A Project Report On

MOVIE RECOMMENDATION SYSTEM

Submitted for partial fulfillment of the requirement of the Main Project (4MCA1) for

MASTER OF COMPUTER APPLICATIONS

ಬೆಂಗಳೂರು ನಗರ ವಿಶ್ವವಿದ್ಯಾಲಯ
BENGALURU CITY UNIVERSITY
BE BOUNDLESS

Submitted by

SHREYAS K M (P18IW21S0022)

Name of Guide
Ms. Lakshmi Devi C

UNITY LOVE KNOWLEDGE SEARCH
ST. FRANCIS COLLEGE
KORAMANGALA, BENGALURU

ST. FRANCIS COLLEGE

MCA DEPARTMENT

Affiliated To Bengaluru City University

3rd Block, 8th Main, Koramangala

Bangalore-560034

ST. FRANCIS COLLEGE

CERTIFICATE

This is to certify that the report entitled "**MOVIE RECOMMENDATION SYSTEM"** embodies the original work by **SHREYAS K M (P18IW21S0022)** the fulfilment of the requirements for Main Project (**4MCA1**) for MCA, IVth Semester course during the academic semester from August 2023 to December 2023 as prescribed by Bengaluru City University.

Project Guide **Head of the Department**

Ms. Lakshmi Devi C **Dr. Nazura Javed Kutty**

Place:Bangalore

Date:

ST. FRANCIS COLLEGE

CERTIFICATE

This is to certify that the report entitled "**MOVIE RECOMMENDATION SYSTEM"** embodies the original work by **SHREYAS K M (P18IW21S0022)** the fulfilment of the requirements for Main Project (**4MCA1**) for MCA, IVth Semester course during the academic semester from August 2023 to December 2023 as prescribed by Bengaluru City University.

External Examiner

1._____

2._____

Place:Bangalore

Date:

Acknowledgement

We would like to express our gratitude and appreciation to all those who gave us the possibility to complete this report

We take pleasure in expressing gratitude towards our institution for providing us the opportunity to work on this project as a part of our course. We would like to express our gratitude and sincere thanks to our management, **Bro. Peter,** Director for providing us with a congenial environment for studies. We would like to express our gratitude to Principal, **Dr. R N Subba Rao** for providing the right atmosphere at the institute. We express our heart full thanks to **Dr. Nazura Javed Kutty**, Head of the Department, for being helpful through the project. We express our deep sense of gratitude to our project guide **Ms. Lakshmi Devi C**, Assistant Professor for her unfailing encouragement, valuable guidance and suggestions to us in the course of project work. We also express our heartfelt thanks to our parents for being supportive in all our activities without whom it wouldn't have been possible for us to complete our project.

Table of Contents

ACKNOWLEDGEMENT 3
1. ABSTRACT 6
2. LITERATURE REVIEW 7
3. PROBLEM STATEMENT 7
4. OBJECTIVE 8
5. METHODOLOGY 8
5.1 CONTENT-BASED FILTERING 8
5.2 POPULARITY-BASED RECOMMENDATIONS 9
5.3 COLLABORATIVE FILTERING 9
6. LIBRARIES AND FRAMEWORKS 10
7. DIAGRAM 12
7.1 ARCHITECTURE DIAGRAM 12
7.2 BLOCK DIAGRAM 13
7.3 FLOW CHART 14
7.4 USE CASE DIAGRAM 15
8. DATASET 16
9. EXPERIMENT 17
10. IMPLEMENTATION 23
10.1 CONTENT-BASED RECOMMENDATION 23
10.2 COLLABORATIVE-FILTERING RECOMMENDATION 25
10.3 POPULARITY-BASED RECOMMENDATION 28
11. HARDWARE & SOFTWARE REQUIREMENTS 29
12. CODING 30
13. SCREENSHOT 35
14. CONCLUSION 38
15. FUTURE ENHANCEMENT 39
16. REFERENCES 40

1. Abstract

In the modern world and the upcoming generation, the internet has become an integral part of human life. People frequently encounter decision-making challenges, whether it's finding a hotel or exploring academic research options. There is an abundance of information available, which can be overwhelming. To help users navigate this information overload, businesses have turned to recommendation systems to assist their customers.

The field of recommendation systems has faced persistent challenges, primarily due to the proliferation of valuable and effective applications and ongoing issues. Various e-recommendation approaches have been developed and applied, such as the recommendation systems for products on Flipkart or shows on Amazon Prime. These systems have greatly benefited e-commerce websites like Flipkart.com, adding to the richness of user experiences.

Recommender systems offer personalized recommendations, helping users discover shows and movies that align with their preferences. These systems can provide instant results or incorporate user feedback for further refinement. These user interactions can shape the recommendation algorithm and guide future suggestions for user-specific content.

The economic viability of recommendation systems has made them essential for major e-commerce platforms like Flipkart.com and Amazon.in. These platforms have successfully integrated recommendation systems, enhancing user experiences by delivering high-quality personalized recommendations. Customized recommendations add an extra layer of value to user interactions, making recommender systems a pivotal component of modern applications.

2. Literature Review

In the digital age, the entertainment industry grapples with an inundation of movie and TV show choices, necessitating the integration of movie recommendation systems. These systems have undergone a transformative journey, evolving from rudimentary content-based methods to sophisticated collaborative filtering approaches, such as Netflix's pioneering Cinematch. This literature review outlines the fundamental theories and concepts behind these systems, encompassing content-based and collaborative filtering, with an increasing focus on hybrid models. Recent research developments have seen the rise of deep learning techniques, exemplified by neural collaborative filtering and deep neural networks, and have refined matrix factorization methods. The literature review sets the stage for our report's exploration of this field, emphasizing the dynamic landscape of movie recommendation systems and their promise in enhancing user experiences.

3. Problem Statement

For building a recommender system from scratch, we face several different problems. Currently there are a lot of recommender systems based on the user information, so what should we do if the website has not gotten enough users. After that, we will solve the representation of a movie, which is how a system can understand a movie. That is the precondition for comparing similarity between two movies. Movie features such as genre, actor and director is a way that can categorize movies. But for each feature of the movie, there should be different weight for them and each of them plays a different role for recommendation. So we get these questions

- How to recommend movies when there are no user information.
- What kind of movie features can be used for the recommender system.
- How to calculate the similarity between two movies.
- Is it possible to set weight for each feature.

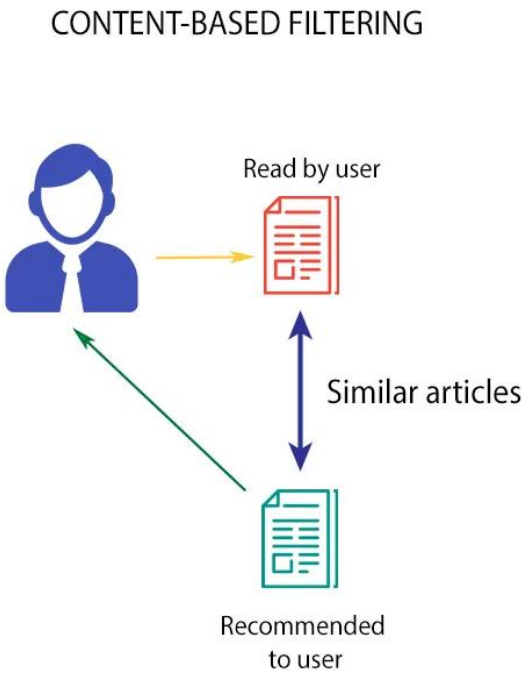
4. Objective

The goals of this thesis project is to do the research of Recommender Systems and find a suitable way to implement it, There are many kinds of Recommender Systems but not all of them are suitable for one specific problem and situation. Our goal is to find a new way to improve the classification of movies, which is the requirement of improving content-based recommender systems.

5. Methodology

5.1 Content-Based Filtering

- The content-based filtering aspect of the system utilizes natural language processing and information retrieval techniques to analyze the movie metadata.
- Feature extraction is performed to identify key attributes for each movie, enabling similarity measurements.
- Recommendations are generated based on user profiles, comparing user preferences with movie attributes like genre, director, and actors.

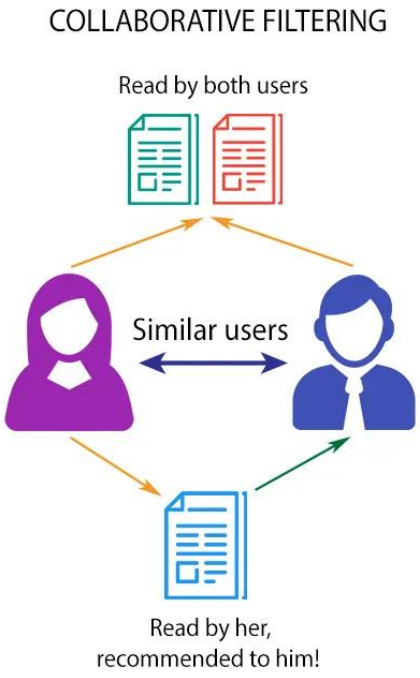


5.2 Popularity-Based Recommendations

- The popularity-based recommendations rely on real-time user engagement data to identify trending and popular movies.
- Movies are ranked based on various metrics, such as total views, user ratings, and recent user activity.
- Users are presented with a list of movies that are currently popular or trending, allowing them to explore content that aligns with broader user interests.

5.3 Collaborative Filtering:

- Collaborative filtering leverages user-user and item-item similarities in the dataset.
- User-based collaborative filtering identifies users with similar preferences and recommends movies that were highly rated by these like-minded users.
- Item-based collaborative filtering suggests movies that are similar to those a user has previously rated or interacted with, effectively offering personalized recommendations.



6. Libraries and Frameworks

6.1 Pandas

- Pandas is a powerful data manipulation library for Python.
- It provides data structures like DataFrames and Series, making it easy to work with structured data.
- Pandas offers tools for data cleaning, transformation, aggregation, and analysis.
- It is widely used for tasks like reading/writing data from/to various file formats, such as CSV, Excel, and SQL databases.
- Data indexing and selection using Pandas allow for efficient data exploration.
- With Pandas, you can handle missing data and perform operations like merging, grouping, and pivoting tables.

6.2 NumPy

- NumPy is a fundamental library for numerical and scientific computing in Python.
- It introduces the ndarray, a multi-dimensional array that provides efficient storage and operations on large datasets.
- NumPy is essential for mathematical operations, linear algebra, and random number generation.
- It is the foundation of many other libraries and frameworks used in data analysis and machine learning.
- NumPy provides performance optimizations that make it ideal for scientific computing tasks.
- Its capabilities include element-wise operations, broadcasting, and integration with C/C++ code.

6.3 Matplotlib:

- Matplotlib is a popular data visualization library in Python.
- It offers a wide range of chart types, including line plots, bar charts, scatter plots, and more.

- Matplotlib allows for fine-grained customization of plot appearance, such as labels, colors, and legends.
- It supports interactive plotting and can be used for creating publication-quality figures.
- Matplotlib can be integrated with various user interface toolkits for GUI applications.
- Matplotlib is widely used in scientific research, data analysis, and data presentation.

6.4 Scikit-Learn:

- Scikit-Learn is a machine learning library for Python, designed for simplicity and efficiency.
- It offers a wide range of machine learning algorithms for classification, regression, clustering, and more.
- Scikit-Learn provides tools for data preprocessing, feature selection, and model evaluation.
- It emphasizes consistent and easy-to-use APIs for machine learning tasks.
- Scikit-Learn is widely used for building, training, and deploying machine learning models.
- It is an essential tool in the data science and machine learning ecosystem.

