Movie Recommendation System

Group Members With Role:

Anant kumar (Dataset, Data analysis, Model Training, Data preprocessing) Akash Kumar (Frontend Design, API intergration)

Dated : 8/5/24

Problem Statement

- Objective & Approach

  Develop a movie recommendation system inspired by platforms like Netflix, integrating user personalization with movie features (genre, tags, keywords, descriptions) from a “Kaggle” data-set.

- Implementation & Outcome :

  Utilize machine learning techniques to analyze date-set and movie meta data, resulting in a sophisticated recommendation system that enhances user engagement with personalized movie suggestions.

DataSet Extraction :

- The source for the datasets are “**Kaggle**”with links :
  - ➢ **Movies_data ->** https://www.kaggle.com/datasets/tmdb_5000_movies.csv
  - ➢ **Credits_data ->** https://www.kaggle.com/datasets/tmdb_5000_credits.csv
- Kaggle dataset review (TMDB movie dataset)
  - ➢ Columns : 20
  - ➢ Movies: 5000 (5k)
  - ➢ Dataset: 2 (movies_data.csv, credits_data.csv)
- Integrated the dataset with **TMDB** data set publically available.
- Merged both the dataset for more accuracy .

Data Analysis and preprocessing :

"Data analysis of a movie recommendation system data-set involves examining genres,null or duplicate values, movie keywords,overview, cast, crew and more patterns like this to derive insights for enhancing personalized movie suggestions."

Steps are as Discussed below :

1. Data Loading and Merging :
- Loaded movie data (`tmdb_5000_movies.csv`) and credits data (`tmdb_5000_credits.csv`) using pandas.
- Merged these datasets based on movie titles.

2. Data Cleaning and Feature Selection :
- Selected relevant columns (`movie_id`, `title`, `keywords`, `genres`, `overview`, `cast`, `crew`) for analysis.
- Removed rows with null values.

3. Data Transformation:
Converted string representations of lists (e.g., genres, keywords) into lists of strings using **ast.literal_eval**.
Extracted key information from lists (e.g., genre names, top cast members, directors) using custom functions.

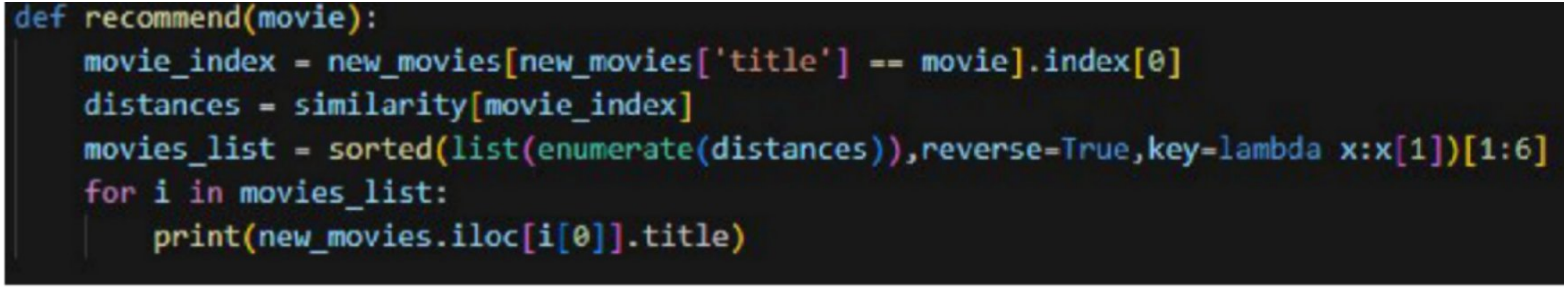
4. Text Processing:
Processed text data by splitting overviews into lists of words and handling issues like spaces between words. Created a consolidated 'tags' column by combining relevant text features.

5. Text Vectorization:
Used **CountVectorizer** from sklearn to convert text tags into numerical vectors.
Calculated cosine similarity between movie vectors to measure similarity between movies.

