Hood
The Batman
Batman: The Dark Knight Returns, Part 2
Batman: The Dark Knight Returns, Part 1
```

```
get_overview_based_recommendations('Spider-Man: No Way Home')
```

```
Spider-Man: Into the Spider-Verse
Spider-Man: Across the Spider-Verse
The Hustler
Sound of Metal
Emancipation
```

- The current recommendation system accurately identifies movies with similar plot descriptions but lacks nuance.
- There is a need for an enhanced recommendation system to capture these nuances.

4. Data Modeling

Title, Genre and Cast Based Recommender

- Without a doubt, enhancing the metadata used by our recommender would lead to improved recommendations. We will focus on building a recommender that takes into account the following factors: **actors**, associated **genres**, and **movie titles**.

Technical steps:

- The steps are the same as what we did for our **Plot description based recommender** before.
- One important difference is that instead of **TF-IDF**, we opt for **CountVectorizer**. This choice ensures we do not undervalue actors/genres who have been part of more movies, which makes more intuitive sense.

4. Data Modeling

Title, Genre and Cast Based Recommender

- Enter the movie title and get recommendations:

```
get_tags_based_recommendations_cosine("Harry Potter and the Deathly Hallows: Part 2") # example 1
```

```
Harry Potter and the Deathly Hallows: Part 1
Harry Potter and the Half-Blood Prince
Harry Potter and the Order of the Phoenix
Harry Potter and the Prisoner of Azkaban
Harry Potter and the Chamber of Secrets
```

```
get_tags_based_recommendations_cosine("Avengers: Infinity War") # example 2
```

```
Avengers: Endgame
Guardians of the Galaxy Vol. 2
Guardians of the Galaxy
The Dark Knight
The Avengers
```

- Our recommender, enriched with additional metadata, has proven successful in capturing more information and providing arguably improved recommendations.

4. Data Modeling

Title, Genre and Cast Based Recommender

- *In the above steps, we used **Cosine similarity**. We can try to calculate the count matrix with **Euclidean similarity.***

```
get_tags_based_recommendations_euclidean('Harry Potter and the Deathly Hallows: Part 2')
```

```
Harry Potter and the Half-Blood Prince
Harry Potter and the Deathly Hallows: Part 1
Harry Potter and the Prisoner of Azkaban
Harry Potter and the Order of the Phoenix
Harry Potter and the Philosopher's Stone
```