6.5 NetworkX:

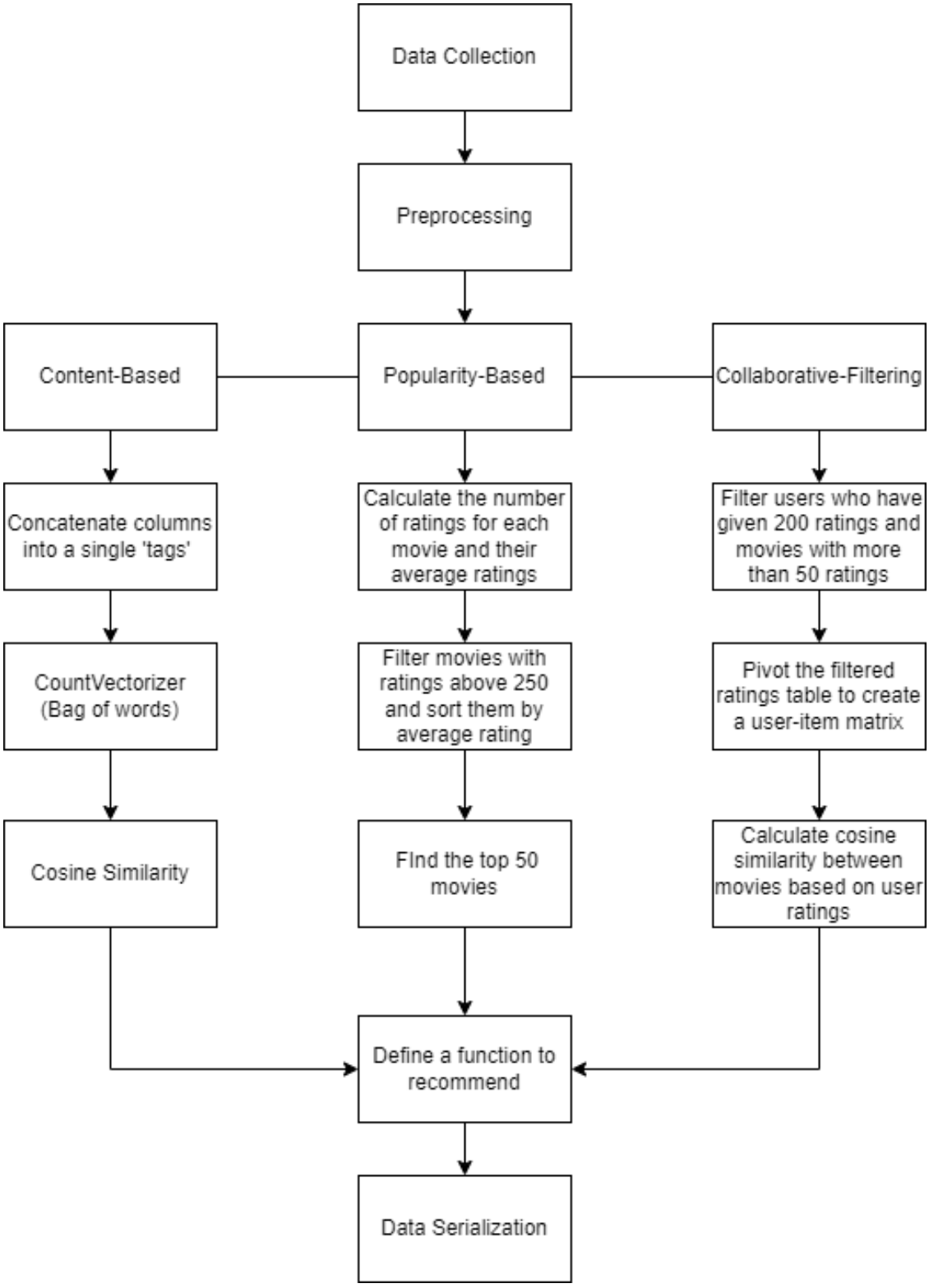
- NetworkX is a Python library for creating and analyzing complex networks and graphs.
- It provides data structures for representing graphs, nodes, and edges.
- NetworkX supports the study of network structures, algorithms, and properties.

6.6 NLTK (Natural Language Toolkit):

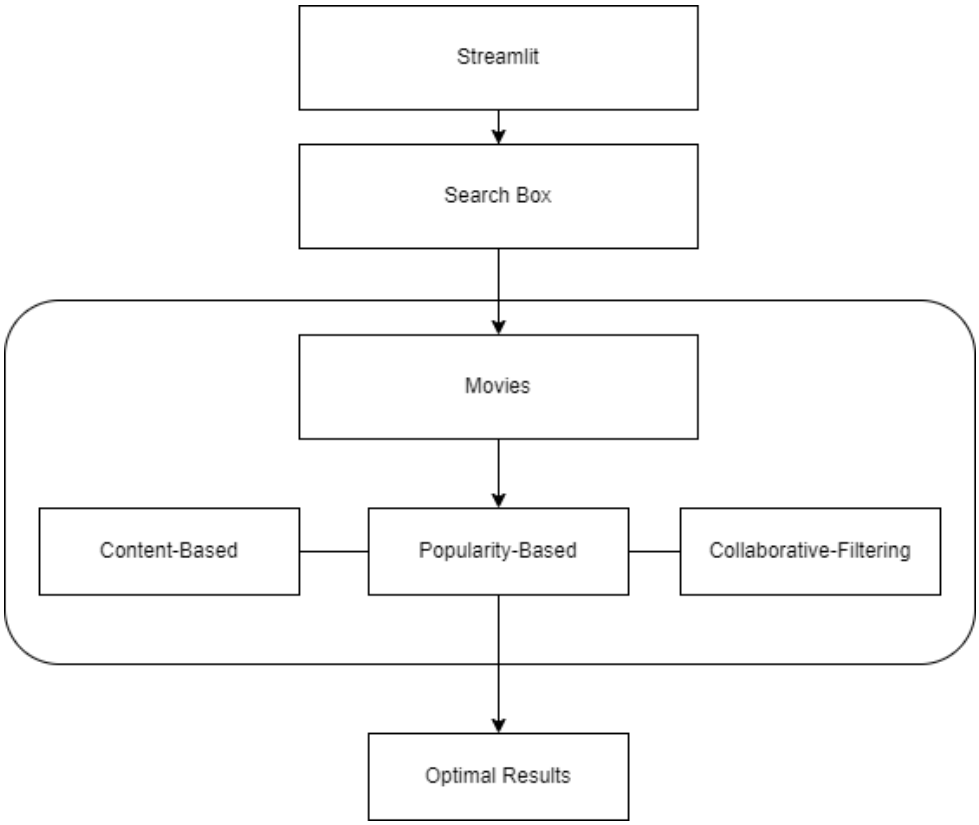
- NLTK is a comprehensive Python library for natural language processing (NLP).
- It provides tools, resources, and libraries for working with human language data.
- NLTK includes functionalities for tokenization, stemming, part-of-speech tagging, and named entity recognition.
- It offers access to various corpora and lexical resources for NLP research and development.