6. Recommendation Function:
Created a function (**recommend**) to recommend similar movies based on cosine similarity scores.

7. Saving Model:
Saved the processed data (**new_movies**) and the similarity matrix (**similarity**) using pickle for future use.

This approach involves comprehensive data preprocessing, text processing, and vectorization to build a movie recommendation system based on content similarity. The **recommend** function utilizes cosine similarity to suggest movies similar to a given movie title.



```python
def recommend(movie):
    movie_index = new_movies[new_movies['title'] == movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:6]
    for i in movies_list:
        print(new_movies.iloc[i[0]].title)
```

Models

The movie recommendation system described above utilizes **cosine similarity** as the core model for generating movie recommendations based on content similarity.

This system involves several key steps:

first, the textual data (such as movie tags derived from genres, cast, crew, and keywords) is processed and transformed into numerical vectors using `CountVectorizer` from `sklearn.feature_extraction.text`.

This vectorization step enables the representation of textual features in a format suitable for similarity calculations.
Next, the `cosine_similarity` function from `sklearn.metrics.pairwise` is applied to compute the cosine similarity between pairs of movie vectors. .

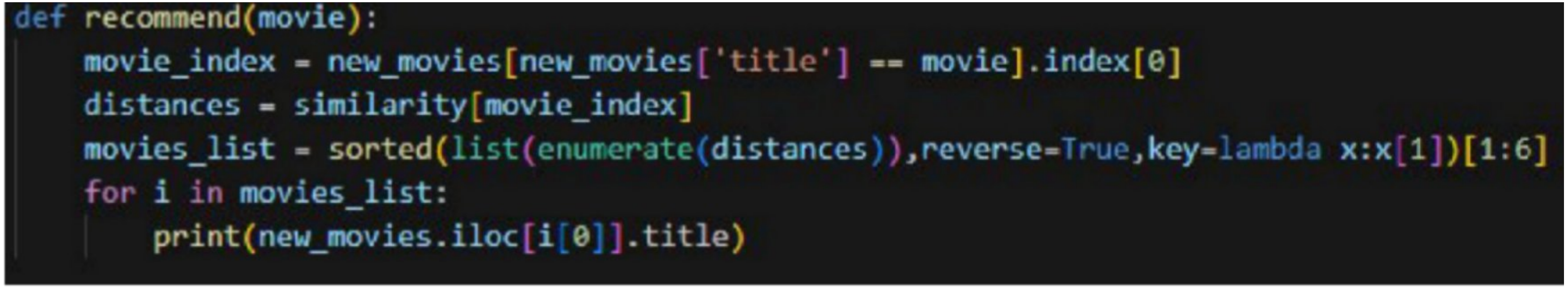
COSINE_SIMILARITY

Cosine similarity is a metric that measures the cosine of the angle between two vectors, providing a measure of similarity that is independent of vector magnitude.

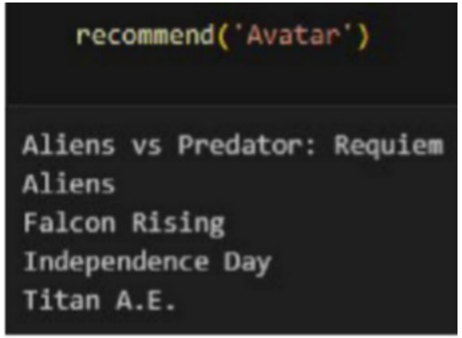
The `recommend` function leverages these cosine similarity scores to identify and recommend movies that are most similar to a given movie title, based on their textual content.

While this approach does not involve traditional model training in the sense of supervised learning, it focuses on leveraging vectorization and similarity calculation to deliver content-based movie recommendations. This content-based filtering technique is particularly useful for recommending items (in this case, movies) based on their intrinsic features rather than relying on user-item interactions or ratings.

