رواد مصر الرقمية

Personalized Recommendation System

Supervised By: AST Academy - 2024

Team Members

❖ Ahmed Salah Emam
❖ Amir Tarek Ali
❖ Aya AbdElnaser Muhammed
❖ Dina Saed Farag
❖ Shadan Mohamed Aref
❖ Youssef Ezzeldin Mahmoud

Table of contents

1. EXECUTIVE SUMMARY

The objective of this project was to develop a personalized recommendation system for movies, leveraging datasets from Netflix and Disney. The goal was to provide users with tailored movie suggestions based on content features such as genres, cast, and descriptions. Due to the absence of user interaction data, we adopted a content-based approach, utilizing key attributes of each movie to predict relevance to users' preferences. Additionally, collaborative filtering methods were explored to enhance the recommendation quality by using item similarity.

We employed machine learning models and advanced techniques, including matrix factorization and hybrid recommendation algorithms, to optimize the recommendation process. The system was integrated with Azure for scalable deployment and continuous model updates using MLOps principles.

Key outcomes included an efficient recommendation engine capable of delivering relevant movie suggestions, as well as a robust Azure infrastructure supporting seamless scaling and real-time recommendations. The final evaluation demonstrated the system's effectiveness in improving user satisfaction by delivering personalized content.

2. INTRODUCTION

Problem Statement

With the ever-growing libraries of content on platforms like Netflix and Disney, users face challenges in discovering movies that align with their preferences. The vast array of options makes it difficult for users to find relevant content quickly, leading to choice overload and potential dissatisfaction. Without tailored recommendations, users may either miss out on desirable movies or spend excessive time searching for something to watch. A recommendation system is essential to enhance user experience by delivering personalized suggestions that cater to individual tastes.

Goals and objectives of the recommendation system

- **Personalization**: The core objective is to provide personalized movie recommendations based on the user's preferences and past behavior.
- **Efficiency**: To reduce the time users spend searching for relevant movies by providing accurate suggestions.
- **Content Exploration**: Enable users to discover new and diverse content that matches their interests, while also promoting movies they might not have considered otherwise.
- **Scalability**: Develop a system that can efficiently scale with increasing user and content data, providing real-time recommendations.
- **Azure Integration**: Implement cloud-based deployment using Azure to ensure robust infrastructure for handling large datasets and seamless model updates.

Relevance in the context of Netflix and Disney datasets

The Netflix and Disney datasets contain a wealth of movie-related information, including genres, cast, directors, and descriptions, which are critical for building a content-based recommendation system. Although user interaction data (such as ratings) is unavailable in these datasets, the rich metadata allows us to explore various recommendation techniques. By leveraging these datasets, we aim to create a system that understands content similarities and delivers accurate recommendations even in the absence of explicit user preferences. This approach is particularly relevant for enhancing the user experience on platforms with extensive content libraries like Netflix and Disney.

3. DATA COLLECTION AND PREPROCESSING

Datasets used and why

For this project, we utilized two datasets: Netflix and Disney. These datasets were chosen for their extensive movie metadata, which includes features like title, genre, director, cast, country, rating, and release year. The Netflix dataset provides detailed information on movies and TV shows available on the platform, while the Disney dataset offers a similar scope of metadata for Disney's content.

Both datasets are rich in content-related information, making them ideal for a **content-based recommendation system**, especially in the absence of user interaction data (e.g., ratings or clicks). By merging these datasets, we were able to expand our content library, which is crucial for improving recommendation diversity and accuracy.

Data preprocessing techniques

Before building our recommendation models, we undertook several preprocessing steps to ensure the data was clean and ready for use. The following key steps were implemented:

1. **Concatenation of Datasets**:

   The Netflix and Disney datasets were concatenated into a single dataset to allow the system to recommend content from both platforms. Since both datasets had similar columns, this was a straightforward process.

2. **Handling Missing Values:**

   Missing values, especially in crucial columns like director, cast, and country, were replaced with the placeholder 'No Data'. This approach allowed us to retain the rows without introducing bias from imputation methods like filling with the mean or mode.

3. **Cleaning Text Data:**

   The duration column contained strings such as '120 min'. We stripped the text and converted it into integers for easier manipulation and analysis.

4. **Categorical Data Encoding:**

   Some features such as rating were transformed into categorical variables, making it easier for models to understand.

5. **Feature Engineering:**

   A new feature year_added was created from the date_added column to allow analysis of when content was added to the platforms.

Key challenges in merging and cleaning the data

1. **Inconsistent Feature Formats:**

   The Netflix and Disney datasets, although similar, had slight variations in feature formats. For example, date fields were formatted differently, and the duration column had various formats that needed to be standardized across both datasets.

2. **Missing Data:**

Several rows were missing critical metadata (e.g., director, cast, country), which made it challenging to apply advanced recommendation techniques like collaborative filtering. To address this, we focused on content-based recommendations, leveraging available metadata.

3. **Duplicate Content:**

Some content appeared in both Netflix and Disney datasets. We handled duplicates by identifying identical content based on titles, release year, and other attributes, retaining unique entries.

Final data prepared for modeling

After preprocessing, the final dataset used for modeling contained the following cleaned and feature-engineered columns:

- Title: Movie or TV Show title.
- Type: Whether the content is a Movie or a TV Show.
- Director: Name of the director (with missing values filled as 'No Data').
- Cast: Actors/actresses in the movie/show (with missing values filled as 'No Data').
- Country: Country of origin (with missing values filled as 'No Data').
- Rating: Age-based content rating.
- Duration: Length of the movie or TV show in minutes.
- Release Year: Year the content was originally released.
- Year Added: The year the content was added to the respective platform (Netflix or Disney).

This dataset was then used to build the recommendation models in the following phase.

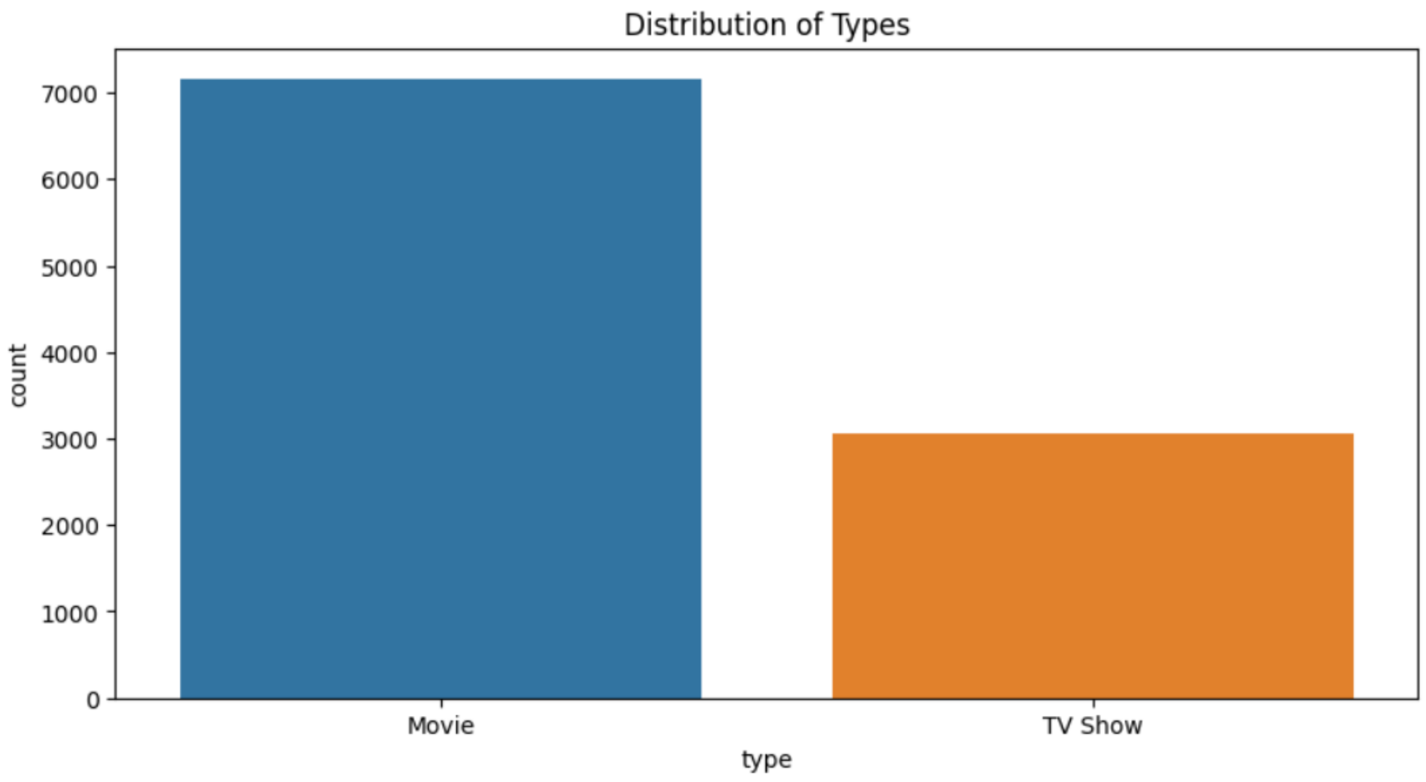
4. EXPLORATORY DATA ANALYSIS (EDA)

In this phase, we performed an in-depth analysis of the combined Netflix and Disney datasets to gain insights into their characteristics, distribution of content types, and other important variables. Here's a breakdown of the key aspects explored in this section:

- **Descriptive statistics**

  We first examined the overall structure of the dataset to understand the distribution of content types and other important variables.

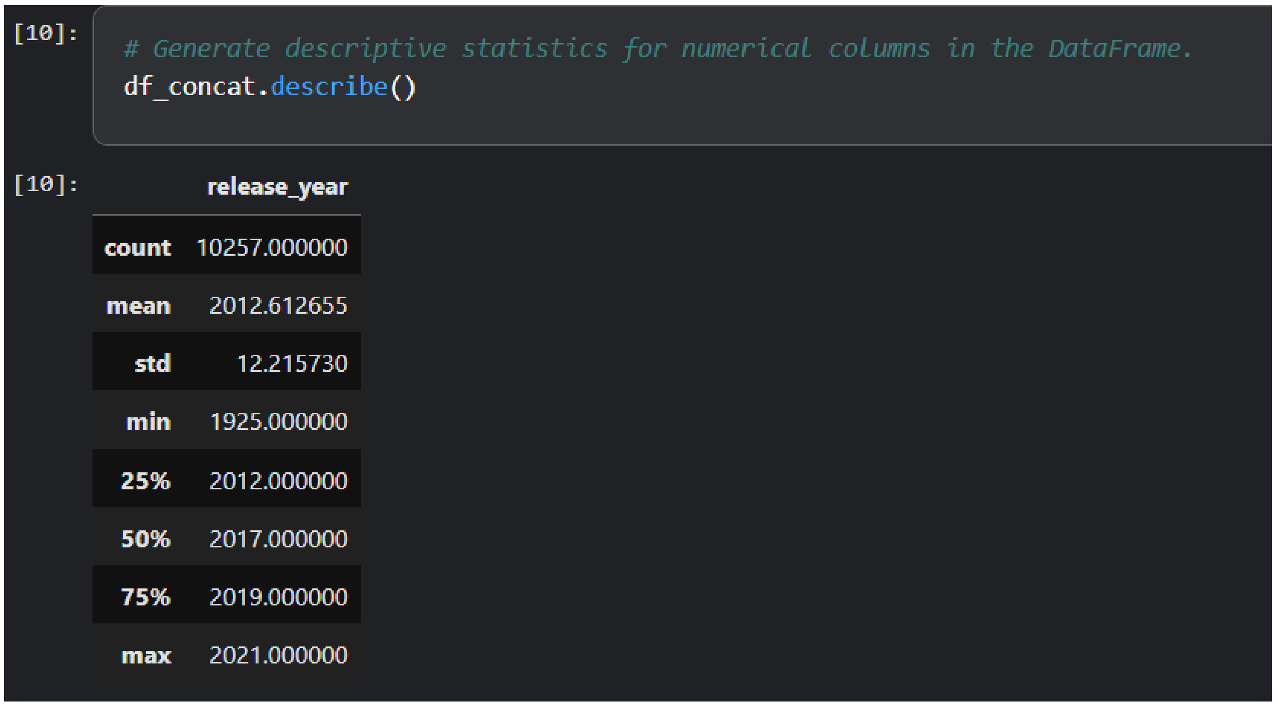
- **Count of Movies and TV Shows**: We computed the total number of Movies and TV Shows to understand the balance between the two content types:



Pic (1) Distribution of Types

This indicates that the combined dataset is heavily skewed towards Movies, which could potentially affect recommendations based on content type.