7.1 Architecture Diagram

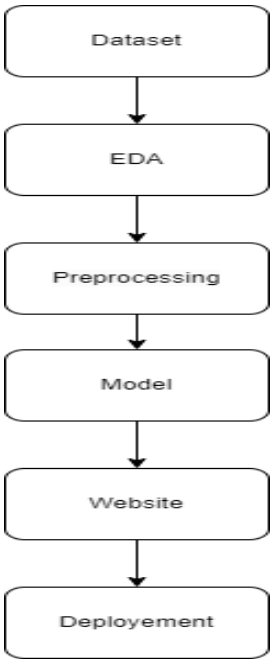
Backend/Notebook



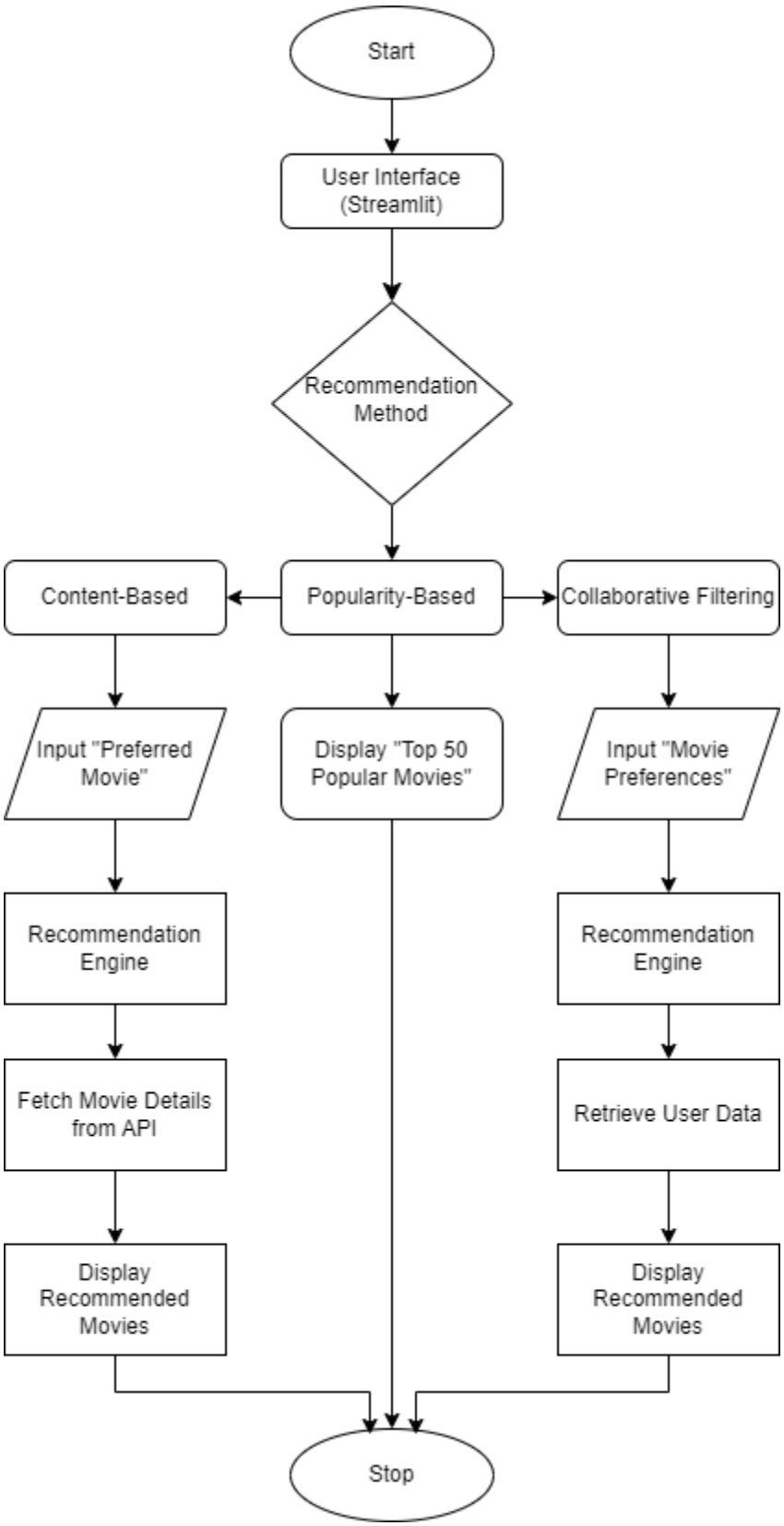
Frontend/Streamlit



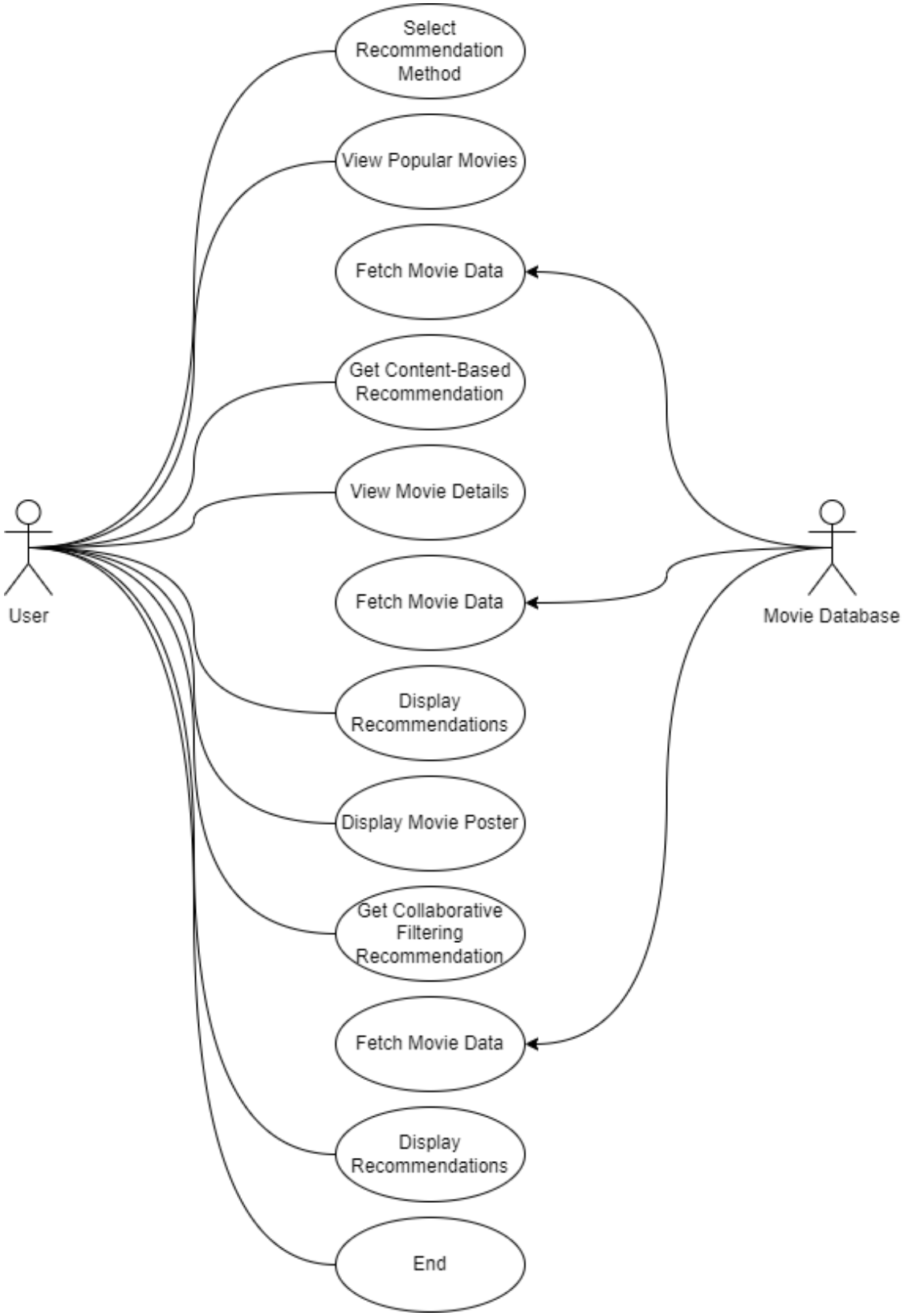
7.2 Block Diagram



7.3 Flow Chart



7.4 Use Case Diagram



8 Dataset

TMDb Movie Metadata Dataset:

- The TMDb Movie Metadata Dataset is a comprehensive and widely used dataset in the field of movie data analysis and recommendation systems.
- This dataset is hosted on Kaggle and provides valuable information about thousands of movies, including both popular and lesser-known titles.
- It contains a variety of data fields, such as movie titles, release dates, genres, budgets, revenues, and user ratings, making it suitable for a wide range of data analysis tasks.
- The dataset also includes cast and crew information, allowing for the exploration of movie personnel.
- It has been used for tasks like movie recommendation, genre analysis, box office predictions, and more.
- This dataset serves as a valuable resource for researchers, data scientists, and movie enthusiasts interested in analyzing and understanding the characteristics of movies.

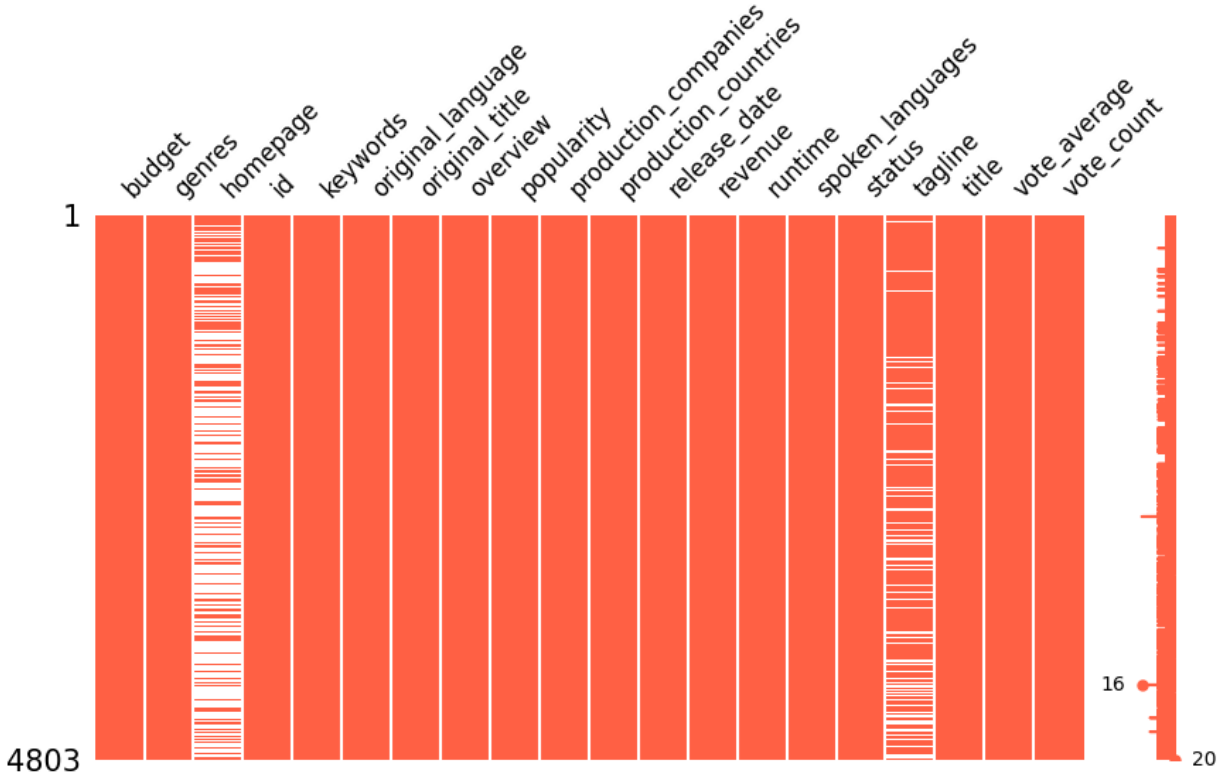
MovieLens 20M Dataset:

- The MovieLens 20M Dataset is a rich and extensive dataset for movie ratings and recommendations.
- It is provided by GroupLens Research and is one of the most widely used datasets for collaborative filtering and recommendation system development.
- This dataset contains 20 million movie ratings from users, covering a diverse set of movies and genres.
- Users provide ratings and reviews for movies, creating a valuable resource for user-based and item-based collaborative filtering approaches.
- The dataset also includes additional information such as movie metadata, tags, and user profiles, enhancing its usability for recommendation system development.
- Researchers and data scientists use this dataset to build, evaluate, and fine-tune movie recommendation algorithms.

9 Experiment

- **Data Cleaning**

Data cleaning involves the identification and handling of missing values, duplicated records, and inconsistencies in the dataset. In the movie datasets, missing values can exist in various columns, such as budgets, revenues, or cast and crew information. These missing values must be addressed through imputation or data removal.



Duplicated records, if any, are also identified and removed to ensure data integrity.

- **Data Integration**

Movie datasets often consist of multiple files or tables with related information. To perform comprehensive analyses, these tables may need to be integrated or merged using common identifiers, such as movie titles or IDs.

For example, the TMDb Movie Metadata dataset involves merging movie information with cast and crew details to create a more comprehensive dataset for analysis.

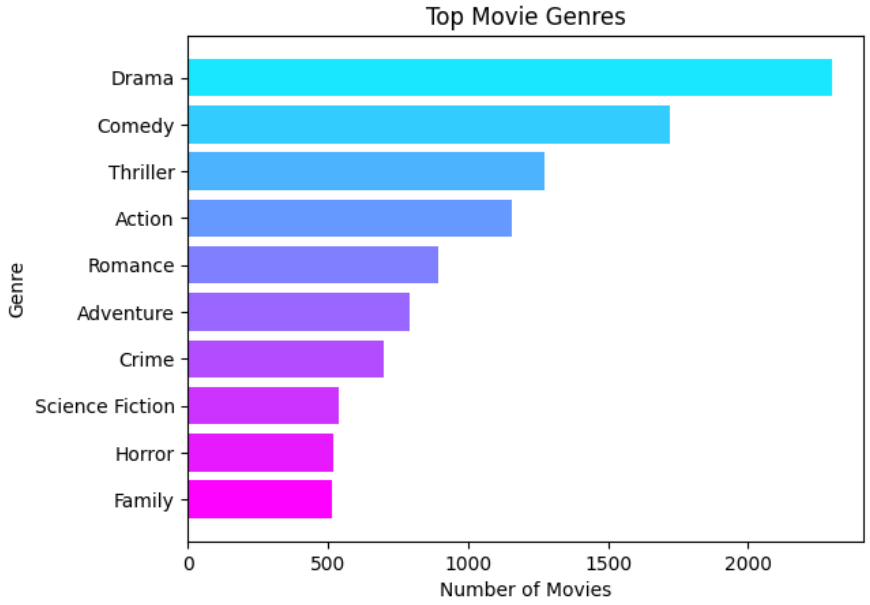
- **Data Transformation**

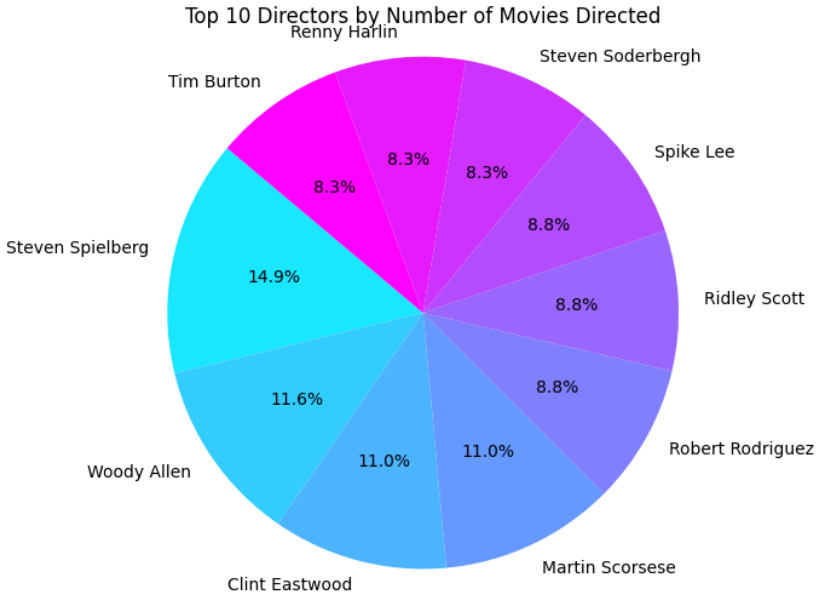
Data transformation encompasses changing the format or structure of the data to better suit the analysis or modeling needs. In movie datasets, this might involve:

Converting categorical data into a more usable format, such as one-hot encoding genres or keywords.