Representation of the functions With examples:

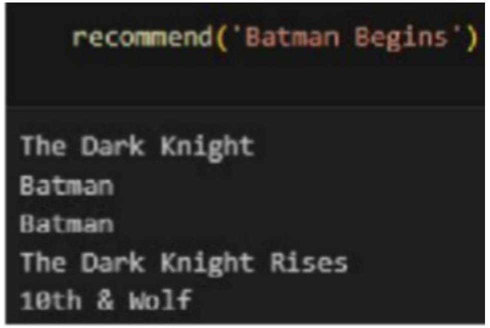


```python
def recommend(movie):
    movie_index = new_movies[new_movies['title'] == movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:6]
    for i in movies_list:
        print(new_movies.iloc[i[0]].title)
```

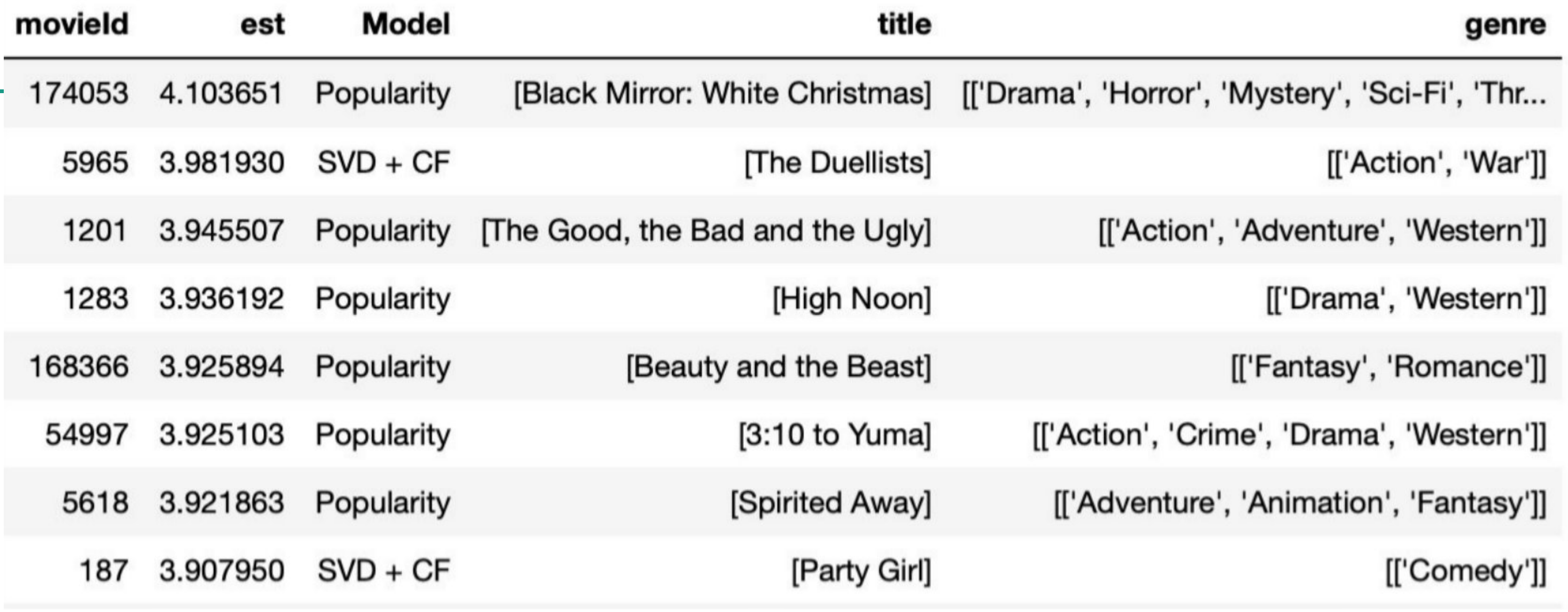


```
recommend('Avatar')

Aliens vs Predator: Requiem
Aliens
Falcon Rising
Independence Day
Titan A.E.
```