- **Descriptive Summary of Numeric Features**: We also calculated summary statistics (e.g., mean, median, standard deviation) for numerical columns like **release_year** and **duration**.



```
[10]: # Generate descriptive statistics for numerical columns in the DataFrame.
      df_concat.describe()
```

[10]:

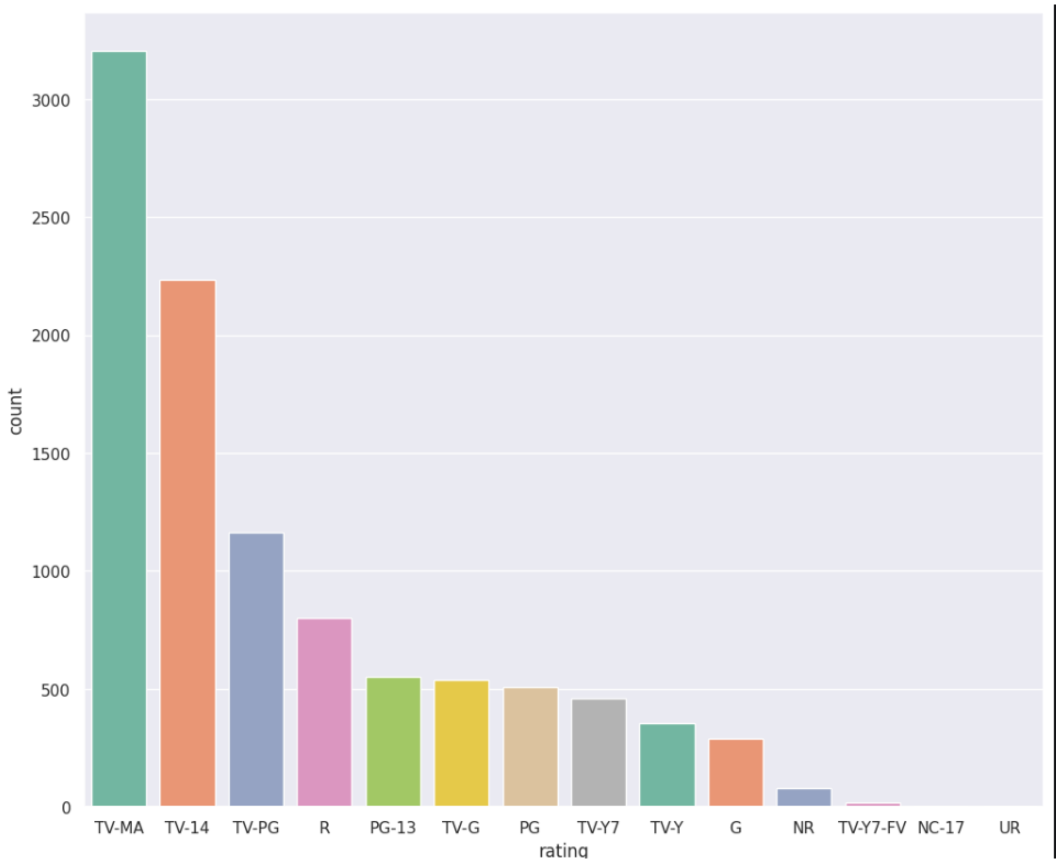
	release_year
count	10257.000000
mean	2012.612655
std	12.215730
min	1925.000000
25%	2012.000000
50%	2017.000000
75%	2019.000000
max	2021.000000

Pic (2) Descriptive Summary of Numeric Features

Distribution of key variables

We analyzed the distribution of several key features, including rating, genres, release year, and duration to understand the structure of the data more deeply. Visualizations were employed to make these distributions more comprehensible.

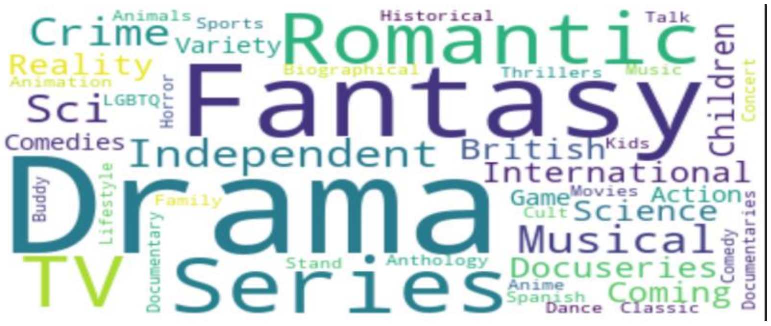
- **Ratings Distribution**: The rating system is a crucial feature in any content platform. We explored how content is distributed across different rating categories.



Pic (3). Rating Distribution

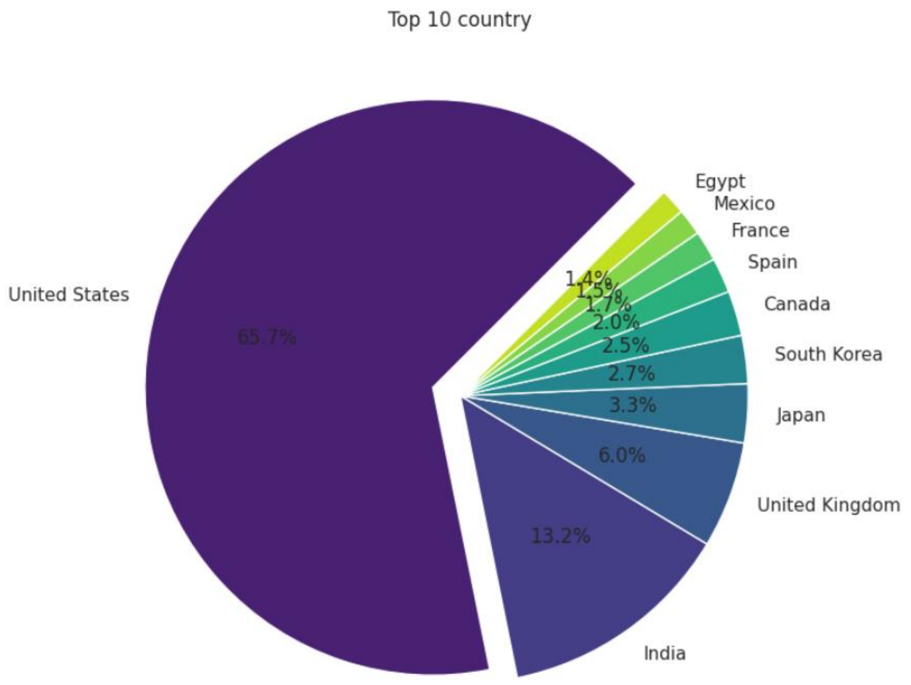
This allowed us to see which ratings are most common for both movies and TV shows. We found that certain ratings are more prevalent, which could influence the recommendation system.

- **Genre Distribution:** Movies and TV shows span a wide range of genres, and understanding the distribution of genres in our dataset is essential for content-based recommendations.



Pic (4) Genre Distribution

- **Country-Based Content Distribution:** A significant portion of the content came from the USA, followed by India, the UK, and Canada. This was visualized using the following plot:



Pic (5) Country-Based Content Distribution

Insights:

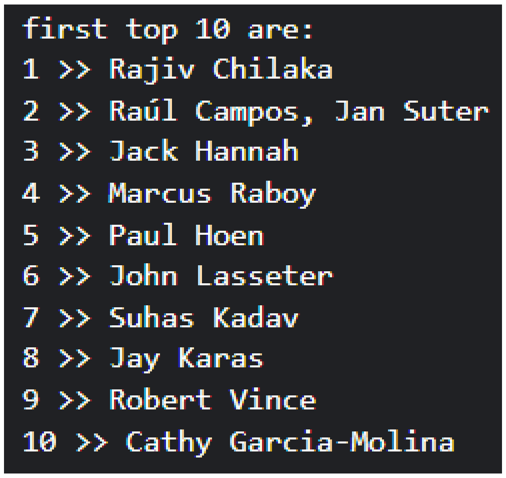
- The USA has the most diverse and consistent content release over the years.
- Countries like India and the UK contribute significant volumes, especially in specific genres such as drama and family-oriented content.

Insights discovered during EDA

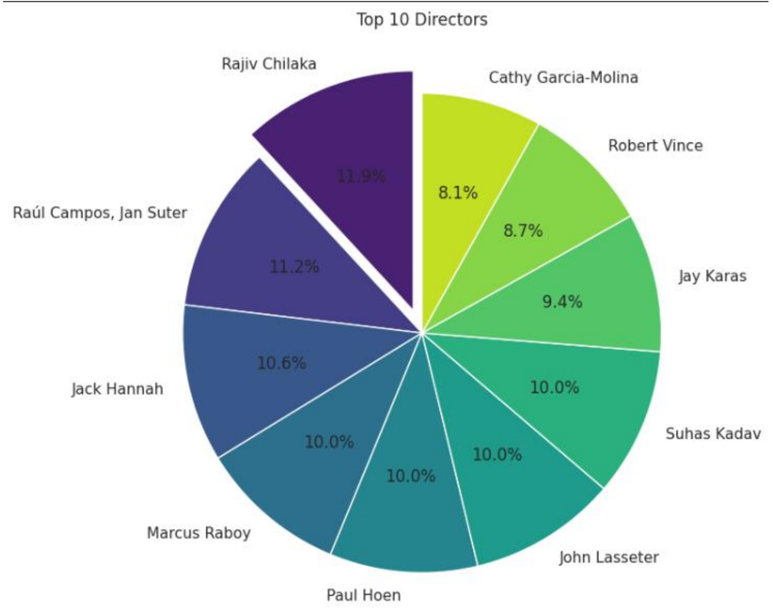
1. **Dominance of Recent Content:** A significant portion of the content was released in recent years, **w**ith a noticeable uptick after 2015. This suggests that both Netflix and Disney have been aggressively adding new content to their platforms in response to increasing competition.
2. **Rating Distribution:** The most common ratings for movies are PG-13 and TV-MA, which reflects the target audience for much of the content on these platforms. TV shows tend to have more diverse ratings, ranging from TV-G (general audience) to TV-MA (mature audience), indicating a wide variety of content catering to all age groups.
3. **Popular Genres:** The most common genres in the dataset include Drama, Comedy, and Documentary. These genres are popular across both movies and TV shows, which might suggest these are the genres users frequently engage with.
4. **Movies vs. TV Shows:** Movies outnumber TV shows in the combined dataset. This trend may reflect the higher volume of movie content produced historically and the increasing focus on TV series only in recent years with the rise of streaming services.

5. **Country-Wise Analysis:** Content originates from a variety of countries, with the United States leading in both movies and TV shows. Other significant contributors include India, the UK, and Canada. This may influence the cultural diversity of the recommendations and the need to balance content from different regions.

6. **Director Trends**: We explored the top 10 directors by the number of movies and TV shows they have contributed to. This helps identify the most prolific creators on the platforms, which could be useful when personalizing recommendations for users who prefer certain directors.



```
first top 10 are:
1 >> Rajiv Chilaka
2 >> Raúl Campos, Jan Suter
3 >> Jack Hannah
4 >> Marcus Raboy
5 >> Paul Hoen
6 >> John Lasseter
7 >> Suhas Kadav
8 >> Jay Karas
9 >> Robert Vince
10 >> Cathy Garcia-Molina
```

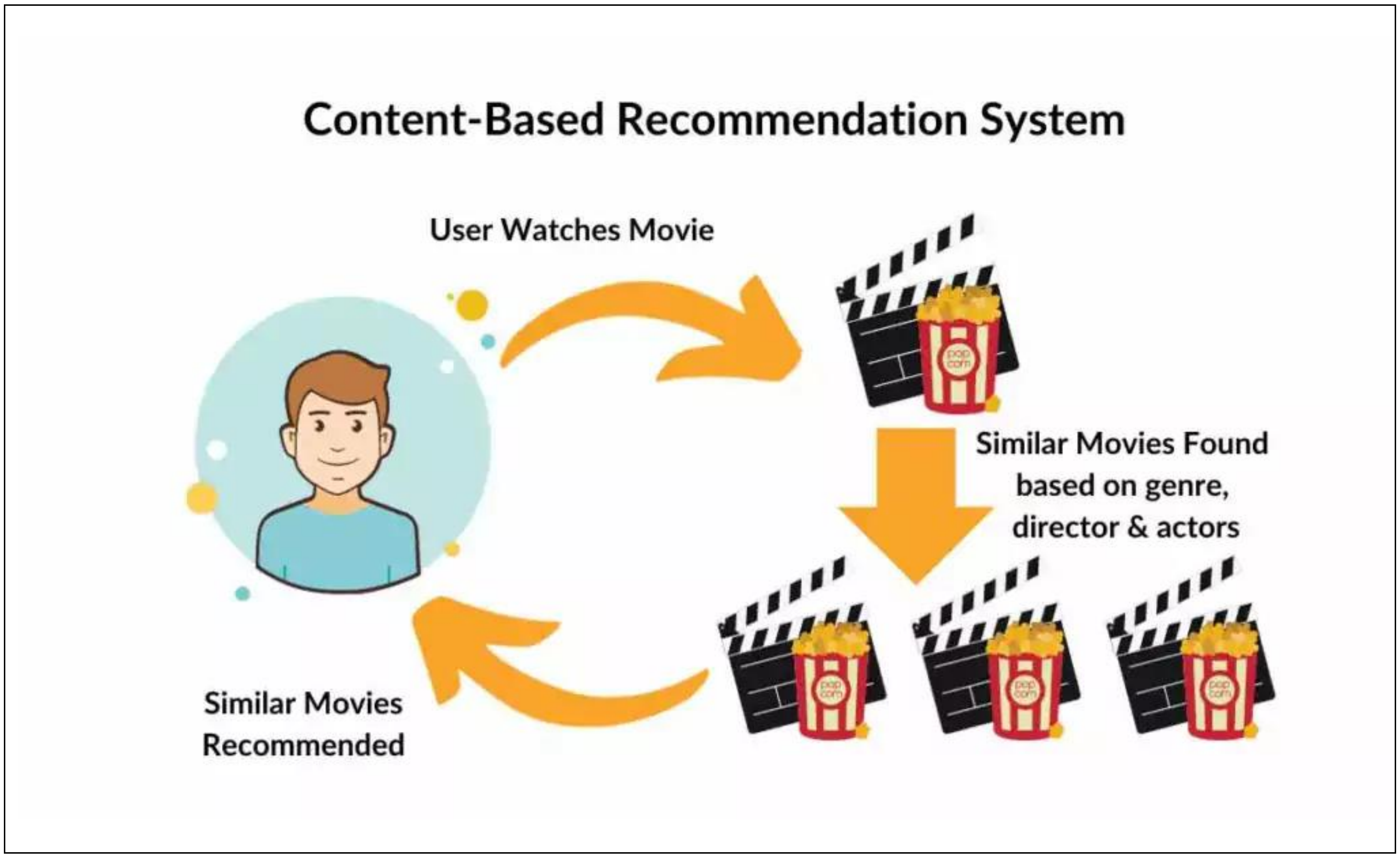


Pic (6) Director Trends

5. MACHINE LEARNING AND RECOMMENDATION MODELING

In this project, we developed a movie recommendation system using a combination of Natural Language Processing (NLP) and machine learning techniques. This report details the reasoning and methodology at each stage of the workflow, from data preparation to recommendation generation.

In related applications in movie recommendation system, next figure shows the used approach of content based recommendation system that designed to understand and cater to individual user preferences by analyzing the intrinsic features of items.