Extracting relevant information from text data, such as movie overviews, cast, and crew details.

For instance, in the TMDb dataset, extracting the top genres and directors.





Creating new features or aggregating existing ones, such as calculating the average rating for each movie or combining different textual features into a unified "tags" field.

- **Text Preprocessing (NLP)**

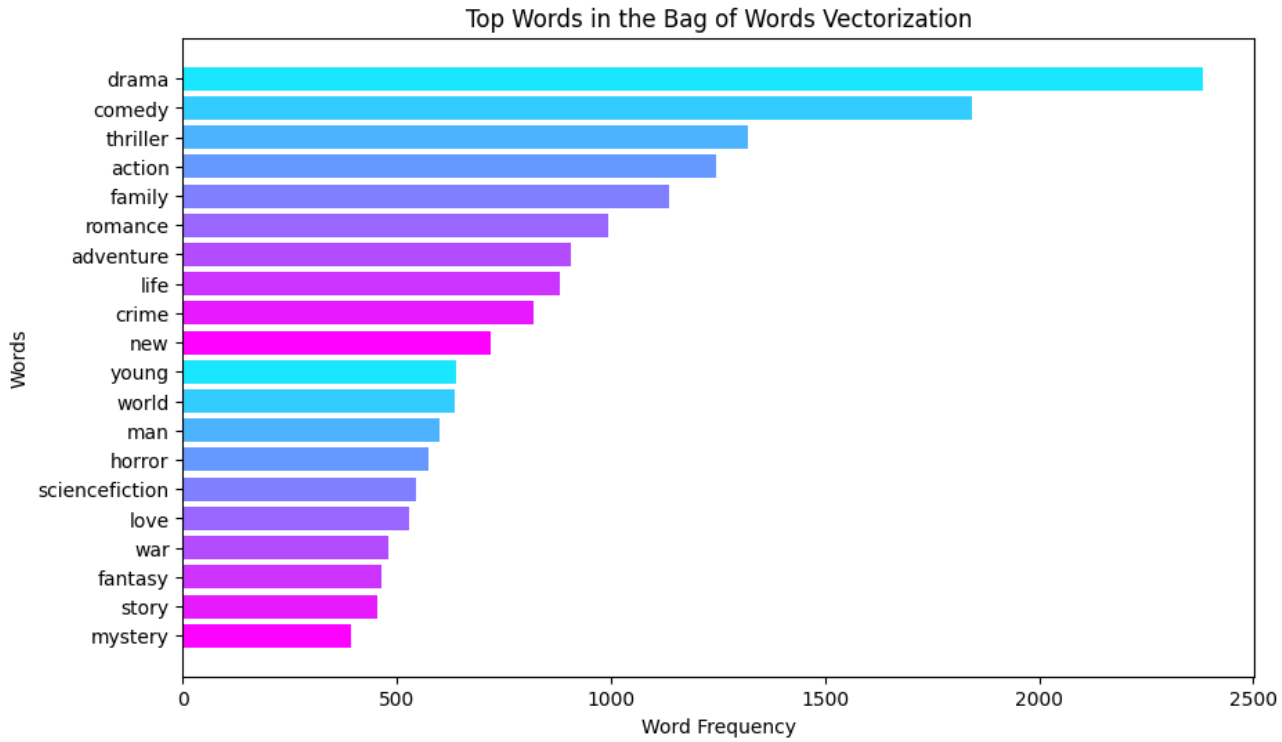
When dealing with textual data, natural language processing (NLP) techniques are often applied to preprocess and clean text. This includes:

Tokenization: Splitting text into individual words or tokens.

Removing stop words: Eliminating common words (e.g., "the," "and") that do not carry significant meaning.

Stemming or lemmatization: Reducing words to their base or root forms to standardize text.

Removing special characters, punctuation, and numbers.



Text preprocessing is essential for generating word clouds, performing text-based analysis, and building content-based recommendation systems.

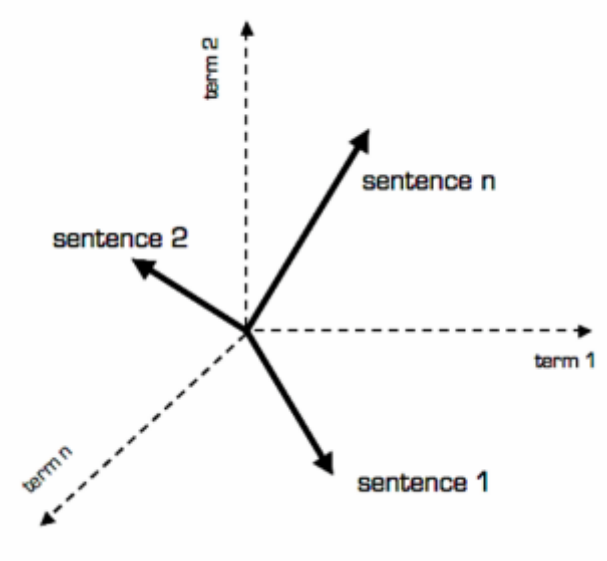
- **Feature Engineering**

Feature engineering involves creating new features from existing ones to enhance the dataset's informativeness and predictive power. In movie datasets, this may include generating features like tags, which consolidate information from multiple columns (e.g., overview, genres, keywords, cast, crew).

These engineered features can be used for content-based recommendation systems, text analysis, and data visualization.

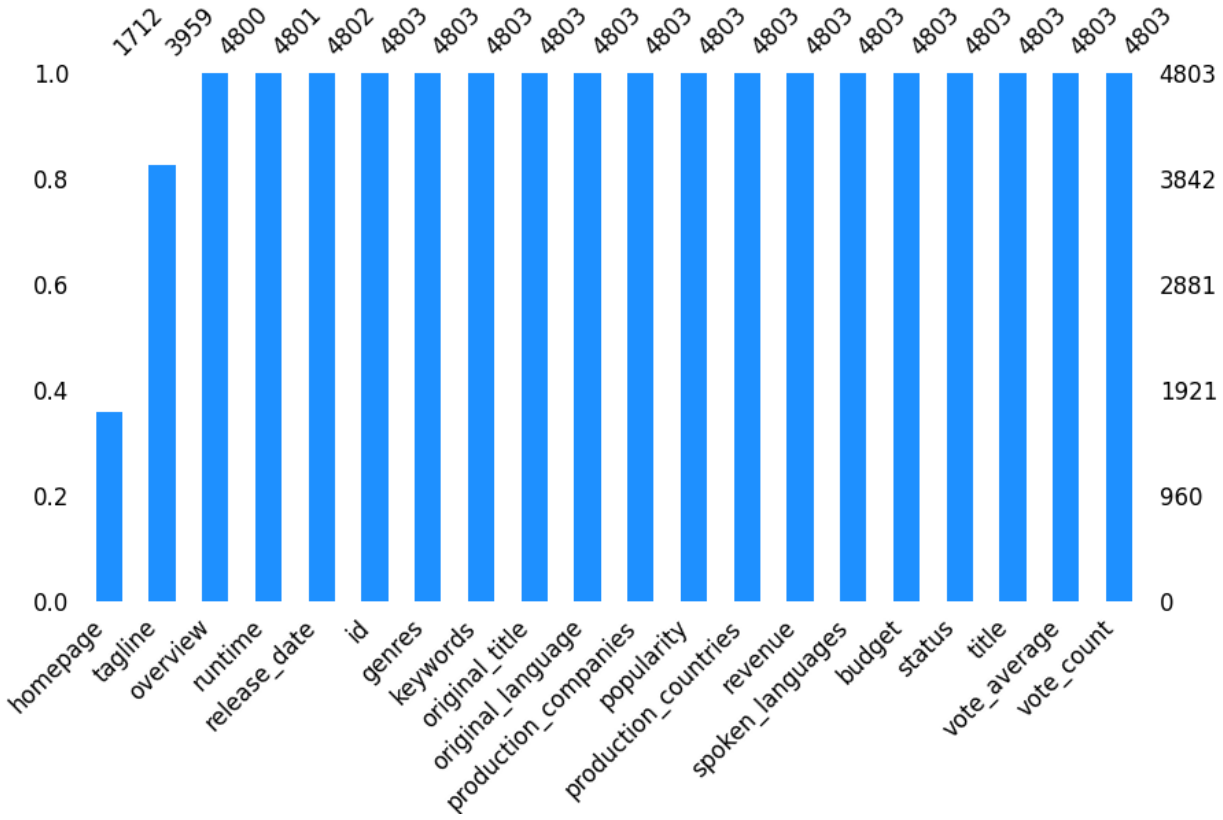
- **Vectorization**

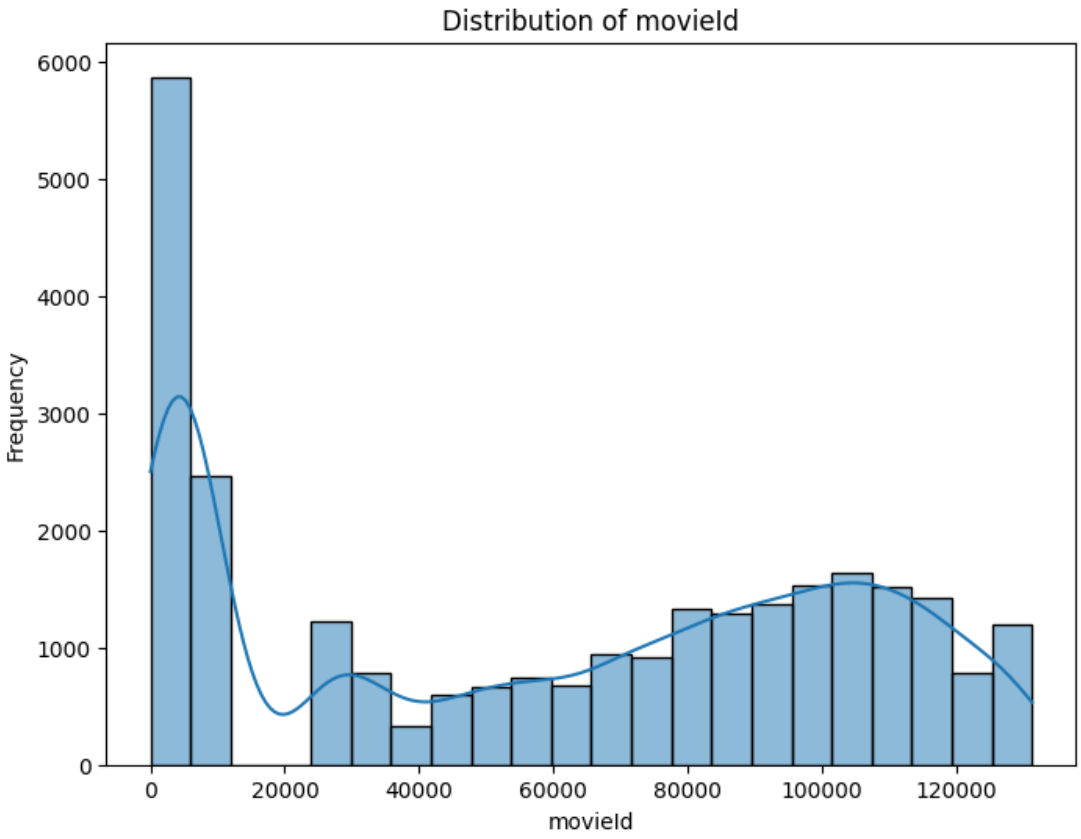
In content-based recommendation systems and text analysis, text data is often converted into numerical vectors. Techniques like Bag of Words (BoW) or Term Frequency-Inverse Document Frequency (TF-IDF) are employed for text vectorization. This allows for numerical analysis and similarity calculations.