Some examples with the recommend func:



```
recommend('Batman Begins')

The Dark Knight
Batman
Batman
The Dark Knight Rises
10th & Wolf
```



movieId	est	Model	title	genre
174053	4.103651	Popularity	[Black Mirror: White Christmas]	[['Drama', 'Horror', 'Mystery', 'Sci-Fi', 'Thr...
5965	3.981930	SVD + CF	[The Duellists]	[['Action', 'War']]
1201	3.945507	Popularity	[The Good, the Bad and the Ugly]	[['Action', 'Adventure', 'Western']]
1283	3.936192	Popularity	[High Noon]	[['Drama', 'Western']]
168366	3.925894	Popularity	[Beauty and the Beast]	[['Fantasy', 'Romance']]
54997	3.925103	Popularity	[3:10 to Yuma]	[['Action', 'Crime', 'Drama', 'Western']]
5618	3.921863	Popularity	[Spirited Away]	[['Adventure', 'Animation', 'Fantasy']]
187	3.907950	SVD + CF	[Party Girl]	[['Comedy']]

Takeaways:

Key Takeaways from Building a Movie Recommendation System

Building a movie recommendation system involves several important steps and considerations, each contributing to the effectiveness and performance of the system. Here are some key takeaways from the process:

1. Data Preprocessing is Crucial: Cleaning and preprocessing the dataset are essential steps before modeling. This includes handling missing values, converting data formats, and extracting relevant features for analysis.

2. Text Processing for Feature Extraction: Textual data such as movie overviews, genres, and cast need to be processed to extract meaningful features. Techniques like tokenization, stemming, and entity extraction help in transforming text into usable features.

3. Utilizing Content-Based Filtering: Content-based filtering methods like cosine similarity allow for recommendations based on item similarity. This approach focuses on leveraging item features (e.g., movie tags) to recommend similar items, making it suitable for scenarios with sparse user-item interactions.

4. Vectorization for Numerical Representation: Transforming textual data into numerical vectors using techniques like `CountVectorizer` enables quantitative analysis and similarity calculations.

5. Custom Functions for Data Transformation: Writing custom functions to extract specific information from complex data structures (e.g., nested dictionaries) facilitates feature engineering and enhances the quality of input data for modeling.

6. Optimizing Recommendation Performance: Experimenting with parameters like `max_features` in
`CountVectorizer` and exploring different text processing techniques (e.g., stemming, stop-word removal) can optimize the performance and relevance of movie recommendations.

7. Persistence with Pickle: Saving processed data and model outputs using `pickle` ensures that the system's state can be preserved and reused for future recommendations or analyses.

8. Continuous Iteration and Improvement: Building a recommendation system is an iterative process. Continuously refining data preprocessing, feature extraction, and modeling techniques based on feedback and evaluation is key to improving recommendation accuracy and user satisfaction.

In summary, developing a movie recommendation system involves a blend of data preprocessing, text processing, vectorization, and content-based filtering techniques. Understanding and implementing these steps effectively are critical for building a robust and efficient recommendation system that delivers relevant and personalized movie suggestions to users.

Integrating Model into a Web Platform with Streamlit

Objective: Implement a movie recommendation system on the web using machine learning techniques.