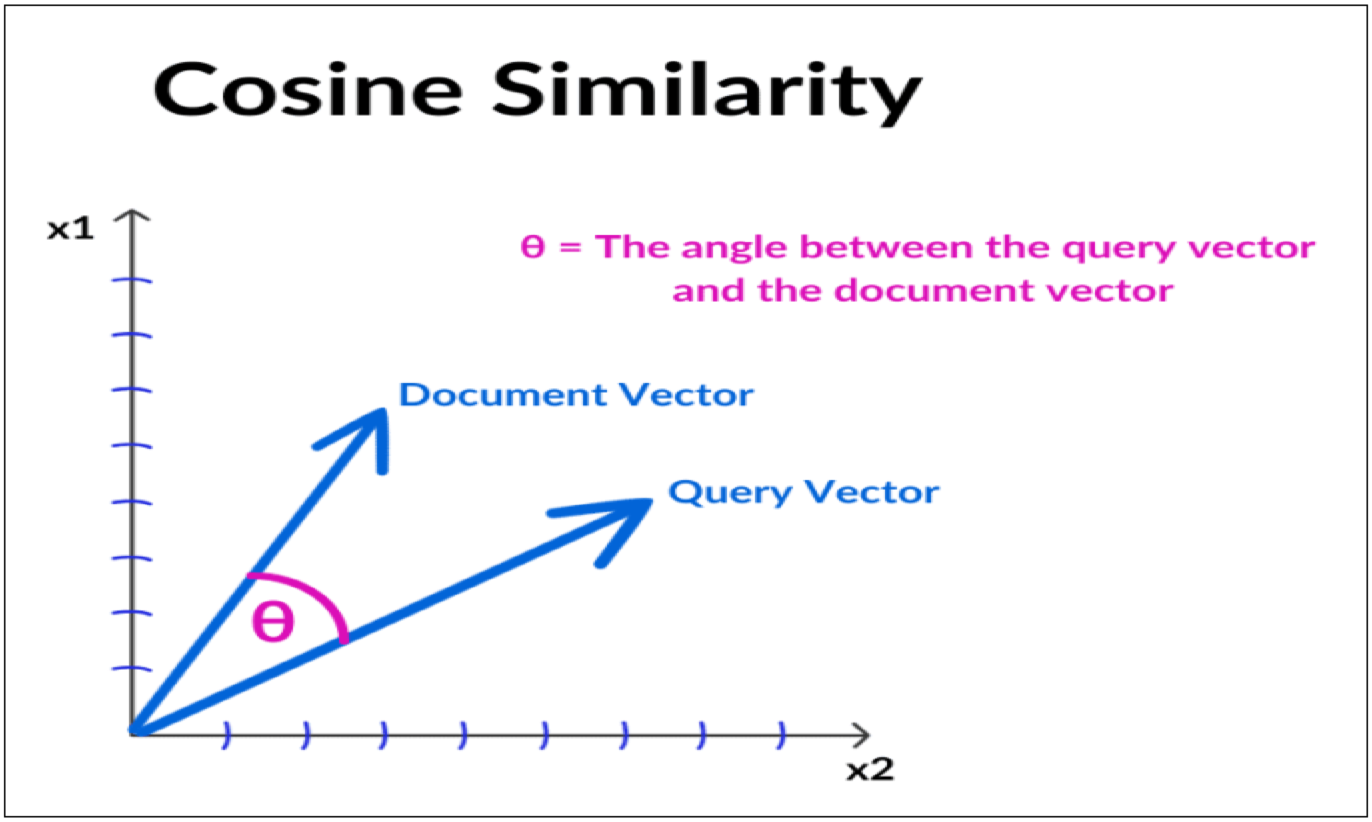

Content-Based Recommendation System
User Watches Movie
Similar Movies Found based on genre, director & actors
Similar Movies Recommended

In item profiles there are some steps to follow in order to prepare a model for recommendations

- **Content Representation for Items:** identifying and encoding these features, our textual features in a movie recommendation system include genre, director, and otherass
- **Feature Extraction for Items**: Extracting meaningful features from the content is crucial in building to create a comprehensive representation that captures the essence of each item.

Working Mechanism of Content-Based Recommendation Systems

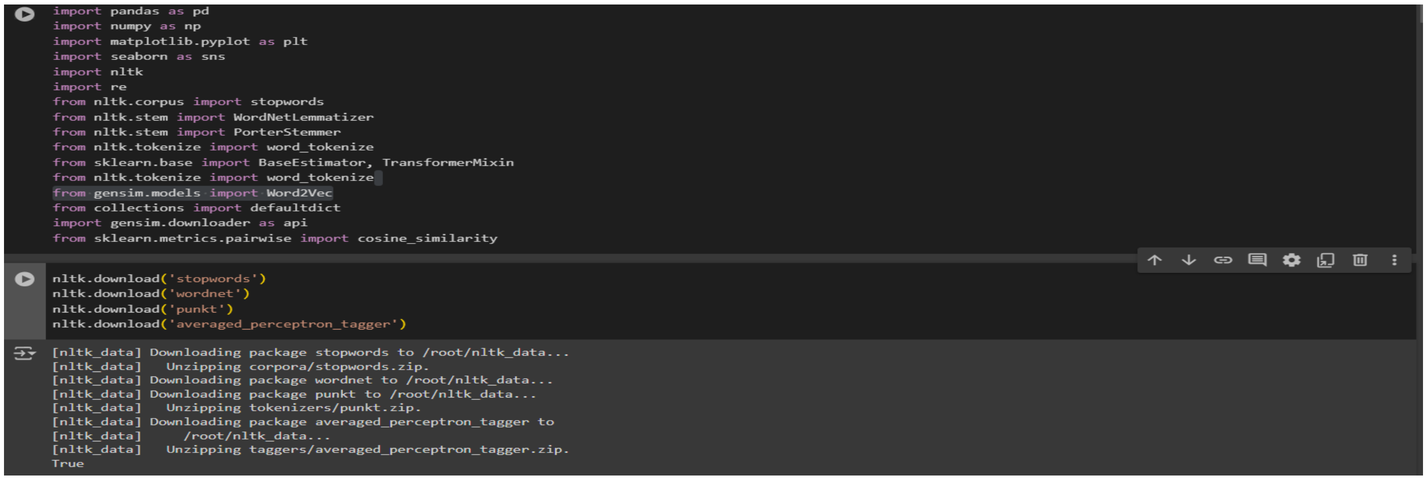
- **Cosine Similarity:** Cosine similarity, a widely used metric, quantifies the cosine of the angle between two vectors. In content-based systems, these vectors represent the user and item profiles. The closer the cosine similarity is to 1, the more similar the items are deemed to be in the feature space.


Cosine Similarity
x1
θ = The angle between the query vector and the document vector
Document Vector
Query Vector
θ
x2

- **Word Embeddings (Word2Vec):** Word embeddings are dense vector representations of words that capture semantic relationships between words. They can be used for content-based recommendation systems that involve textual data as words with similar meanings are represented as vectors close to each other in the embedding space. These embeddings can be used to describe items and users.

The next is our project main modeling steps:

1. **Importing Libraries:** The project required a diverse set of libraries for data manipulation, visualization, NLP, and machine learning. Key libraries included Pandas for data handling, Matplotlib and Seaborn for visualizations, and NLTK and Gensim for text processing and embedding generation. These libraries were crucial to handle both the dataset and the advanced NLP tasks efficiently.

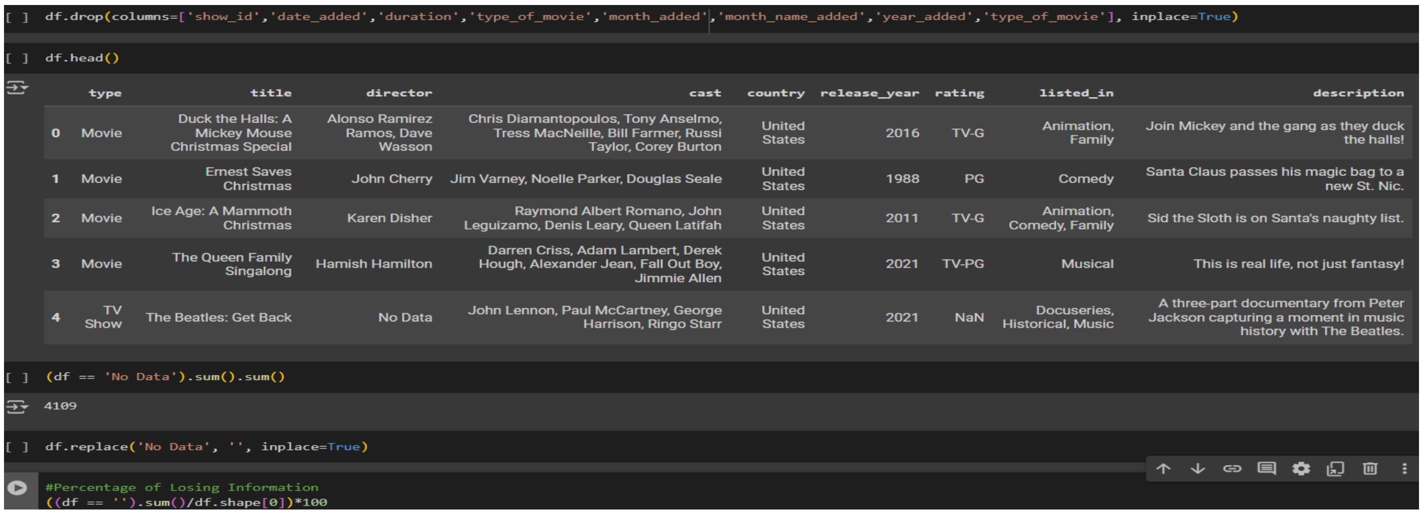


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from sklearn.base import BaseEstimator, TransformerMixin
from nltk.tokenize import word_tokenize
from gensim.models import Word2Vec
from collections import defaultdict
import gensim.downloader as api
from sklearn.metrics.pairwise import cosine_similarity
```

```
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

2. **Loading and Preparing Data:** The dataset was loaded and a deep copy was created to ensure data integrity. The initial stages focused on feature engineering, including removing redundant columns and addressing missing data, which was essential for improving data quality and reducing noise. In particular, we removed less informative columns and replaced missing values with appropriate placeholders.



```
df.drop(columns=['show_id','date_added','duration','type_of_movie','month_added','month_name_added','year_added','type_of_movie'], inplace=True)
```

```
df.head()
```

	type	title	director	cast	country	release_year	rating	listed_in	description
0	Movie	Duck the Halls: A Mickey Mouse Christmas Special	Alonso Ramirez Ramos, Dave Wasson	Chris Diamantopoulos, Tony Anselmo, Tress MacNeille, Bill Farmer, Russi Taylor, Corey Burton	United States	2016	TV-G	Animation, Family	Join Mickey and the gang as they duck the halls!
1	Movie	Ernest Saves Christmas	John Cherry	Jim Varney, Noelle Parker, Douglas Seale	United States	1988	PG	Comedy	Santa Claus passes his magic bag to a new St. Nic.
2	Movie	Ice Age: A Mammoth Christmas	Karen Disher	Raymond Albert Romano, John Leguizamo, Denis Leary, Queen Latifah	United States	2011	TV-G	Animation, Comedy, Family	Sid the Sloth is on Santa's naughty list.
3	Movie	The Queen Family Singalong	Hamish Hamilton	Darren Criss, Adam Lambert, Derek Hough, Alexander Jean, Fall Out Boy, Jimmie Allen	United States	2021	TV-PG	Musical	This is real life, not just fantasy!
4	TV Show	The Beatles: Get Back	No Data	John Lennon, Paul McCartney, George Harrison, Ringo Starr	United States	2021	NaN	Docuseries, Historical, Music	A three-part documentary from Peter Jackson capturing a moment in music history with The Beatles.

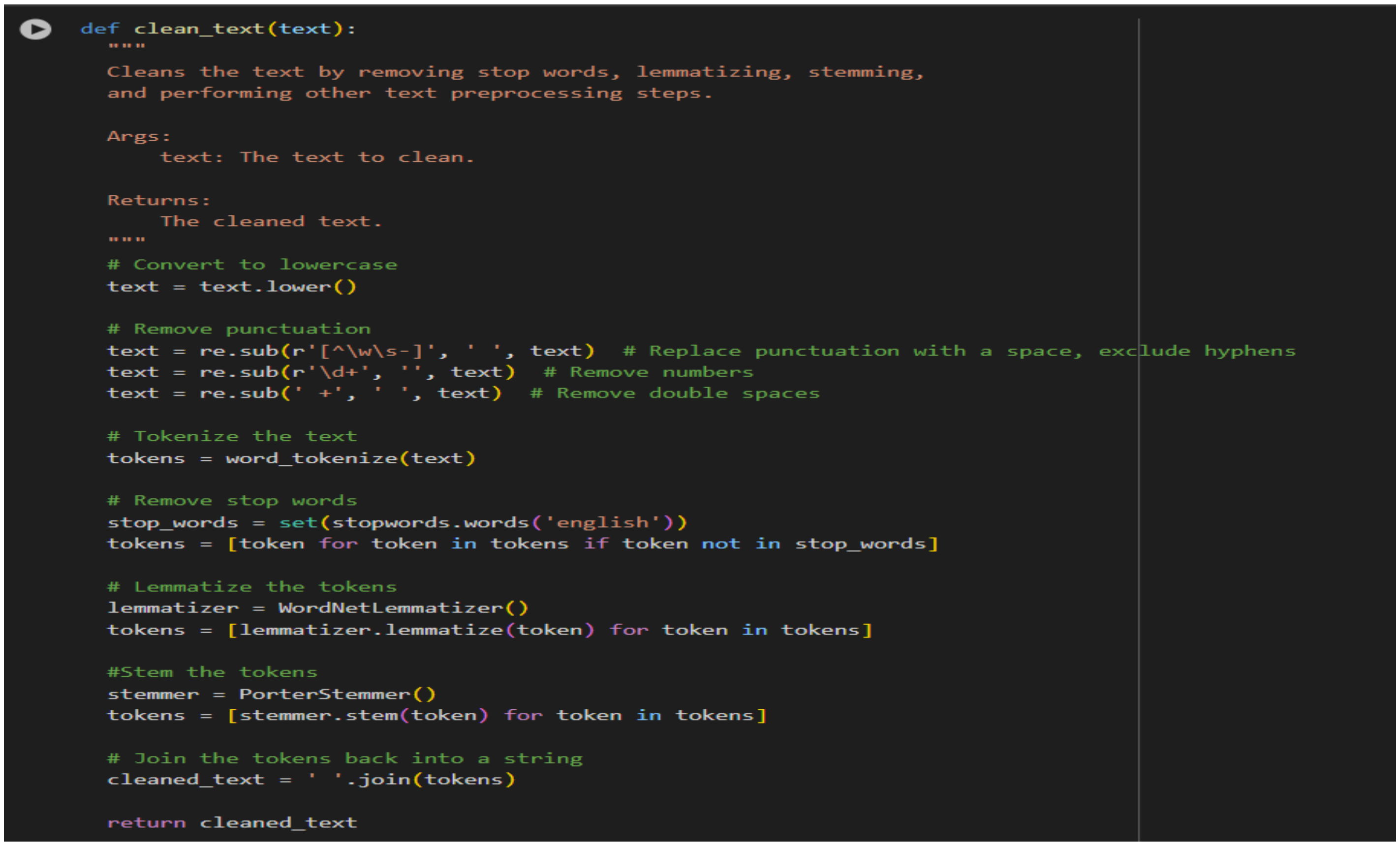
```
(df == 'No Data').sum().sum()
```

```
4109
```