- **Data Exploration and Visualization**

After preprocessing, it is common to perform exploratory data analysis (EDA) to gain insights into the data. Visualization tools and libraries like Matplotlib and Seaborn are used to create informative plots and visualizations.





Data preprocessing is a critical phase in the data analysis pipeline, and the quality of preprocessing directly impacts the accuracy and reliability of any subsequent analyses or modeling. The goal is to ensure that the dataset is ready for tasks such as building recommendation systems, movie trend analysis, or predictive modeling.

10 Implementation

10.1 Content-based Recommendation

Content-based recommendation is an important approach in recommender systems. The basic idea is to recommend items that are similar with what user liked before. The core mission of content-based recommender system is to calculate the similarity between items. There are a lot of methods to model item and the most famous one is Vector Space Model. The model extracts keywords of the item and calculate the weight by TF-IDF. For example, set ki as the ith keyword of item dj , wij is the weight of ki for dj , then the content of dj can be defined as:

$$Content(d_j) = \{w_{1j}, w_{2j}, ...\}$$

As we talked before, content-based recommender system recommends items that are similar with what user liked before. So the tastes of a user can be modeled according to the history of what the user liked. Consider ContentBasedP rof ile(u) as the preference vector of user u, the definition is:

$$ContentBasedProfile(u) = \frac{1}{|N(u)|} sum_{d \in N(u)} Content(d)$$

N(u) is what the user u liked before. After calculating content vector Content(.) and content preference vector ContentBasedP rof ile(.) of all users, given any user u and an item d, how the user like the item is defined as the similarity between ContentBasedP rof ile(u) and Content(d)

$$p(u, d) = sim(ContentBasedProfile(u), Content(d))$$

Using keywords to model item is an important step for many recommender systems. But extracting keywords of an item is also a difficult problem, especially in media field, because it is very hard to extract text keywords from a video. For solving this kind of problem, there are two main ways. One is letting experts tag the items and another one is letting users tag them. The representative of expert tagged systems are Pandora for music and Jinni for movies. Let's take Jinni as an example, the researchers of Jinni defined more than 900 tags as movie gene, and they let movie experts to make tags for them. These tags belong to different categories, including movie genre, plot, time, location and cast. Figure 10.1.1 below is from Jinni, which are the tags

for movie Kung Fu Panda. As we can see from the figure 10.1.1, the tags of Kung Fu Panda are divided into ten categories totally, Mood, Plot, Genres, Time, Place, Audience, Praise, Style, Attitudes and Look. These tags contain all aspects of movie information, which can describe a movie very accurately

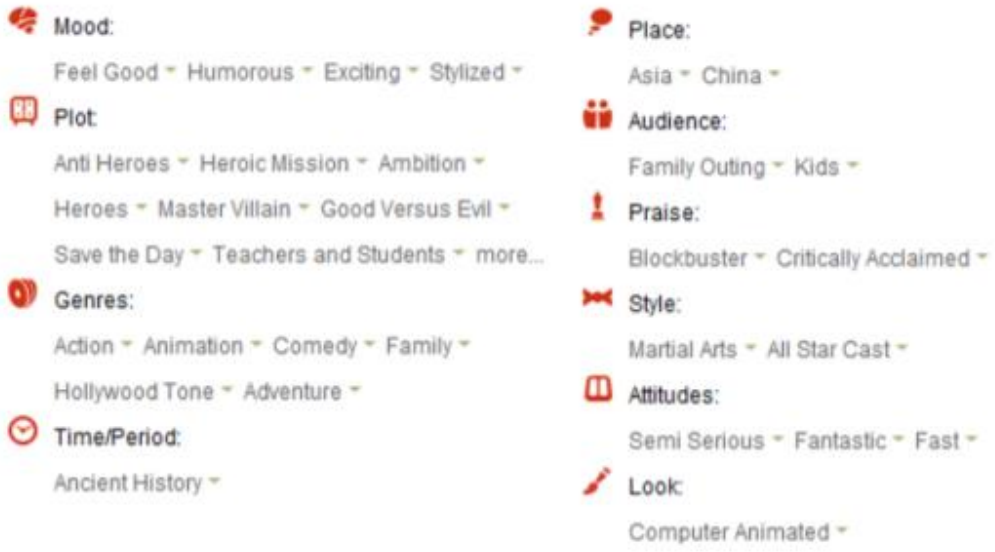
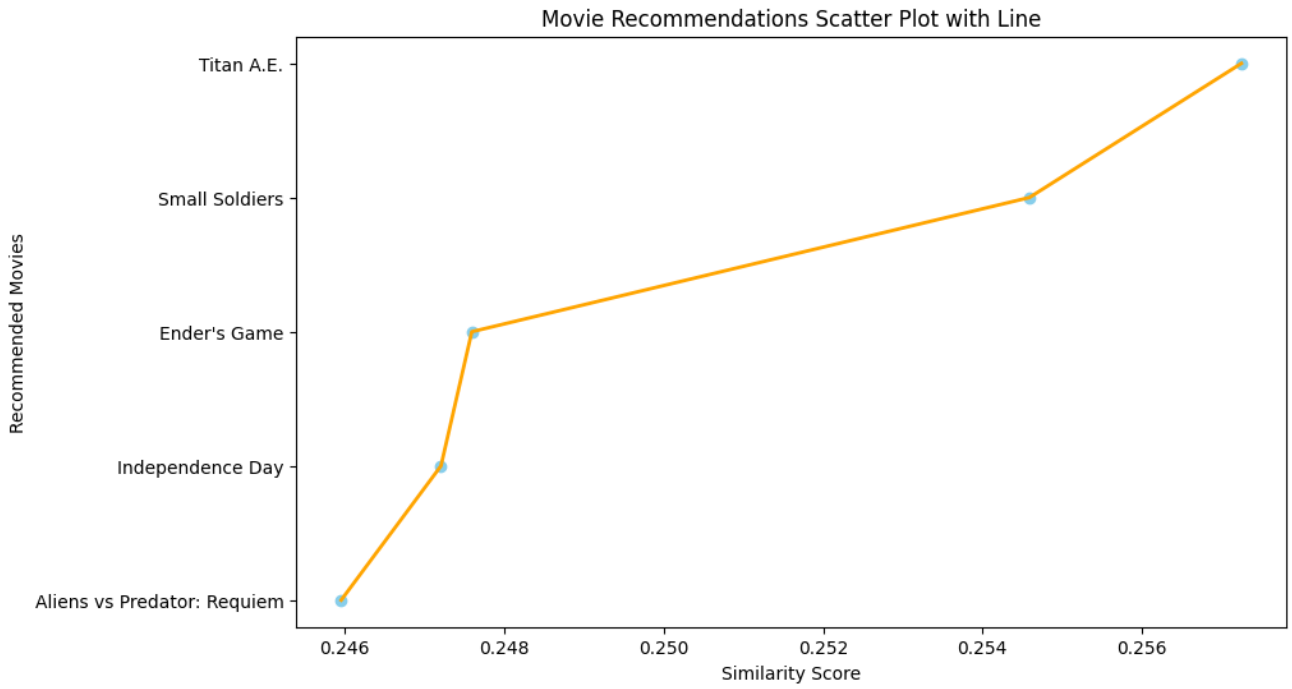


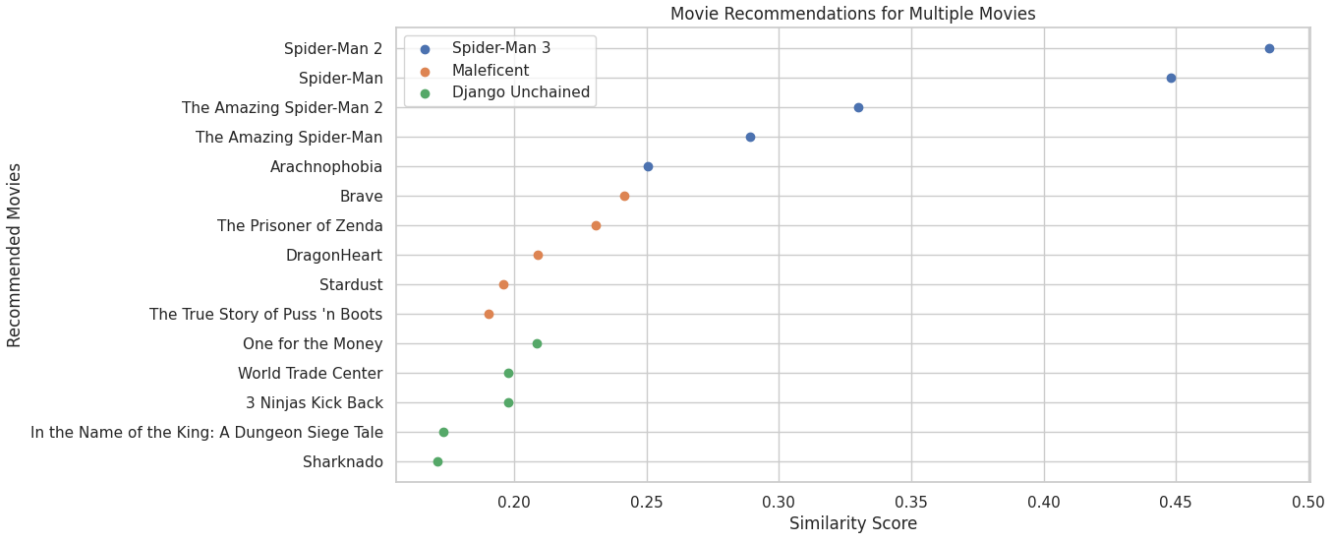
Figure 10.1.1

Compared with expert-tagged system, user-tagged system is applied more widely. The feature of user-tagged system is the tags are more diversity than that of expert-tagged system. But the weakness is that the tags are of lower quality, even there are a lot of wrong tags. So in the user-tagged systems, there are two main problems, one is tag recommendation, which means when a user tags an item, the system can recommend some relative

Results



Recommenden Movie – Avatar



Recommenden Movie - Spider-Man 3, Maleficent, Django Unchained

10.2 Collaborative-filtering Recommendation

Collaborative-filtering recommendation is the most famous algorithm in recommender systems. This algorithm models user's taste according to the history of user behavior. GroupLens published the first pape about collaborative filtering and the paper raised user-based collaborative filtering. In 2000, Amazon came up with item-based collaborative filtering in their paper. These two algorithms are very famous in business recommender systems.