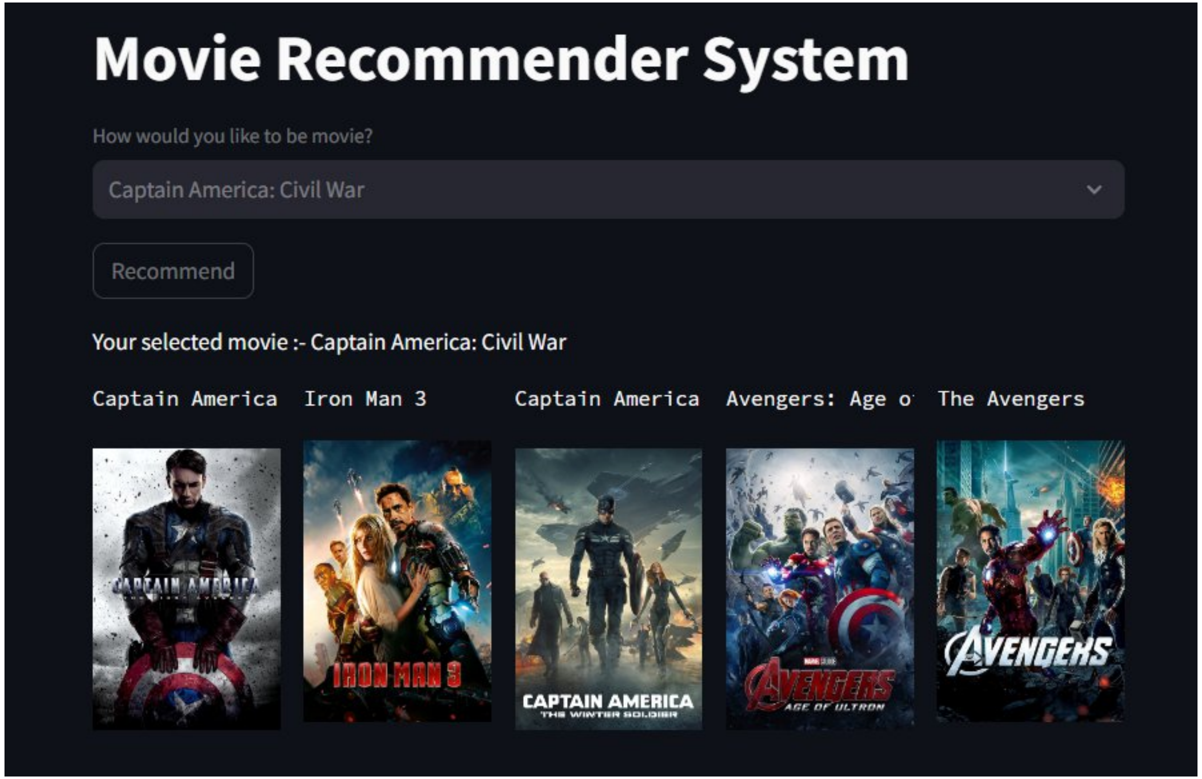
Framework Used : Leveraged Streamlit, a Python library for building interactive web applications.
Key Components:
- Frontend Design: Designed a user-friendly interface with input fields for movie titles.
- Backend Processing: Integrated pre-trained recommendation model using Streamlit's Python integration.
- User Interaction: Enabled real-time movie recommendations based on user input.

Benefits:
- Streamlined development process with Streamlit's intuitive APIs.
- Enhanced user experience through interactive movie suggestions.
- Facilitated iterative improvements based on user feedback.

Outcome:
- Engaging and responsive web application showcasing machine learning capabilities.
- Demonstrated potential for personalized recommendations in a user-friendly format.

Next Steps**:
- Continuously refine and optimize the recommendation algorithm.
- Explore additional features and enhancements based on user interactions and feedback.

Representation of Web-App


Movie Recommender System

How would you like to be movie?

Captain America: Civil War

Recommend

Your selected movie :- Captain America: Civil War

Mentor-ship & guidance:

This was our project Movie Recommendation system using classical Model
A Project Report in a partial fulfillment of the requirements for the award of the degree of
Bachelor in Computer Applications
Under the Guidance of
MAHENDRA DUTTA
BY
ANANT KUMAR, AKASH KUMAR, ABHAY KUMAR DUBEY

BOKARO STEEL CITY COLLEGE, BOKARO
Constituent Unit of Binod Bihari Mahato Koylanchal University, Dhanbad
In association with

ARDENT
COMPUTECH PVT. LTD.
High End Technology Training and Project