```
get_tags_based_recommendations_euclidean('Avengers: Infinity War')
```

```
The Boy, the Mole, the Fox and the Horse
Piper
Far from the Tree
La Maison en Petits Cubes
Vincent
```

We can draw some comments:

- **Cosine similarity** emphasizes the direction of similarity, capturing thematic relationships.
- **Euclidean similarity** considers both direction and magnitude, introducing more variety in recommendations.
- The choice between **Cosine** and **Euclidean similarity** depends on the desired balance between thematic cohesion and diversity in recommendations.

4. Data Modeling

Formula

- **Cosine similarity**

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\cdot\|\vec{b}\|}$$

- **TF-IDF**

$$TFIDF\ score\ for\ term\ i\ in\ document\ j = TF(i,j) * IDF(i)$$

where

IDF = Inverse Document Frequency

TF = Term Frequency

$$TF(i,j) = \frac{\text{Term i frequency in document j}}{\text{Total words in document j}}$$

$$IDF(i) = \log_2\left(\frac{\text{Total documents}}{\text{documents with term i}}\right)$$

and

t = Term

j = Document

- **Euclidean similarity**

$$d_{L2}(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- **CountVectorizer**

Data = ['The', 'quick', 'brown', 'fox', 'jumps', 'over', ' the', 'lazy', 'dog']

	The	quick	brown	fox	jumps	over	lazy	dog
Data	2	1	1	1	1	1	1	1

5. Deploy model

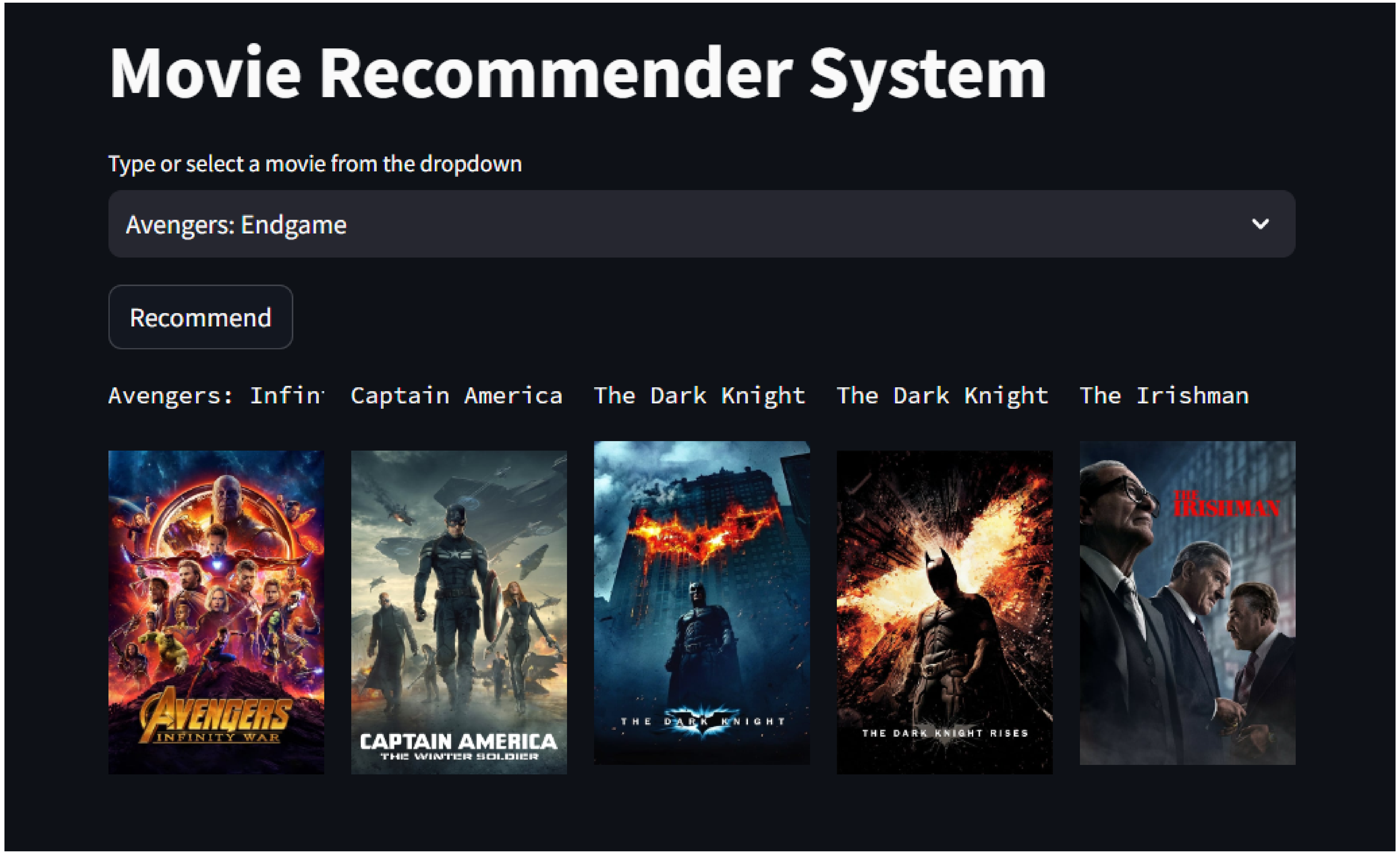
Result

- We have successfully developed the recommendation system and it has been deployed in the **App.py** file.
- The recommendation system model (**Title, Genre and Cast Based Recommender**), based on **cosine similarity**, has been deployed and is accessible through the **web interface**.
- We have integrated **Streamlit**, a Python library for creating interactive web applications, to build the user interface. Streamlit offers simplicity and flexibility in designing web applications with minimal code.
- Users can choose a movie title through a user-friendly interface on the web page.
- The recommended movies are displayed on the web page, providing users with clear and visually appealing movie titles and posters.

5. Deploy model

Link: https://i2ds-movie-recommendation-project.streamlit.app

The interface on our web page



6. References

- Cosine Similarity - Wikipedia
- Euclidean Distance - Wikipedia
- TF-IDF (Term Frequency-Inverse Document Frequency) - TFIDF.com
- Content-Based Movie Recommendation System - Medium
- CountVectorizer - scikit-learn Documentation

-The end-