User-based collaborative-filtering In user-based collaborative filtering, it is considered that a user will like the items that are liked by users with whom have similar taste. So the first step of user-based collaborative-filtering is to find users with similar taste. In collaborative filtering, the users are considered similar when they like similar items. Simply speaking, given user u and v, N(u) and N(v) are items set liked by u and v respectively. So the similarity of u and v can be simply defined as

$$s_{uv} = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

There are a lot of similarity algorithm, Equation 2.4 is one of them. User u's likeability for item i can be calculated by

$$p_{ui} = \sum_{v \in S(u,k) \cap N(i)} s_{uv} p_{vi}$$

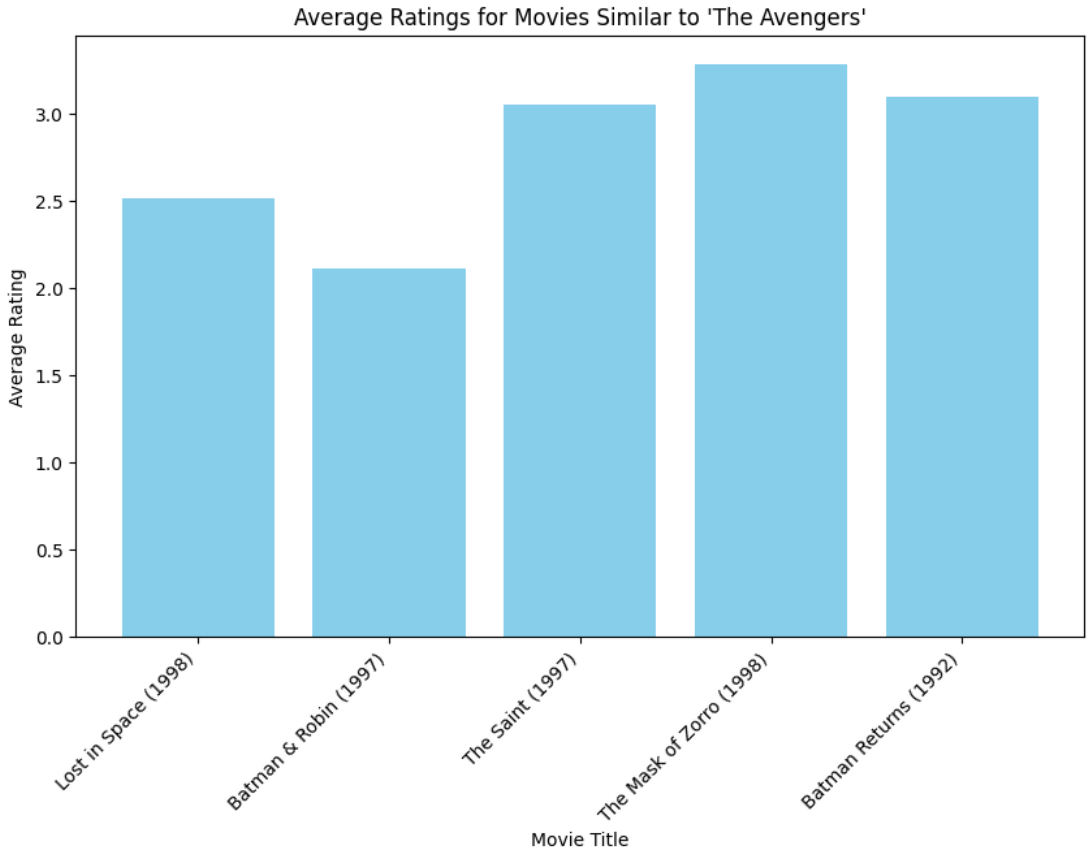
User/Item	Item A	Item B	Item C	Item D
User A	✓		✓	recommend
User B		✓		
User C	✓		✓	✓

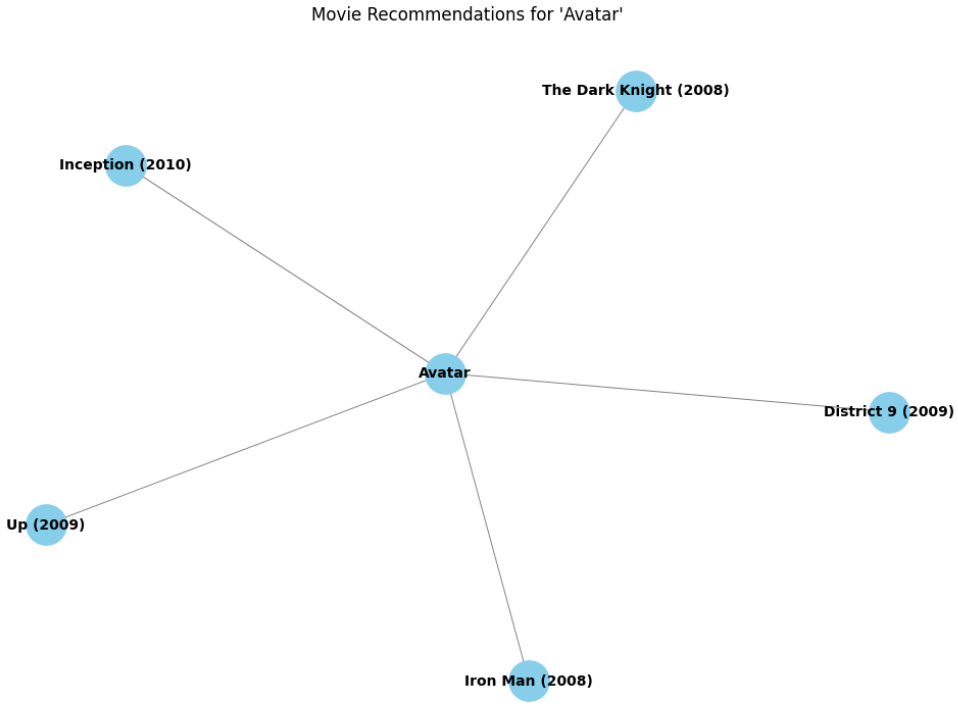
Figure

Figure is an example of User-based CF recommendation. According to the interest history of

User A, only User C can be the neighbor of him, so Item D will be recommended to User A.

Results


Average Ratings for Movies Similar to 'The Avengers'
Average Rating
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Lost in Space (1998)
Batman & Robin (1997)
The Saint (1997)
The Mask of Zorro (1998)
Batman Returns (1992)
Movie Title


Movie Recommendations for 'Avatar'
The Dark Knight (2008)
Inception (2010)
Avatar
District 9 (2009)
Up (2009)
Iron Man (2008)

10.3 Popularity-Based Recommendation

Popularity-based recommendation is one of the simplest and most straightforward recommendation algorithms. Instead of making personalized recommendations for each user, popularity-based systems suggest items based on their overall popularity within the entire user population. In other words, these systems recommend the most popular or highly-rated items to every user, regardless of their individual preferences. Popularity-based recommendation systems are particularly useful when there is a lack of user-specific data or for providing general recommendations to a large user base.

User-based collaborative filtering is a widely used algorithm in recommender systems that leverages user behavior and interactions to make personalized recommendations. The fundamental idea behind this approach is that users who have shown similar preferences in the past are likely to have similar preferences in the future. The algorithm identifies users with similar tastes based on their historical item interactions and recommends items that the similar users have liked. Here's how user-based collaborative filtering works

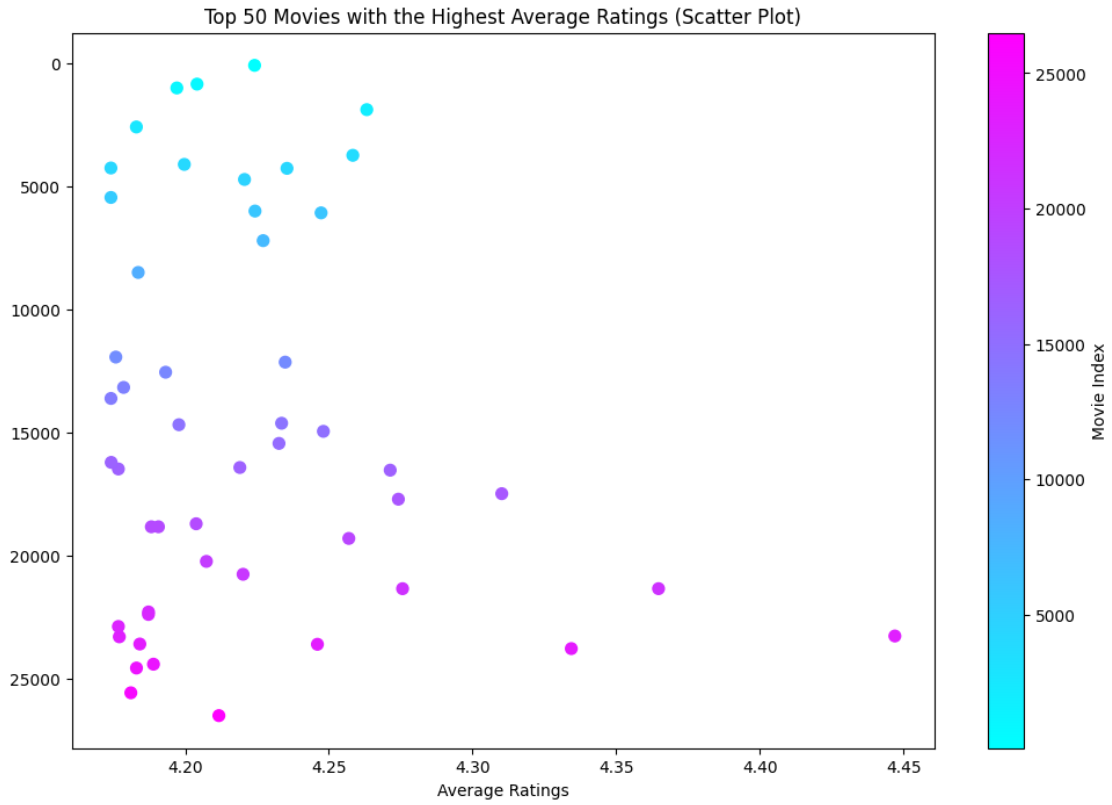
The first step in user-based collaborative filtering is to calculate the similarity between users. Various similarity metrics can be used, with Equation 2.4 being one of them. In this equation, "N(u)" represents the set of items liked by user "u," and "N(v)" represents the set of items liked by user "v."

Once user similarity scores are determined, the system predicts the user's likability for an item that they haven't interacted with based on the ratings of similar users.

User u's predicted rating for item i can be calculated using Equation 2.5. The equation considers the ratings of similar users who have interacted with item i and combines them to make a prediction for user u.

After predicting ratings for all items, the system generates recommendations for the user. The items with the highest predicted ratings are recommended to the user.

Results



11 Hardware & Software Requirements

Software Requirements

- Amazon SageMaker/ Jupyter Notebook
- Visual Studio
- Libraries & Frameworks - Pandas, NumPy, Matplotlib, Scikit-Learn, NLTK
- Windows 10
- Web Browser