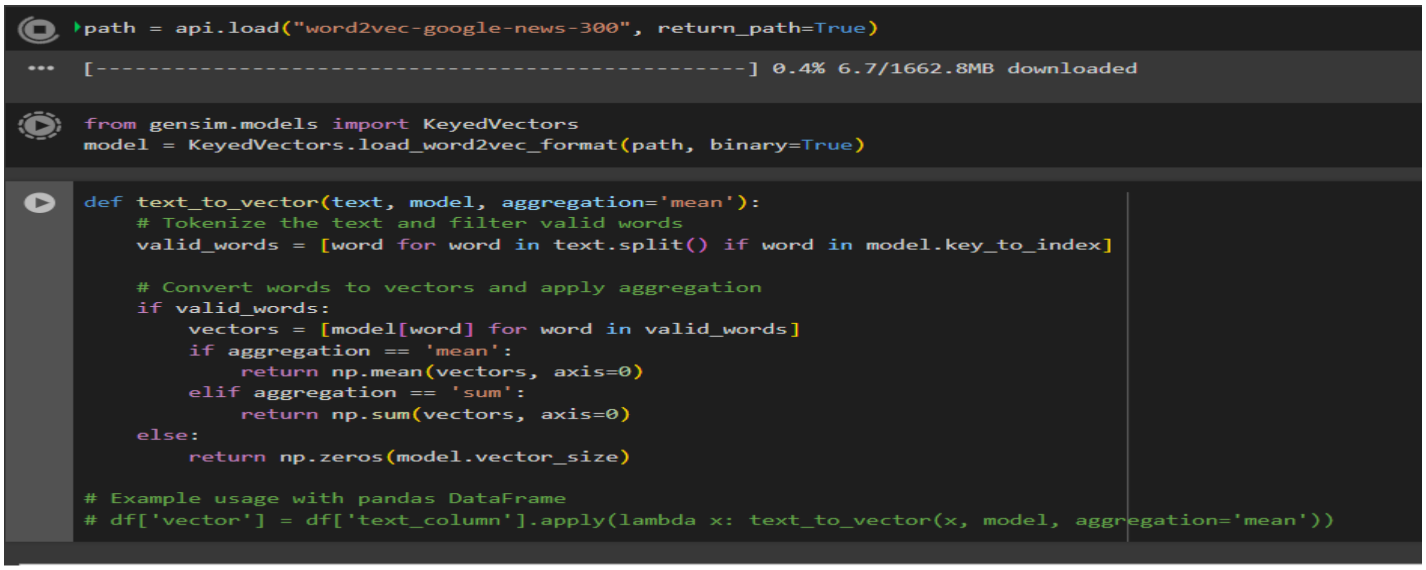
```
df.replace('No Data', '', inplace=True)
```

```
#Percentage of Losing Information
((df == '').sum()/df.shape[0])*100
```

3. **Text Cleaning and Vectorization:** A custom text-cleaning function was developed to process movie descriptions. This function involved converting text to lowercase, removing punctuation and numbers, and lemmatization and stemming the words to standardize them. Subsequently, we used the Word2Vec model to convert processed text into embeddings. This step enabled us to represent complex text data as numerical vectors, crucial for calculating similarity between movies



```python
def clean_text(text):
    """
    Cleans the text by removing stop words, lemmatizing, stemming,
    and performing other text preprocessing steps.

    Args:
        text: The text to clean.

    Returns:
        The cleaned text.
    """
    # Convert to lowercase
    text = text.lower()

    # Remove punctuation
    text = re.sub(r'[^\w\s-]', ' ', text)  # Replace punctuation with a space, exclude hyphens
    text = re.sub(r'\d+', '', text)  # Remove numbers
    text = re.sub(' +', ' ', text)  # Remove double spaces

    # Tokenize the text
    tokens = word_tokenize(text)

    # Remove stop words
    stop_words = set(stopwords.words('english'))
    tokens = [token for token in tokens if token not in stop_words]

    # Lemmatize the tokens
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(token) for token in tokens]

    #Stem the tokens
    stemmer = PorterStemmer()
    tokens = [stemmer.stem(token) for token in tokens]

    # Join the tokens back into a string
    cleaned_text = ' '.join(tokens)

    return cleaned_text
```



```python
path = api.load("word2vec-google-news-300", return_path=True)
```

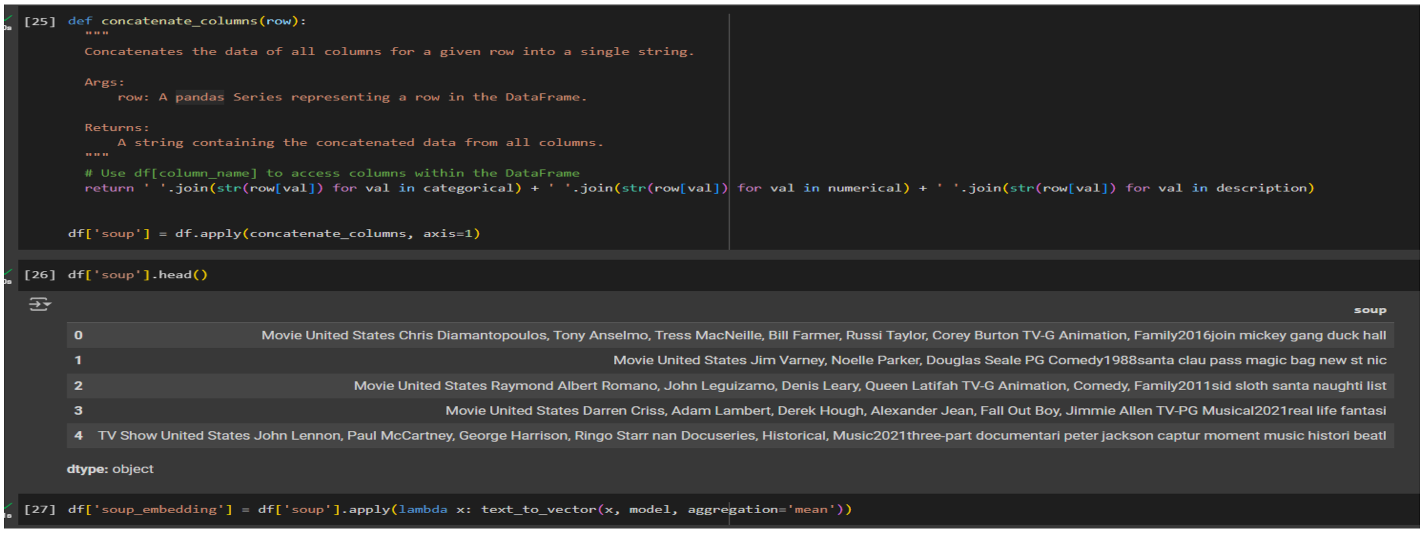
```
[--------------------------------------------------] 0.4% 6.7/1662.8MB downloaded
```

```python
from gensim.models import KeyedVectors
model = KeyedVectors.load_word2vec_format(path, binary=True)
```

```python
def text_to_vector(text, model, aggregation='mean'):
    # Tokenize the text and filter valid words
    valid_words = [word for word in text.split() if word in model.key_to_index]

    # Convert words to vectors and apply aggregation
    if valid_words:
        vectors = [model[word] for word in valid_words]
        if aggregation == 'mean':
            return np.mean(vectors, axis=0)
        elif aggregation == 'sum':
            return np.sum(vectors, axis=0)
    else:
        return np.zeros(model.vector_size)

# Example usage with pandas DataFrame
# df['vector'] = df['text_column'].apply(lambda x: text_to_vector(x, model, aggregation='mean'))
```

4. **Recommendation System Development:** The recommendation system was built using two primary approaches: Demographic Filtering and Content-Based Filtering. Given the lack of user ratings, Demographic Filtering was not applied. Content-Based Filtering was divided into two strategies: NLP-based "soup" vectorization and feature-based filtering. For the NLP-based approach, all relevant text features were concatenated and vectorized, then cosine similarity was calculated between vectors to

recommend similar movies. The feature-based method used PCA to reduce dimensionality for categorical and numerical features such as genre, cast, and release year. By applying these techniques, we captured the essence of the movies and minimized computational complexity.

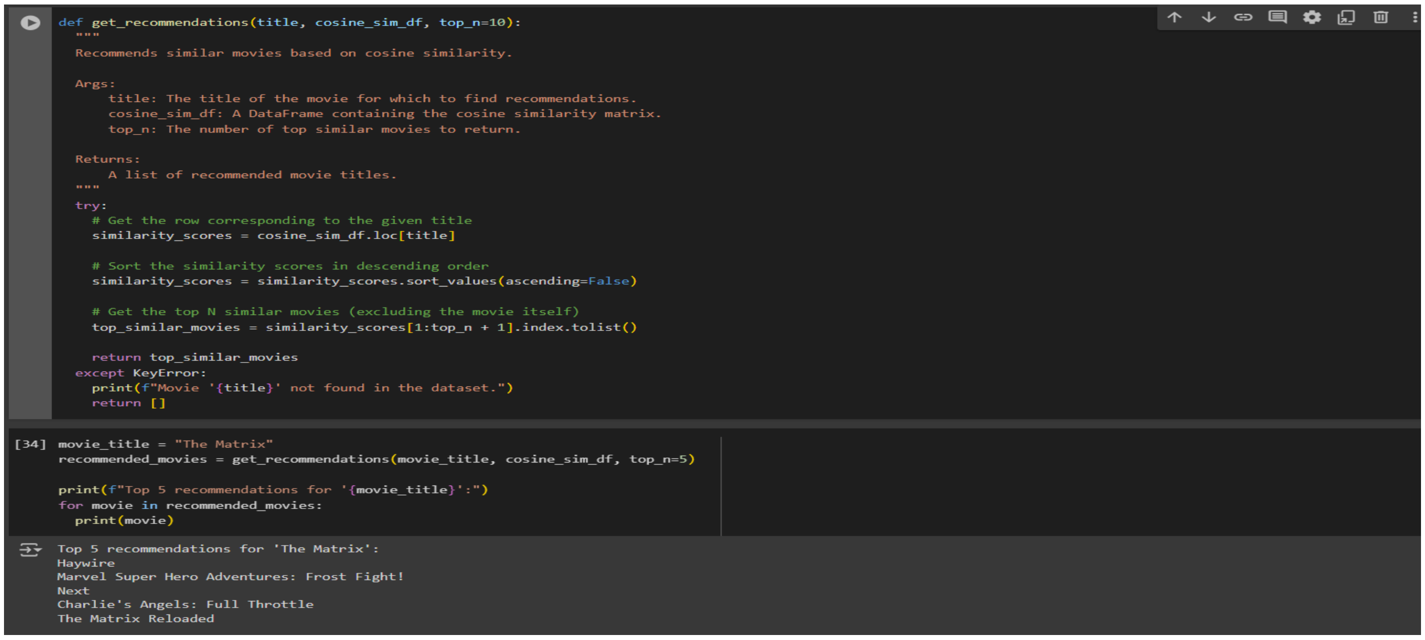


```
[25] def concatenate_columns(row):
       """
       Concatenates the data of all columns for a given row into a single string.

       Args:
           row: A pandas Series representing a row in the DataFrame.

       Returns:
           A string containing the concatenated data from all columns.
       """
       # Use df[column_name] to access columns within the DataFrame
       return ' '.join(str(row[val]) for val in categorical) + ' '.join(str(row[val]) for val in numerical) + ' '.join(str(row[val]) for val in description)


     df['soup'] = df.apply(concatenate_columns, axis=1)
```

```
[26] df['soup'].head()
```

```
                                                                                                                                                                   soup
0                Movie United States Chris Diamantopoulos, Tony Anselmo, Tress MacNeille, Bill Farmer, Russi Taylor, Corey Burton TV-G Animation, Family2016join mickey gang duck hall
1                                                                      Movie United States Jim Varney, Noelle Parker, Douglas Seale PG Comedy1988santa clau pass magic bag new st nic
2                                  Movie United States Raymond Albert Romano, John Leguizamo, Denis Leary, Queen Latifah TV-G Animation, Comedy, Family2011sid sloth santa naughti list
3                                                   Movie United States Darren Criss, Adam Lambert, Derek Hough, Alexander Jean, Fall Out Boy, Jimmie Allen TV-PG Musical2021real life fantasi
4 TV Show United States John Lennon, Paul McCartney, George Harrison, Ringo Starr nan Docuseries, Historical, Music2021three-part documentari peter jackson captur moment music histori beatl

dtype: object
```

```
[27] df['soup_embedding'] = df['soup'].apply(lambda x: text_to_vector(x, model, aggregation='mean'))
```