Hardware Requirements

- RAM - 30GB
- SSD – 10GB

12 Coding

```
import streamlit_authenticator as stauth
import streamlit as st
import pandas as pd
import requests
import pickle
import time
from pathlib import Path
from streamlit_extras.badges import badge
from streamlit_option_menu import option_menu

from PIL import Image
im = Image.open('images/video.png')
st.set_page_config(page_title="Movie Recommender System App",page_icon = im)

import base64
def add_bg_from_local(image_file):
    with open(image_file, "rb") as image_file:
        encoded_string = base64.b64encode(image_file.read())
    st.markdown(
    f"""
    <style>
    .stApp {{
        background-image: url({encoded_string.decode()});
        background-size: cover
    }}
    </style>
    """,
    unsafe_allow_html=True
    )
add_bg_from_local('images/wallpaper_login.gif')

# --- USER AUTHENTICATION ---
names = ["Admin_Login"]
usernames = ["admin"]

# load hashed passwords
file_path = Path(__file__).parent / "pkl/hashed_pw.pkl"
with file_path.open("rb") as file:
    hashed_passwords = pickle.load(file)

authenticator = stauth.Authenticate(names, usernames, hashed_passwords,
    "movies_dashboard", "abcdef", cookie_expiry_days=30)

name, authentication_status, username = authenticator.login("Login", "main")
```

```
if authentication_status == False:
   st.error("Username/password is incorrect")

if authentication_status == None:
   st.warning("Please enter your username and password")

if authentication_status:

   selected = option_menu(None, ["Home", "Popularity", "Content", 'Collaborative'],
      icons=['house', 'fire', "camera-reels", 'people-fill'],
      menu_icon="cast", default_index=0, orientation="horizontal")

   if selected == "Home":
      st.title("Movie Recommendation System")
      st.write("Welcome to Movie Recommendation System! We help you discover movies
you'll love.")

      st.header("Popularity-Based Recommendations")
      st.write("Looking for trending or popular movies? Our popularity-based
recommendation system "
         "suggests movies that are currently trending or highly rated by a large number of
users. "
         "These recommendations are great for finding crowd-pleasers.")

      st.header("Content-Based Recommendations")
      st.write("Prefer movies similar to your favorite ones? Our content-based
recommendation system "
         "analyzes movie content (such as genre, director, actors) and suggests movies with
similar "
         "attributes to those you've enjoyed. This method is perfect for personalized
recommendations "
         "based on your unique taste.")

      st.header("Collaborative Filtering")
      st.write("Want recommendations based on user behavior? Our collaborative filtering
recommendation "
         "system leverages the preferences and behaviors of users similar to you to suggest
movies "
         "you might like. It's a powerful approach to discovering new favorites.")

      st.write("Get started by selecting one of the recommendation methods from the Menu
bar. Happy movie watching!")

      authenticator.logout('Logout','main')
```

```
        badge(type="github", name="shreyas-k-m/Movie-Recommendiation-System")

    if selected == "Popularity":
        @st.cache_data
        def load_movie_data():
            return pd.read_pickle('pkl/popular.pkl')
        movies_df = load_movie_data()

        st.title('Top 50 Popular Movies')
        st.write("Explore the top 50 popular movies with details on the number of ratings and
average ratings.")

        top_50_movies = movies_df[['title', 'num_ratings', 'avg_ratings']].head(50)

        column_names = {
            'title': 'Title',
            'num_ratings': 'No. Of Ratings',
            'avg_ratings': 'Average Ratings'
        }

        top_50_movies = top_50_movies.rename(columns=column_names)
        top_50_movies.index = range(1, 51)

        st.table(top_50_movies)
        st.line_chart(top_50_movies[['No. Of Ratings', 'Average Ratings']])

    if selected == "Content":
        st.title('Content Based Recommendations')
        st.write("Discover movies with similar attributes to those you like, such as genre actor
and director.")

        @st.cache_data
        def fetch_poster(movie_id):
            url =
"https://api.themoviedb.org/3/movie/{}?api_key=1aa68f108039b9d064664fb1e01c5e7a&la
nguage=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

        def recommend(movie):
            movie_index = movies[movies['title'] == movie].index[0]
            distances = similarity[movie_index]
```

```
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x:
x[1])[1:6]

        recommended_movies = []
        recommended_movie_posters = []

        for i in movies_list:
            # fetch the movie poster
            movie_id = movies.iloc[i[0]].movie_id
            recommended_movies.append(movies.iloc[i[0]].title)
            recommended_movie_posters.append(fetch_poster(movie_id))

        return recommended_movies,recommended_movie_posters

    movies_dict = pickle.load(open('pkl/movies_content.pkl','rb'))
    movies = pd.DataFrame(movies_dict)
    similarity = pickle.load(open('pkl/similarity_content.pkl','rb'))

    selected_movie_name = st.selectbox(
        "Type or select a movie from the dropdown",
        movies['title'].values
    )
    if st.button('Show Recommendation'):
        with st.spinner('Wait for it...'):
            time.sleep(1.5)
        st.success('Done!', icon="✅")
        names,posters = recommend(selected_movie_name)

        #display with the columns
        col1, col2, col3, col4, col5 = st.columns(5)
        with col1:
            st.markdown(names[0],)
            st.divider()
            st.image(posters[0],use_column_width='auto')
        with col2:
            st.markdown(names[1],)
            st.divider()
            st.image(posters[1],use_column_width='auto')
        with col3:
            st.markdown(names[2],)
            st.divider()
            st.image(posters[2],use_column_width='auto')
        with col4:
            st.markdown(names[3],)
            st.divider()
            st.image(posters[3],use_column_width='auto')
```

```
            with col5:
                st.markdown(names[4],)
                st.divider()
                st.image(posters[4],use_column_width='auto')

    if selected == "Collaborative":
        st.title("Collaborative Filtering Recommendations")
        st.write("Find movies similar to your selected one in terms of user ratings and
preferences.")

        pt = pickle.load(open('pkl/pt_collaborative.pkl', 'rb'))
        similarity_scores = pickle.load(open('pkl/similarity_collaborative.pkl', 'rb'))
        movies = pickle.load(open('pkl/movies_collaborative.pkl', 'rb'))

        available_movie_titles = pt.index.values
        selected_movie = st.selectbox("Type or select a movie from the dropdown",
available_movie_titles)

        def recommend(movie_name):
            matching_indices = [i for i, name in enumerate(pt.index) if movie_name in name]

            if matching_indices:
                index = matching_indices[0]
                similar_items = sorted(list(enumerate(similarity_scores[index])), key=lambda x:
x[1], reverse=True)[1:6]
                recommended_movies = [pt.index[i[0]] for i in similar_items]
                return recommended_movies
            else:
                return []

        if st.button("Show Recommendation"):
            with st.spinner('Wait for it...'):
                time.sleep(1.5)
            st.success('Done!', icon="✅")
            recommended_movies = recommend(selected_movie)

            if recommended_movies:
                st.subheader("Recommended Movies:")
                for i, movie in enumerate(recommended_movies):
                    st.write(f"{i + 1}. {movie}")
            else:
                st.warning("Movie not found in the dataset.")

    def add_bg_from_local(image_file):
        with open(image_file, "rb") as image_file:
            encoded_string = base64.b64encode(image_file.read())
```

13 Screenshot

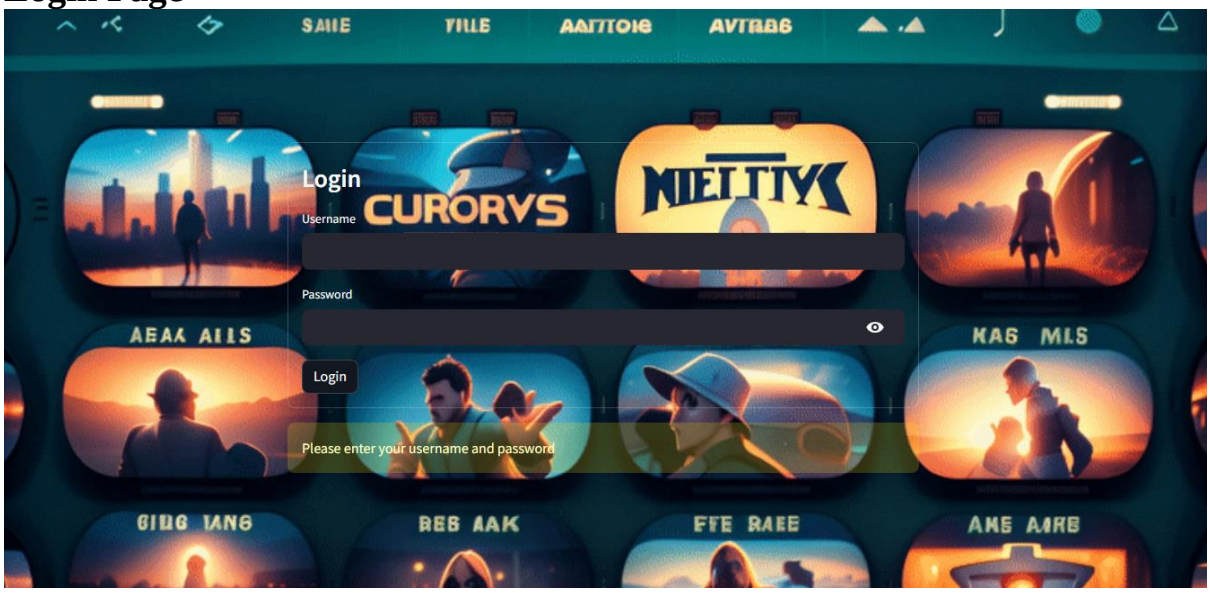
Notebook

```
#function to recommend the movies
def recommend(movie):

    #find the index of the movies
    movie_index = new_df[new_df['title']==movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:6]

    #to fetch movies from indeces
    for i in movies_list:
        print(new_df.iloc[i[0]].title)
```

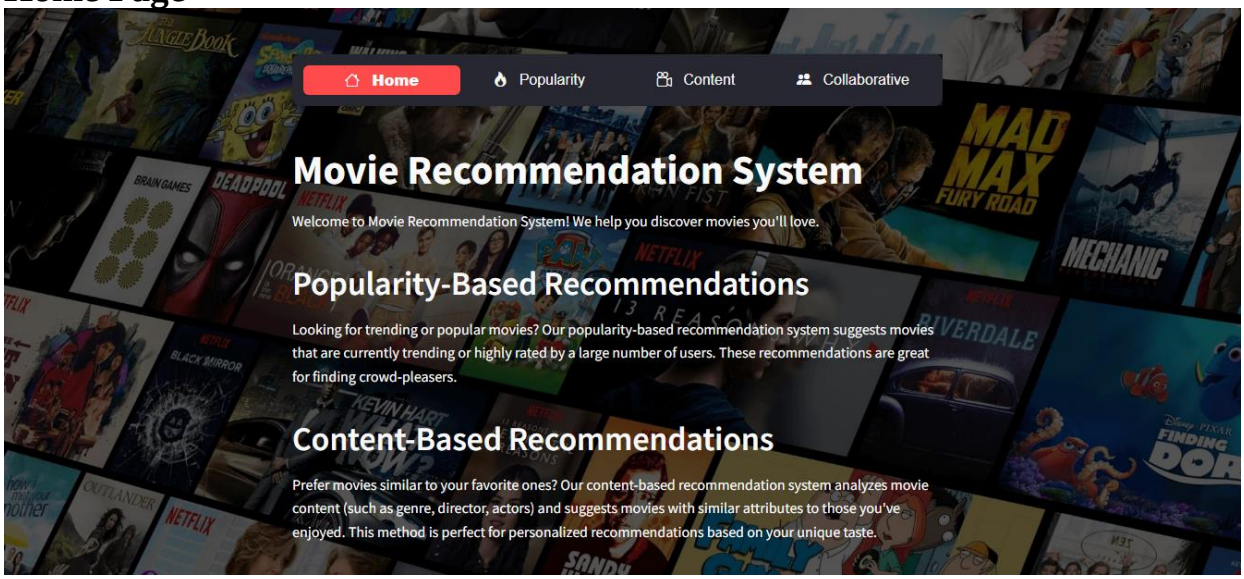
```
recommend('Batman Begins')
```

```
The Dark Knight
The Dark Knight Rises
Batman
Batman & Robin
Batman
```

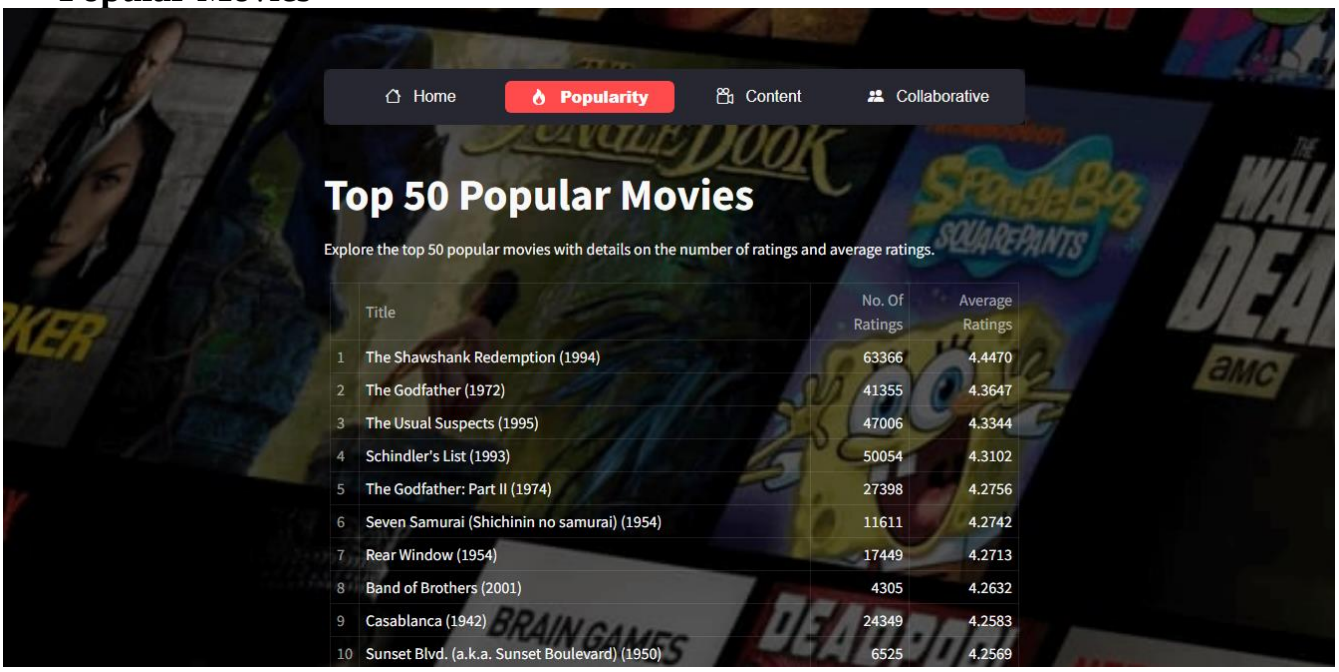
Login Page



Home Page

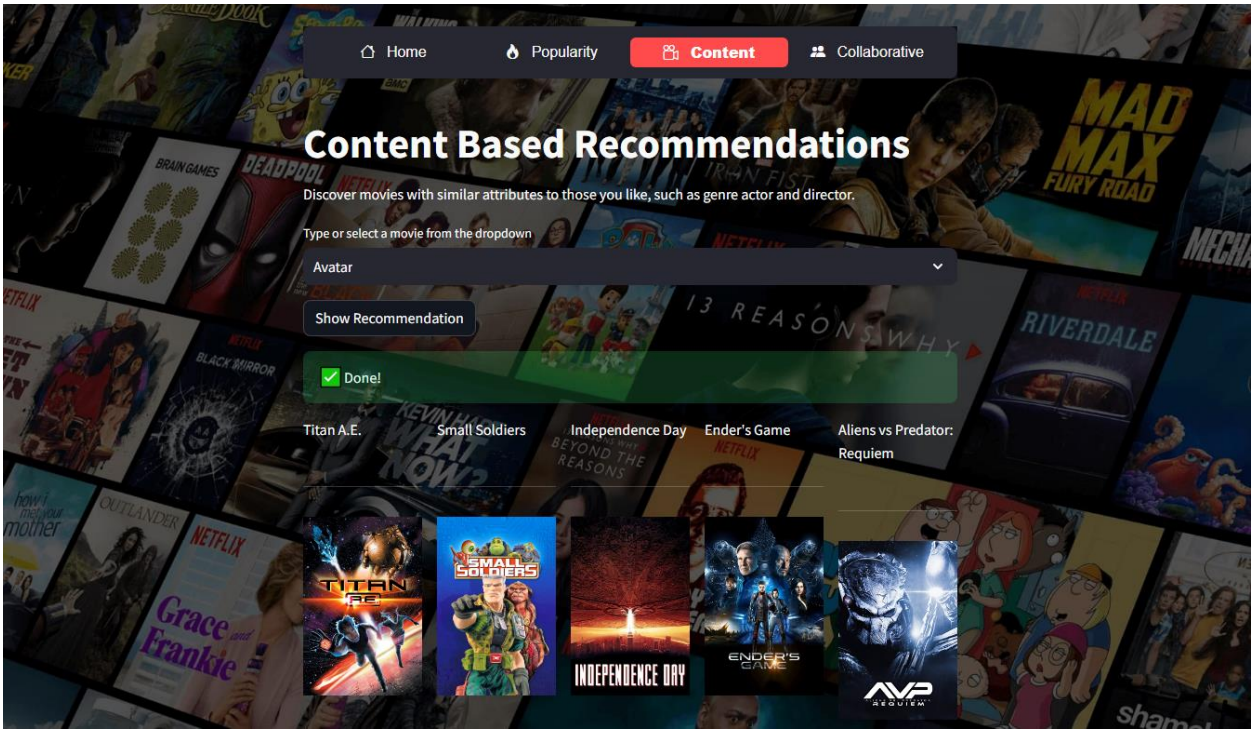


Popular Movies

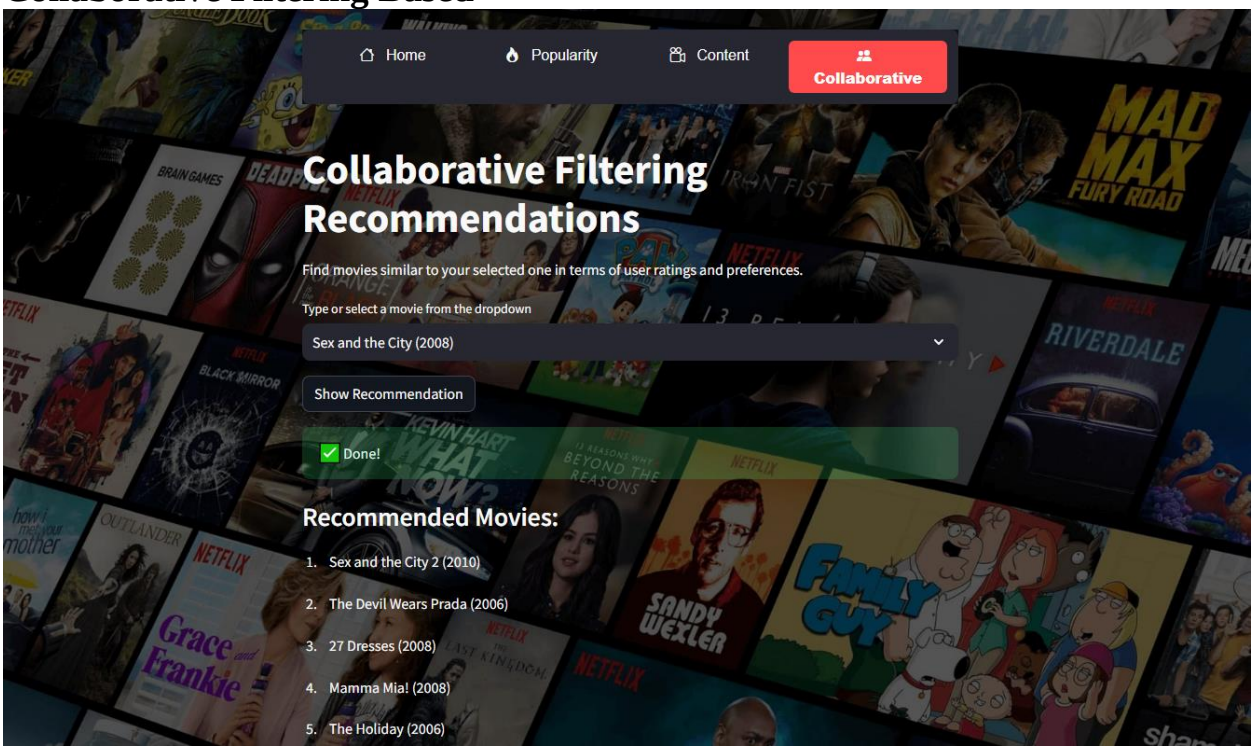


	Title	No. Of Ratings	Average Ratings
1	The Shawshank Redemption (1994)	63366	4.4470
2	The Godfather (1972)	41355	4.3647
3	The Usual Suspects (1995)	47006	4.3344
4	Schindler's List (1993)	50054	4.3102
5	The Godfather: Part II (1974)	27398	4.2756
6	Seven Samurai (Shichinin no samurai) (1954)	11611	4.2742
7	Rear Window (1954)	17449	4.2713
8	Band of Brothers (2001)	4305	4.2632
9	Casablanca (1942)	24349	4.2583
10	Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	6525	4.2569

Content Based



Collaborative Filtering Based



14 Conclusion

The movie recommendation system represents a powerful and versatile application of machine learning and data analysis in the entertainment industry. This system leverages user preferences and movie attributes to offer personalized movie suggestions, enhancing the overall viewing experience for users. Through the analysis of user behavior and content features, the recommendation system can efficiently match viewers with movies that align with their tastes and preferences.

The movie recommendation system employs a variety of recommendation algorithms, including collaborative filtering, content-based filtering, and hybrid models, to provide accurate and diverse movie recommendations. This approach accounts for both the user's historical interactions with movies and the inherent characteristics of the films themselves.

Moreover, the system relies on a robust technological infrastructure, encompassing hardware resources, software components, and data processing tools. High-performance servers or cloud resources, coupled with efficient data management systems and machine learning libraries, are critical for the successful operation of the system. The use of GPUs, containerization, and web frameworks further enhances its capabilities.

The deployment of movie recommendation systems has become prevalent in popular streaming platforms, enabling users to discover new and relevant content while boosting user engagement and retention. These systems have a profound impact on the entertainment industry, influencing user choices and fostering the discovery of hidden gems.

As technology continues to advance and data collection grows, movie recommendation systems will likely become even more precise and personalized, delivering an unparalleled viewing experience. The integration of emerging technologies such as natural language processing and deep learning will further enrich the recommendation process.

In essence, the movie recommendation system is an innovative solution that not only enhances user satisfaction but also drives business success in the competitive landscape of the entertainment industry. It demonstrates the potential of data-driven insights to provide users with tailored content recommendations, offering a glimpse into the future of personalized entertainment.

15 Future Enhancement

Recommender system has developed for many years, which ever entered a low point. In the past few years, the development of machine learning, large-scale network and high performance computing is promoting new development in this field. We will consider the following aspects in future work.

- Use collaborative filtering recommendation.
  After getting enough user data, collaborative filtering recommendation will be introduced.collaborative filtering is based on the social information of users, which will be analyzed in the future research.
- Introduce more precise and proper features of movie.
  Typical collaborative filtering recommendation use the rating instead of object features. In the future we should extract features such as color and subtitle from movie which can provide a more accurate description for movie.
- Introduce user dislike movie list.
  The user data is always useful in recommender systems. In the future we will collect more user data and add user dislike movie list. We will input dislike movie list into the recommender system as well and generate scores that will be added to previous result. By this way we can improve the result of recommender system.
- Introduce machine learning.
  For future study, dynamic parameters will be introduced into recommender system, we will use machine learning to adjust the weight of each feature automatically and find the most suitable weights.
- Make the recommender system as an internal service.
  In the future, the recommender system is no longer a external website that will be just for testing. We will make it as an internal APIs for developers to invoke. Some movie lists in the website will be sorted by recommendation.

16 References

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on, 17(6):734–749, 2005.

[2] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. Modern information retrieval, volume 463. ACM press New York, 1999.

[3] Shumeet Baluja, Rohan Seth, D Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In Proceedings of the 17th international conference on World Wide Web, pages 895–904.
ACM, 2008.

[4] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 95–104. ACM, 2007.

[5] Suvir Bhargav. Efficient features for movie recommendation systems. 2014.

[6] Robin Burke. Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4):331–370, 2002.

[7] Robin Burke. Hybrid web recommender systems. In The adaptive web, pages 377–408. Springer, 2007.

[8] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. The youtube video recommendation system. In Proceedings of the fourth ACM conference on Recommender systems, pages 293–296. ACM, 2010.

[9] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. JAsIs, 41(6):391–407, 1990.

[10] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with ap 39 BIBLIOGRAPHY