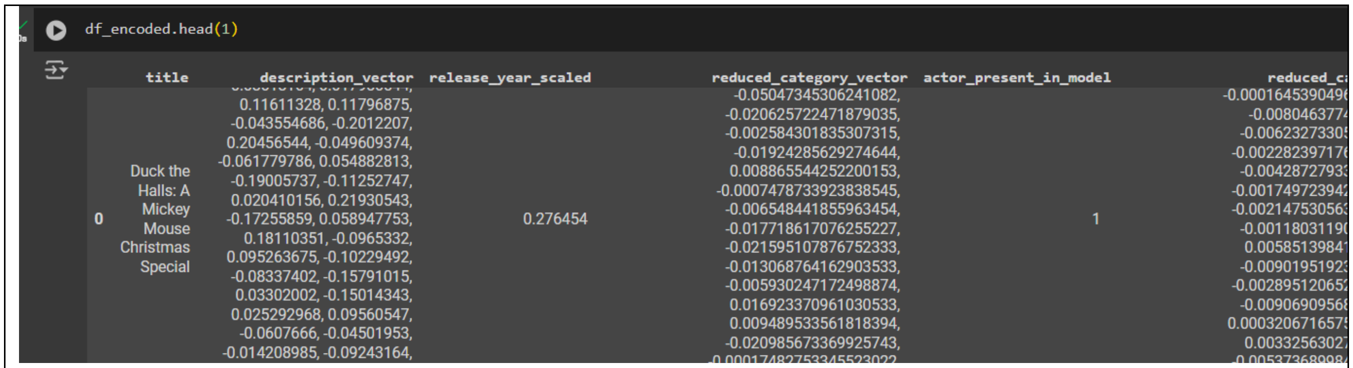


```
def get_recommendations(title, cosine_sim_df, top_n=10):
  """
  Recommends similar movies based on cosine similarity.

  Args:
      title: The title of the movie for which to find recommendations.
      cosine_sim_df: A DataFrame containing the cosine similarity matrix.
      top_n: The number of top similar movies to return.

  Returns:
      A list of recommended movie titles.
  """
  try:
    # Get the row corresponding to the given title
    similarity_scores = cosine_sim_df.loc[title]

    # Sort the similarity scores in descending order
    similarity_scores = similarity_scores.sort_values(ascending=False)

    # Get the top N similar movies (excluding the movie itself)
    top_similar_movies = similarity_scores[1:top_n + 1].index.tolist()

    return top_similar_movies
  except KeyError:
    print(f"Movie '{title}' not found in the dataset.")
    return []
```

```
[34] movie_title = "The Matrix"
     recommended_movies = get_recommendations(movie_title, cosine_sim_df, top_n=5)

     print(f"Top 5 recommendations for '{movie_title}':")
     for movie in recommended_movies:
       print(movie)
```

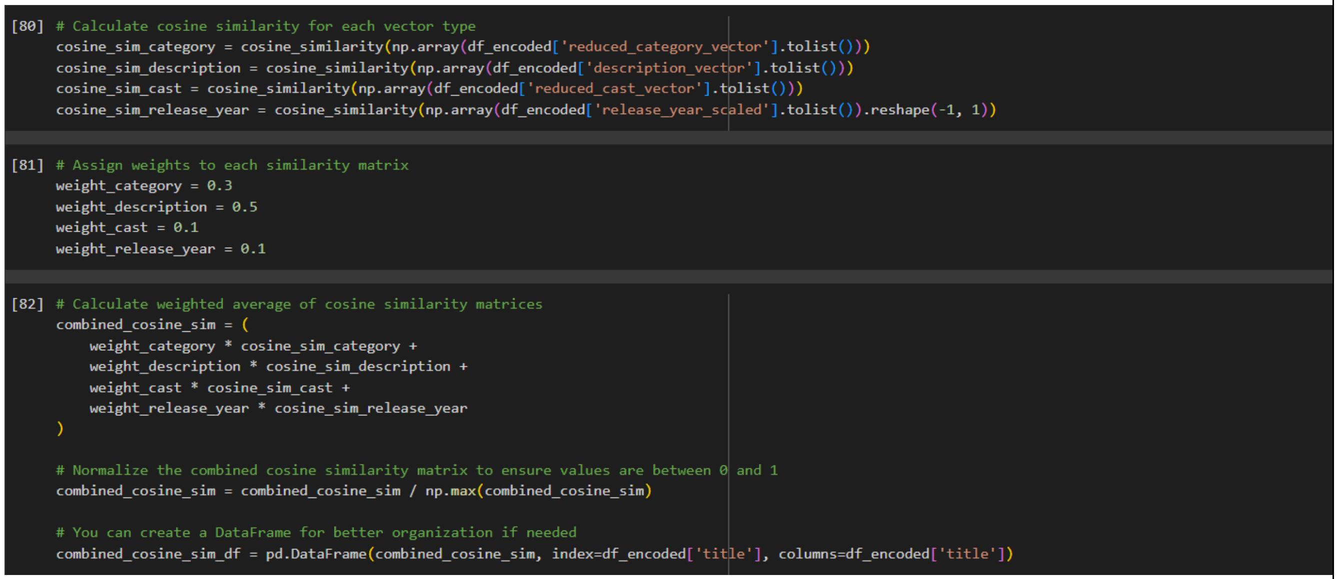
```
Top 5 recommendations for 'The Matrix':
Haywire
Marvel Super Hero Adventures: Frost Fight!
Next
Charlie's Angels: Full Throttle
The Matrix Reloaded
```

5. **Hybrid Similarity Matrix:** A final similarity matrix was created by weighing the similarity scores of each feature type. We combined category, description, cast, and release year similarities, assigning weights based on their assumed influence on recommendation quality. This hybrid approach allowed us to fine-tune recommendations based on specific attributes that align closely with user preferences.



```python
df_encoded.head(1)
```

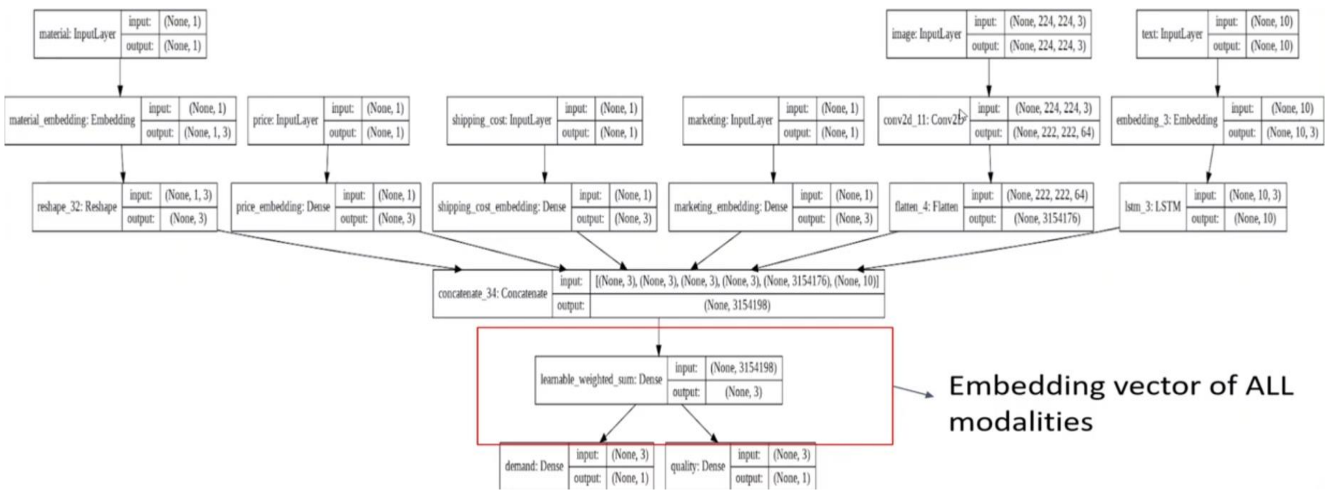
```
     title       description_vector  release_year_scaled     reduced_category_vector  actor_present_in_model      reduced_ca
              0.11611328, 0.11796875,                       -0.05047345306241082,                               -0.0001645390496
             -0.043554686, -0.2012207,                      -0.020625722471879035,                                 -0.0080463774
             0.20456544, -0.049609374,                      -0.002584301835307315,                                 -0.0062327330
           -0.061779786, 0.054882813,                        -0.01924285629274644,                                -0.002282397176
   Duck the  -0.19005737, -0.11252747,                       0.008865544252200153,                                -0.0042872793
   Halls: A  0.020410156, 0.21930543,                      -0.0007478733923838545,                                -0.001749723942
     Mickey  -0.17255859, 0.058947753,                      -0.006548441855963454,                                -0.002147530563
0     Mouse    0.18110351, -0.0965332,        0.276454      -0.017718617076255227,            1                  -0.00118031190
  Christmas  0.095263675, -0.10229492,                      -0.021595107876752333,                                 0.0058513984
    Special  -0.08337402, -0.15791015,                      -0.013068764162903533,                                -0.0090195192
              0.03302002, -0.15014343,                      -0.005930247172498874,                                -0.002895120652
             0.025292968, 0.09560547,                        0.016923370961030533,                                -0.0090690956
             -0.0607666, -0.04501953,                        0.009489533561818394,                                0.000320671657
           -0.014208985, -0.09243164,                       -0.020985673369925743,                                 0.0033256302
```



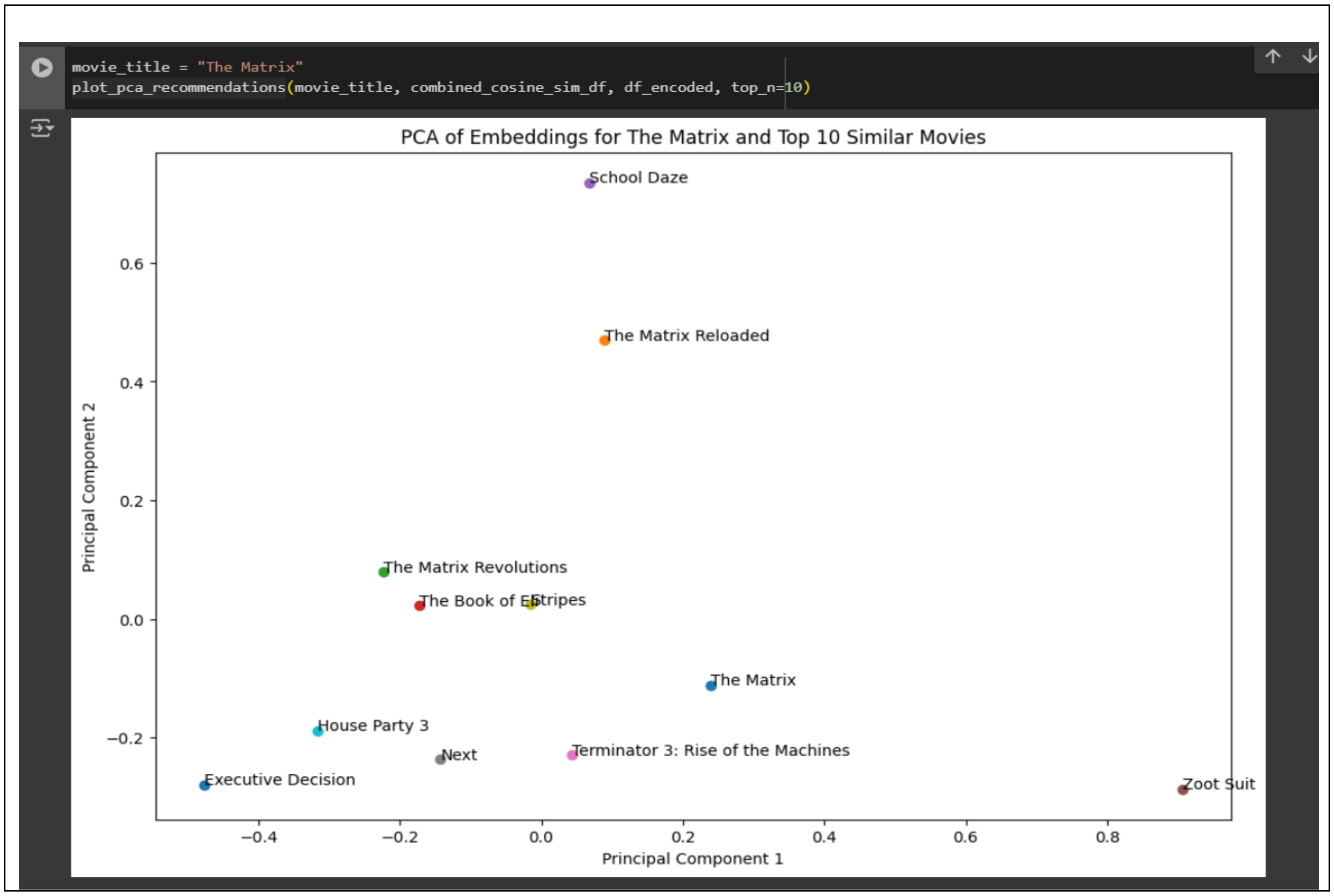
```python
[80] # Calculate cosine similarity for each vector type
     cosine_sim_category = cosine_similarity(np.array(df_encoded['reduced_category_vector'].tolist()))
     cosine_sim_description = cosine_similarity(np.array(df_encoded['description_vector'].tolist()))
     cosine_sim_cast = cosine_similarity(np.array(df_encoded['reduced_cast_vector'].tolist()))
     cosine_sim_release_year = cosine_similarity(np.array(df_encoded['release_year_scaled'].tolist()).reshape(-1, 1))
```

```python
[81] # Assign weights to each similarity matrix
     weight_category = 0.3
     weight_description = 0.5
     weight_cast = 0.1
     weight_release_year = 0.1
```

```python
[82] # Calculate weighted average of cosine similarity matrices
     combined_cosine_sim = (
         weight_category * cosine_sim_category +
         weight_description * cosine_sim_description +
         weight_cast * cosine_sim_cast +
         weight_release_year * cosine_sim_release_year
     )

     # Normalize the combined cosine similarity matrix to ensure values are between 0 and 1
     combined_cosine_sim = combined_cosine_sim / np.max(combined_cosine_sim)

     # You can create a DataFrame for better organization if needed
     combined_cosine_sim_df = pd.DataFrame(combined_cosine_sim, index=df_encoded['title'], columns=df_encoded['title'])
```


material: InputLayer | input: (None, 1) | output: (None, 1)
image: InputLayer | input: (None, 224, 224, 3) | output: (None, 224, 224, 3)
text: InputLayer | input: (None, 10) | output: (None, 10)
material_embedding: Embedding | input: (None, 1) | output: (None, 1, 3)
price: InputLayer | input: (None, 1) | output: (None, 1)
shipping_cost: InputLayer | input: (None, 1) | output: (None, 1)
marketing: InputLayer | input: (None, 1) | output: (None, 1)
conv2d_11: Conv2D | input: (None, 224, 224, 3) | output: (None, 222, 222, 64)
embedding_3: Embedding | input: (None, 10) | output: (None, 10, 3)
reshape_32: Reshape | input: (None, 1, 3) | output: (None, 3)
price_embedding: Dense | input: (None, 1) | output: (None, 3)
shipping_cost_embedding: Dense | input: (None, 1) | output: (None, 3)
marketing_embedding: Dense | input: (None, 1) | output: (None, 3)
flatten_4: Flatten | input: (None, 222, 222, 64) | output: (None, 3154176)
lstm_3: LSTM | input: (None, 10, 3) | output: (None, 10)
concatenate_34: Concatenate | input: [(None, 3), (None, 3), (None, 3), (None, 3), (None, 3154176), (None, 10)] | output: (None, 3154198)
learnable_weighted_sum: Dense | input: (None, 3154198) | output: (None, 3)
demand: Dense | input: (None, 3) | output: (None, 1)
quality: Dense | input: (None, 3) | output: (None, 1)
Embedding vector of ALL modalities

6. **Visualization and Evaluation:** The final step involved visualizing the cosine similarity matrix using a heatmap to evaluate similarity patterns. PCA was also used to project embeddings into two dimensions for easier visualization, enabling insights into how movies cluster based on their features. The model's effectiveness was further validated by manually assessing recommendations for select movies, confirming its ability to suggest relevant and similar titles.

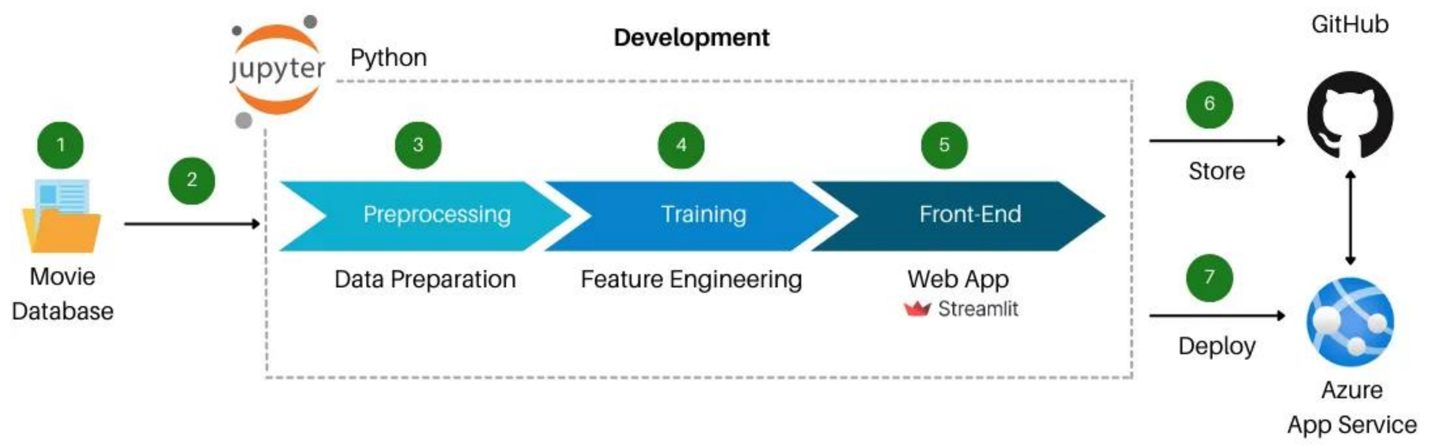


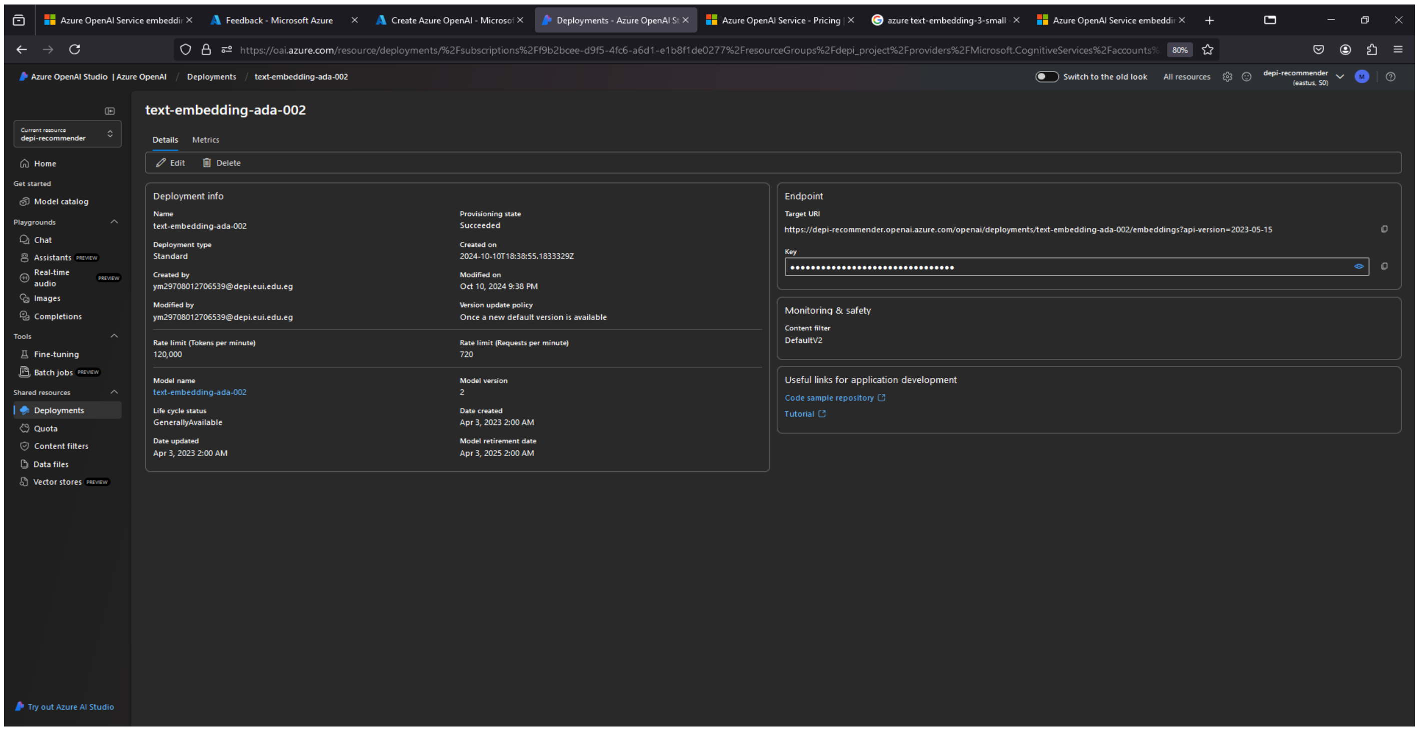
```
movie_title = "The Matrix"
plot_pca_recommendations(movie_title, combined_cosine_sim_df, df_encoded, top_n=10)
```

Overall, this recommendation system effectively combines NLP and machine learning, demonstrating how hybrid techniques can improve recommendation accuracy and cater to diverse user preferences.

6. BUSINESS AZURE INTEGRATION AND DEPLOYMENT

In this advanced solution, Azure OpenAI is integrated into a movie recommendation system, providing scalability, efficiency, and robust computational resources that enhance the overall quality and speed of data processing. By harnessing Azure's capabilities, this approach effectively manages large datasets, enabling us to deliver precise and personalized recommendations with ease as seen in next figure. The implementation leverages Azure OpenAI's powerful embedding models, allowing for the efficient processing and comparison of text data, while the deployment of the application on the web is streamlined using Azure Web Services and Streamlit for user interaction.




Azure OpenAI Service embeddi
Feedback - Microsoft Azure
Create Azure OpenAI - Microsof
Deployments - Azure OpenAI St
Azure OpenAI Service - Pricing |
azure text-embedding-3-small
Azure OpenAI Service embeddi
80%
Azure OpenAI Studio | Azure OpenAI / Deployments / text-embedding-ada-002
Switch to the old look
All resources
depi-recommender (eastus, S0)
Current resource depi-recommender
Home
Get started
Model catalog
Playgrounds
Chat
Assistants PREVIEW
Real-time audio PREVIEW
Images
Completions
Tools
Fine-tuning
Batch jobs PREVIEW
Shared resources
Deployments
Quota
Content filters
Data files
Vector stores PREVIEW
Try out Azure AI Studio
Details
Metrics
Edit
Delete

https://oai.azure.com/resource/deployments/%2Fsubscriptions%2Ff9b2bcee-d9f5-4fc6-a6d1-e1b8f1de0277%2FresourceGroups%2Fdepi_project%2Fproviders%2FMicrosoft.CognitiveServices%2Faccounts%

text-embedding-ada-002

Deployment info

Name text-embedding-ada-002	Provisioning state Succeeded
Deployment type Standard	Created on 2024-10-10T18:38:55.1833329Z
Created by ym29708012706539@depi.eui.edu.eg	Modified on Oct 10, 2024 9:38 PM
Modified by ym29708012706539@depi.eui.edu.eg	Version update policy Once a new default version is available
Rate limit (Tokens per minute) 120,000	Rate limit (Requests per minute) 720
Model name text-embedding-ada-002	Model version 2
Life cycle status GenerallyAvailable	Date created Apr 3, 2023 2:00 AM
Date updated Apr 3, 2023 2:00 AM	Model retirement date Apr 3, 2025 2:00 AM

Endpoint

Target URI
https://depi-recommender.openai.azure.com/openai/deployments/text-embedding-ada-002/embeddings?api-version=2023-05-15

Key
••••••••••••••••••••••••••••••••

Monitoring & safety

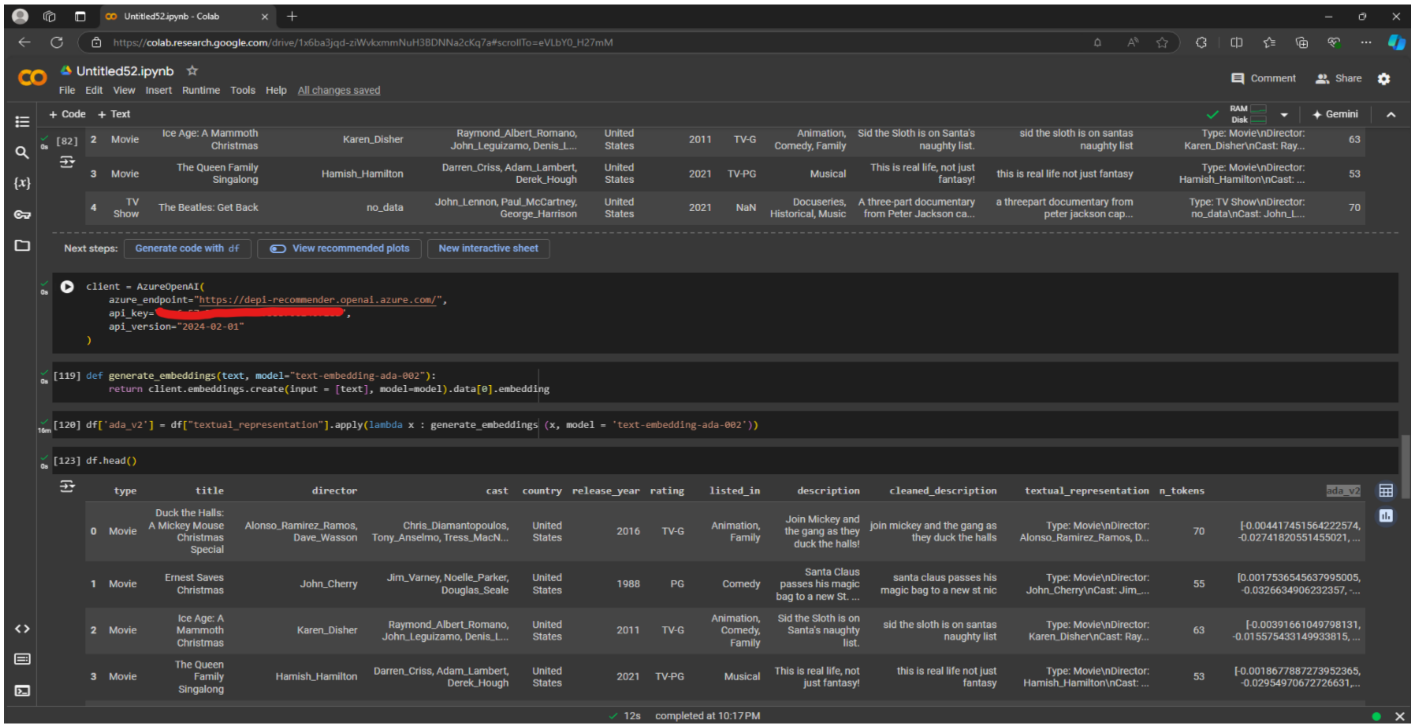
Content filter
DefaultV2

Useful links for application development

Code sample repository

Tutorial

To start, the necessary libraries are installed and imported, including OpenAI's Azure integration, Num2words, Pandas, and other essential machine learning and NLP libraries. These libraries facilitate data preprocessing, model integration, and the deployment pipeline. The Azure OpenAI API connection is then established, configured using environment variables, which securely manage access to the service endpoints.


Untitled52.ipynb - Colab
https://colab.research.google.com/drive/1x6ba3jqd-ziWvkxmmNuH3BDNNa2cKq7a#scrollTo=eVLbY0_H27mM
Untitled52.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Gemini
Next steps: Generate code with df · View recommended plots · New interactive sheet
12s completed at 10:17 PM

[82]

2	Movie	Ice Age: A Mammoth Christmas	Karen_Disher	Raymond_Albert_Romano, John_Leguizamo, Denis_L...	United States	2011	TV-G	Animation, Comedy, Family	Sid the Sloth is on Santa's naughty list.	sid the sloth is on santas naughty list	Type: Movie\nDirector: Karen_Disher\nCast: Ray...	63
3	Movie	The Queen Family Singalong	Hamish_Hamilton	Darren_Criss, Adam_Lambert, Derek_Hough	United States	2021	TV-PG	Musical	This is real life, not just fantasy!	this is real life not just fantasy	Type: Movie\nDirector: Hamish_Hamilton\nCast: ...	53
4	TV Show	The Beatles: Get Back	no_data	John_Lennon, Paul_McCartney, George_Harrison	United States	2021	NaN	Docuseries, Historical, Music	A three-part documentary from Peter Jackson ca...	a threepart documentary from peter jackson cap...	Type: TV Show\nDirector: no_data\nCast: John_L...	70

```
client = AzureOpenAI(
    azure_endpoint="https://depi-recommender.openai.azure.com/",
    api_key=[redacted],
    api_version="2024-02-01"
)
```

```
[119] def generate_embeddings(text, model="text-embedding-ada-002"):
          return client.embeddings.create(input = [text], model=model).data[0].embedding
```

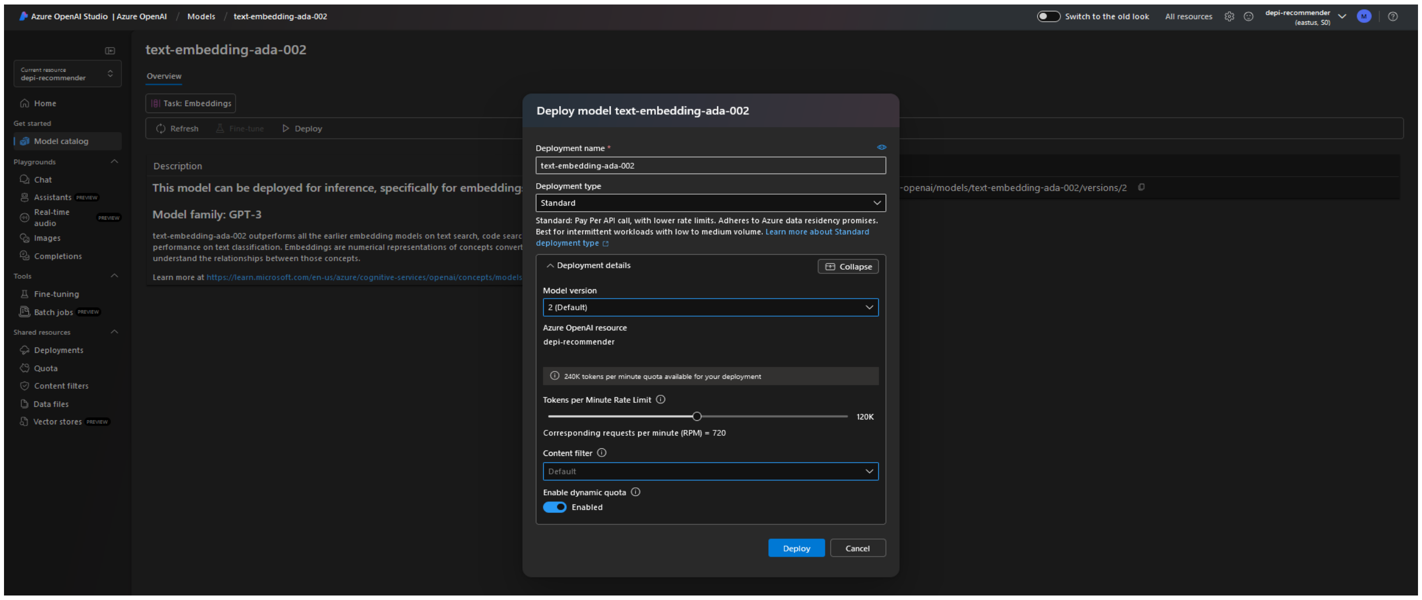
```
[120] df['ada_v2'] = df["textual_representation"].apply(lambda x : generate_embeddings (x, model = 'text-embedding-ada-002'))
```

```
[123] df.head()
```

	type	title	director	cast	country	release_year	rating	listed_in	description	cleaned_description	textual_representation	n_tokens	ada_v2
0	Movie	Duck the Halls: A Mickey Mouse Christmas Special	Alonso_Ramirez_Ramos, Dave_Wasson	Chris_Diamantopoulos, Tony_Anselmo, Tress_MacN...	United States	2016	TV-G	Animation, Family	Join Mickey and the gang as they duck the halls!	join mickey and the gang as they duck the halls	Type: Movie\nDirector: Alonso_Ramirez_Ramos, D...	70	[-0.004417451564222574, -0.02741820551455021, ...
1	Movie	Ernest Saves Christmas	John_Cherry	Jim_Varney, Noelle_Parker, Douglas_Seale	United States	1988	PG	Comedy	Santa Claus passes his magic bag to a new St. ...	santa claus passes his magic bag to a new st nic	Type: Movie\nDirector: John_Cherry\nCast: Jim_...	55	[0.0017536545637995005, -0.0326634906232357, -...
2	Movie	Ice Age: A Mammoth Christmas	Karen_Disher	Raymond_Albert_Romano, John_Leguizamo, Denis_L...	United States	2011	TV-G	Animation, Comedy, Family	Sid the Sloth is on Santa's naughty list.	sid the sloth is on santas naughty list	Type: Movie\nDirector: Karen_Disher\nCast: Ray...	63	[-0.003916610497981 31, -0.015575433149933815, ...
3	Movie	The Queen Family Singalong	Hamish_Hamilton	Darren_Criss, Adam_Lambert, Derek_Hough	United States	2021	TV-PG	Musical	This is real life, not just fantasy!	this is real life not just fantasy	Type: Movie\nDirector: Hamish_Hamilton\nCast: ...	53	[-0.0018677887273952365, -0.02954970672726631,...

The data is meticulously prepared for modeling by first cleaning and transforming it. Non-essential columns are dropped to reduce memory load, and missing values are replaced for consistency. A custom function converts categorical data into a uniform textual representation, aggregating each movie's attributes into a single text block. This function ensures that all movie data is normalized and formatted appropriately for embedding generation.

Azure OpenAI's text-embedding-ada-002 model is deployed to generate embeddings for these textual representations. These embeddings serve as high-dimensional vector representations of the movies, facilitating efficient similarity computations. By calculating the total number of tokens and their associated costs, we can ensure cost-effective usage of Azure OpenAI while still achieving high-quality results.



Azure OpenAI Studio | Azure OpenAI / Models / text-embedding-ada-002

text-embedding-ada-002

Description

This model can be deployed for inference, specifically for embedding

Model family: GPT-3

text-embedding-ada-002 outperforms all the earlier embedding models on text search, code searc
performance on text classification. Embeddings are numerical representations of concepts convert
understand the relationships between those concepts.

Learn more at https://learn.microsoft.com/en-us/azure/cognitive-services/openai/concepts/models

Deploy model text-embedding-ada-002

Deployment name *: text-embedding-ada-002

Deployment type: Standard

Standard: Pay Per API call, with lower rate limits. Adheres to Azure data residency promises. Best for intermittent workloads with low to medium volume. Learn more about Standard deployment type

Deployment details

Model version: 2 (Default)

Azure OpenAI resource: depi-recommender

240K tokens per minute quota available for your deployment

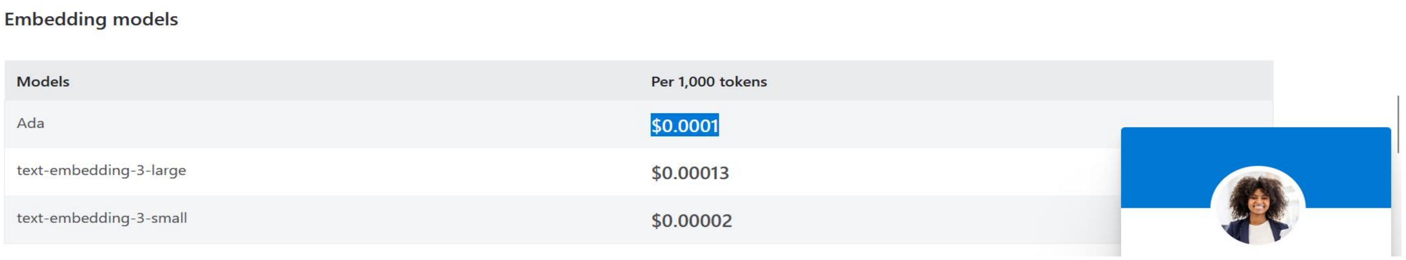
Tokens per Minute Rate Limit: 120K

Corresponding requests per minute (RPM) = 720

Content filter: Default

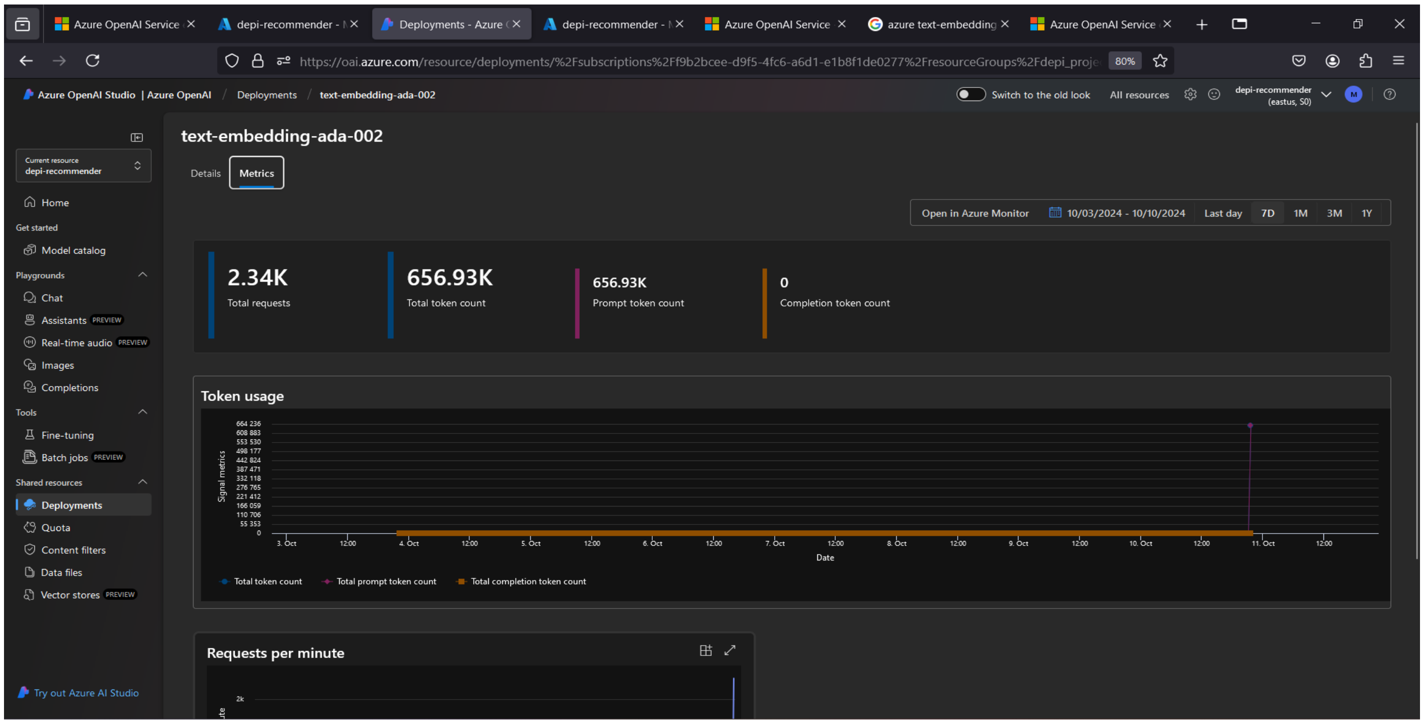
Enable dynamic quota: Enabled

Overview
Task: Embeddings
Refresh
Fine-tune
Deploy
Home
Model catalog
Chat
Deployments
Quota
Collapse
Cancel



Embedding models

Models	Per 1,000 tokens
Ada	$0.0001
text-embedding-3-large	$0.00013
text-embedding-3-small	$0.00002



https://oai.azure.com/resource/deployments/%2Fsubscriptions%2Ff9b2bcee-d9f5-4fc6-a6d1-e1b8f1de0277%2FresourceGroups%2Fdepi_proje

Azure OpenAI Studio | Azure OpenAI / Deployments / text-embedding-ada-002

text-embedding-ada-002

Details | Metrics

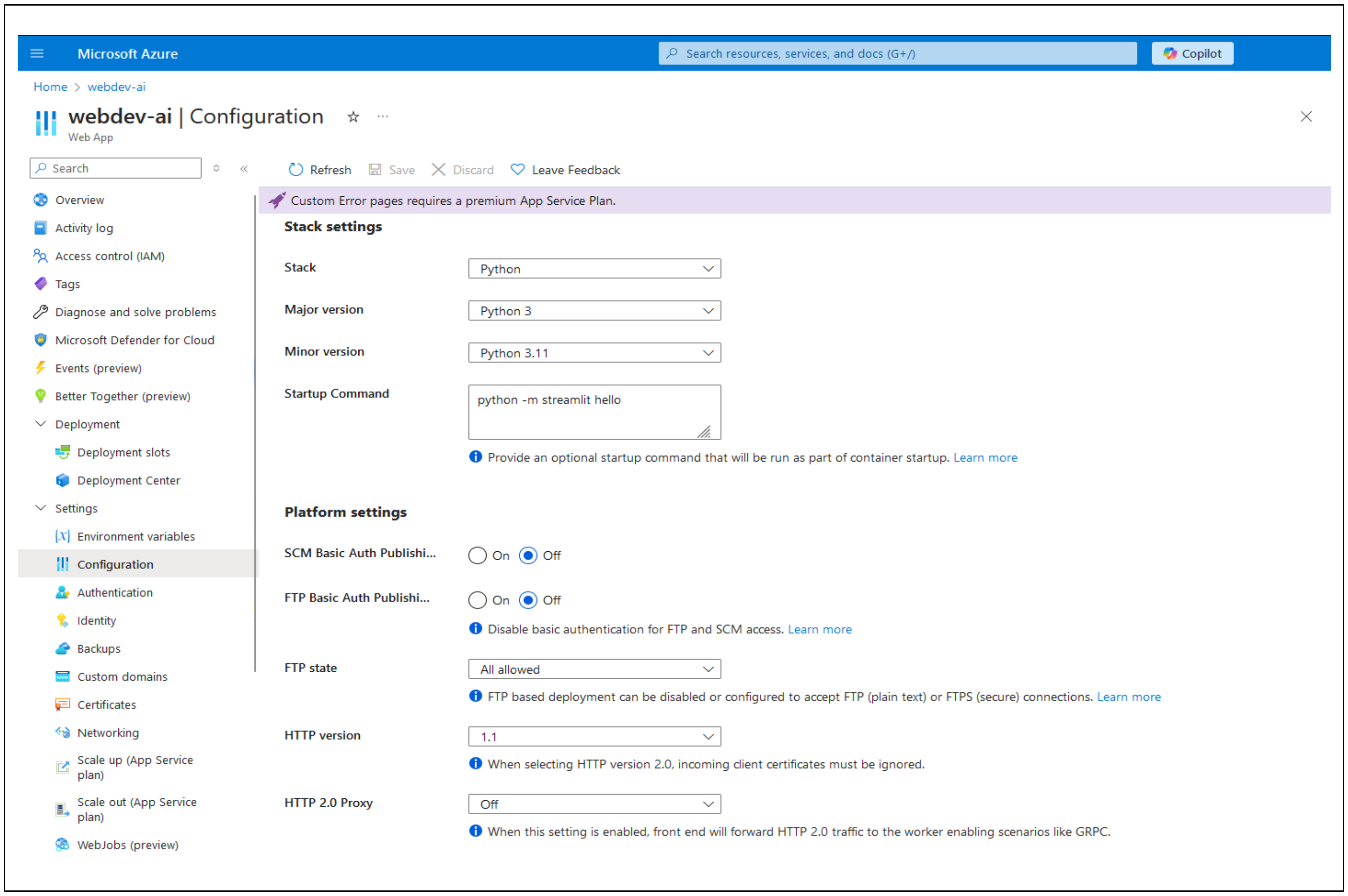
Open in Azure Monitor | 10/03/2024 - 10/10/2024 | Last day | 7D | 1M | 3M | 1Y

2.34K	656.93K	656.93K	0
Total requests	Total token count	Prompt token count	Completion token count

Token usage

For the recommendation process, the embeddings are analyzed using cosine similarity, providing a quantitative measure of how closely related each movie is to others. The resulting similarity scores are organized into a DataFrame, enabling quick access and streamlined recommendation retrieval. The user can specify a movie and the system will instantly display similar titles, delivering personalized suggestions.

In terms of deployment, the application is deployed to the web using Azure Web Services, offering a scalable and secure platform. Streamlit is then utilized to build an intuitive user interface, where users can input movie titles and receive real-time recommendations based on the computed embeddings. This integration ensures that the solution is not only powerful but also accessible and user-friendly.



webdev-ai | Configuration

Custom Error pages requires a premium App Service Plan.

Stack settings

Stack	Python
Major version	Python 3
Minor version	Python 3.11
Startup Command	python -m streamlit hello

Provide an optional startup command that will be run as part of container startup. Learn more

Platform settings

SCM Basic Auth Publishi...	On [✓ Off]
FTP Basic Auth Publishi...	On [✓ Off]

Disable basic authentication for FTP and SCM access. Learn more

FTP state	All allowed

FTP based deployment can be disabled or configured to accept FTP (plain text) or FTPS (secure) connections. Learn more

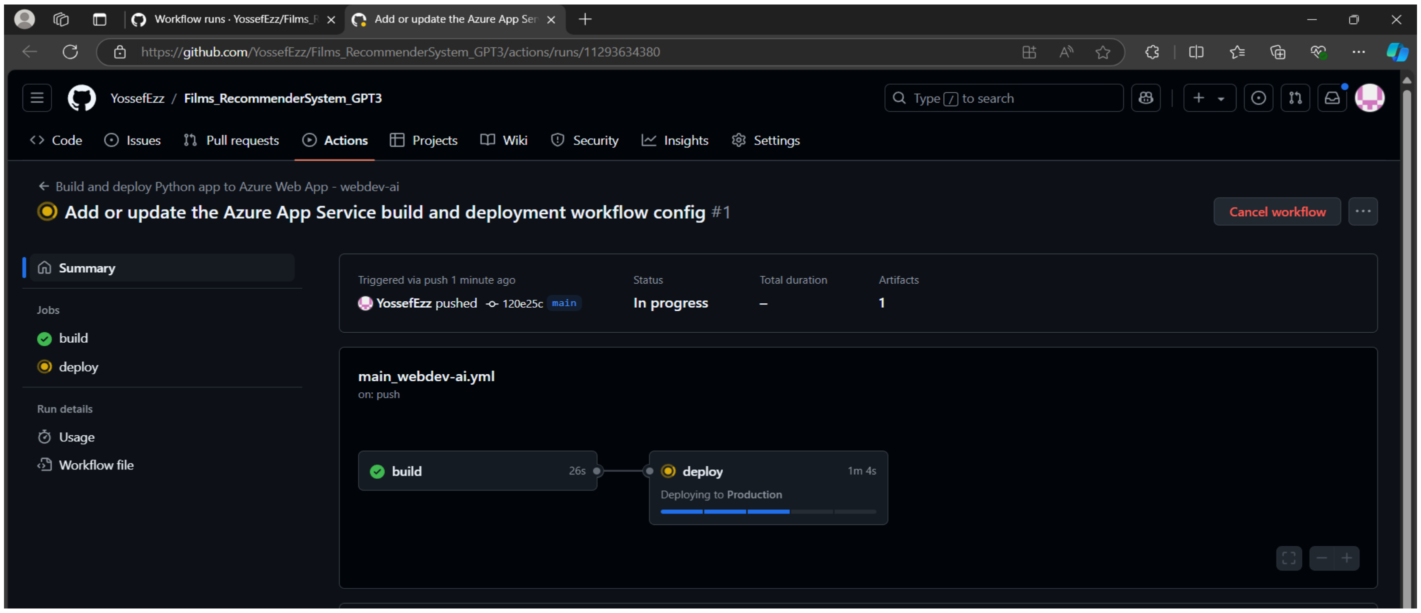
HTTP version	1.1

When selecting HTTP version 2.0, incoming client certificates must be ignored.

HTTP 2.0 Proxy	Off

When this setting is enabled, front end will forward HTTP 2.0 traffic to the worker enabling scenarios like GRPC.

Through Azure OpenAI, Azure Web Services, and Streamlit, this movie recommendation system becomes a comprehensive, efficient, and accessible solution that leverages advanced cloud-based technologies to enhance both the backend processing and user experience, making it a powerful tool for real-time recommendation delivery.


Workflow runs · YossefEzz/Films_F
Add or update the Azure App Ser
https://github.com/YossefEzz/Films_RecommenderSystem_GPT3/actions/runs/11293634380
YossefEzz / Films_RecommenderSystem_GPT3
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Build and deploy Python app to Azure Web App - webdev-ai

Add or update the Azure App Service build and deployment workflow config #1

Cancel workflow
Summary
Jobs
build
deploy
Run details
Usage
Workflow file

Triggered via push 1 minute ago
YossefEzz pushed 120e25c main

Status: In progress
Total duration: –
Artifacts: 1

main_webdev-ai.yml
on: push

build 26s
deploy 1m 4s
Deploying to Production

7. RESULTS AND PERFORMANCE

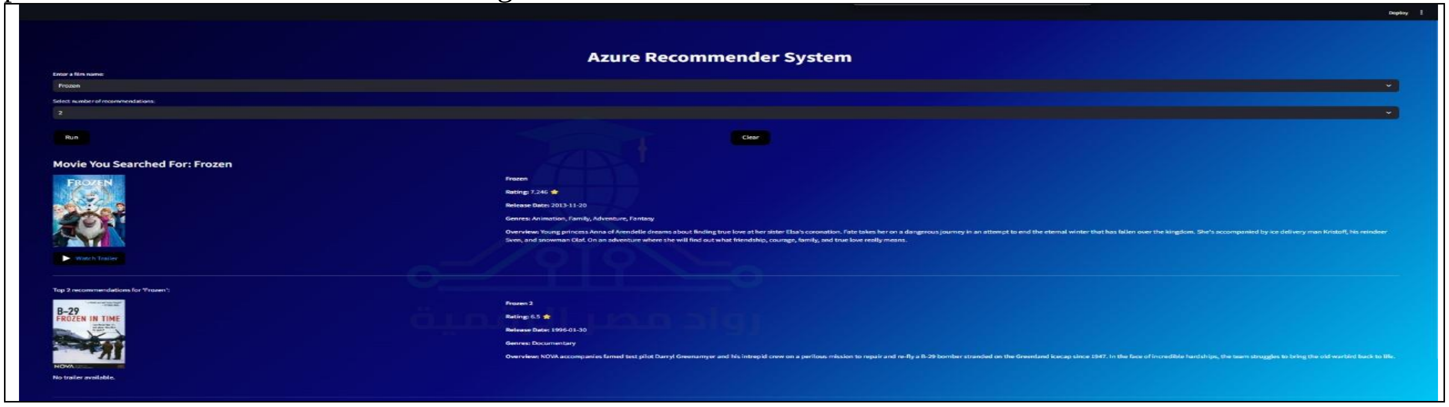
As shown in the attached image, the user searched for the movie "Frozen" using our Azure-based recommendation system. The system provided the following details for the movie, including its rating, release date, genre, and a brief overview of the plot.

Additionally, the system was instructed to return two recommendations related to "Frozen." Based on content similarity, the system recommended the movies:

1. Frozen 2 – A sequel to the original "Frozen" movie, rated 6.8, classified under genres such as Documentary. This recommendation is consistent with the user's interest in the "Frozen" series.

2. B-29 Frozen in Time – A documentary movie, rated 6.5, revolving around a WWII bomber that was stranded in the Arctic.

This output illustrates how the system analyzes the search query and leverages movie metadata to provide personalized recommendations that align with the user's search criteria.



Azure Recommender System

Enter a film name:
Frozen
Select number of recommendations:
2

Movie You Searched For: Frozen

Frozen
Rating: 7.246 ⭐
Release Date: 2013-11-20
Genres: Animation, Family, Adventure, Fantasy
Overview: Young princess Anna of Arendelle dreams about finding true love at her sister Elsa's coronation. Fate takes her on a dangerous journey in an attempt to end the eternal winter that has fallen over the kingdom. She's accompanied by ice delivery man Kristoff, his reindeer Sven, and snowman Olaf. On an adventure where she will find out what friendship, courage, family, and true love really means.

Top 2 recommendations for 'Frozen':

No trailer available.

Frozen 2
Rating: 6.5 ⭐
Release Date: 1996-01-30
Genres: Documentary
Overview: NOVA accompanies famed test pilot Darryl Greenamyer and his intrepid crew on a perilous mission to repair and re-fly a B-29 bomber stranded on the Greenland icecap since 1947. In the face of incredible hardships, the team struggles to bring the old warbird back to life.

8. CHALLENGES AND SOLUTIONS

1. **Cold Start Problem**

Strategies for New Users: The cold start problem, where the system struggles to provide accurate recommendations for new users, as in Content-based recommendation systems address this by leveraging demographic information, preferences from similar user groups, or hybrid approaches that incorporate collaborative filtering for initial recommendations. This ensures a more personalized user experience, even at the onset of their interaction with the platform.

2. **Over-Specialization**

Diversification Techniques: Content-based systems risk over-specialization, where users consistently recommend similar items, limiting their exposure to diverse content. Diversification techniques are employed to counter this. These may include incorporating randomness in recommendations, introducing exploration-exploitation strategies, or adjusting the weighting of features to promote variety in suggested items. Striking a balance between personalized and diverse recommendations is crucial for user satisfaction.

3. **Scalability**

Optimization and Parallelization: Scalability becomes a concern as user and item databases grow. Content-based systems must efficiently handle the increasing volume of data to provide real-time recommendations. Optimization techniques, such as feature selection and dimensionality reduction, contribute to streamlined processing. Parallelization of algorithms across distributed systems is another strategy, ensuring that the recommendation engine remains responsive and compelling even as the scale of data expands.

9. CONCLUSION AND FUTURE DIRECTIONS

In this project, we successfully developed a personalized recommendation system by leveraging the rich metadata from Netflix and Disney datasets. Despite the absence of user interaction data, we employed a content-based approach that utilized key features such as genre, cast, and description to deliver relevant movie suggestions. By incorporating advanced machine learning techniques and hybrid models, we enhanced the recommendation accuracy and relevance. The integration of Azure further allowed for seamless deployment, scaling, and continuous model updates, ensuring a robust system capable of handling real-time recommendations.

Key outcomes of the project included the creation of an efficient recommendation engine and the successful use of Azure for deployment and MLOps integration. The system demonstrated its capability to deliver personalized movie recommendations, which could enhance user satisfaction by reducing search time and improving content discovery. The project highlighted the strength of content-based filtering in the absence of explicit user preferences while also exploring collaborative filtering methods for future enhancement.

Future Directions

Looking ahead, there are several avenues for improving and expanding the recommendation system:

1. **Incorporation of User Interaction Data**: Future iterations of the system could integrate user ratings, watch history, or clickstream data, allowing the model to apply collaborative filtering methods more effectively. This would further enhance personalization by considering user behavior.

2. **Hybrid Recommendation System**: A deeper exploration of hybrid recommendation models could combine the strengths of both content-based and collaborative filtering approaches. This would result in more accurate suggestions that account for both user preferences and content similarities.

3. **Context-Aware Recommendations**: Introducing context-aware recommendation features that consider the user's current mood, time of day, or viewing history could add another layer of personalization, making the system more dynamic and responsive.

4. **Natural Language Processing (NLP) for Reviews**: Expanding the system to analyze user reviews and comments through NLP techniques could provide additional insights into user preferences and sentiments, which could improve recommendation quality.

5. **Diversification and Serendipity**: To avoid content saturation and repetitive recommendations, future models could include algorithms designed to promote serendipity, helping users discover diverse and unexpected content they might enjoy.

6. **Global Expansion and Multilingual Support**: Scaling the system to support multiple languages and incorporate regional content would make the platform more accessible and relevant to a global audience, reflecting the diversity of content on platforms like Netflix and Disney.

10. REFERENCES

Any literature, tools, datasets, or frameworks referenced during the project.

- Goyani, M., & Chaurasiya, N. (2020). A review of movie recommendation system: Limitations, Survey and Challenges. *ELCVIA: electronic letters on computer vision and image analysis*, *19*(3), 0018-37.
- https://spotintelligence.com/2023/11/15/content-based-recommendation-system/
- https://www.ijraset.com/best-journal/movie-recommendation-system-based-on-a-convolutional-neural-network
- https://learn.microsoft.com/en-us/azure/ai-studio/