HARPY AEROSPACE INTERNSHIP

Aiot Project :RECOMMENDATION SYSTEM

By Sneya Gabreate S

2022506062

B.Tech Information Technology

Madras Institute of Technology

Recommendation System 1: Movie Similarity Model

Install and Import

```
!pip install -q tensorflow-recommenders
!pip install -q --upgrade tensorflow-datasets

import numpy as np
import tensorflow as tf
import tensorflow_datasets as tfds
import tensorflow_recommenders as tfrs
```

```
96.2/96.2 kB 1.6 MB/s eta 0:00:00
```

Loading the data

```
# Load movie metadata.
movies = tfds.load('movielens/100k-movies', split="train")

# Extract movie titles.
movies = movies.map(lambda x: x["movie_title"])

# Create movie title vocabulary.
movie_titles_vocabulary = tf.keras.layers.StringLookup(mask_token=None)
movie_titles_vocabulary.adapt(movies)
```

```
Downloading and preparing dataset 4.70 MiB (download: 4.70 MiB, generated: 150.35 KiB, total: 4.84 MiB) to /root/tensorflow_datasets/movielens/100k-movies/0.1.1.
Dl Completed...: 100% 1/1 [00:01<00:00, 1.80s/ url]
Dl Size...: 100% 4/4 [00:01<00:00, 2.27 MiB/s]
Extraction completed...: 100% 23/23 [00:01<00:00, 1.45s/ file]
Dataset movielens downloaded and prepared to /root/tensorflow_datasets/movielens/100k-movies/0.1.1. Subsequent calls will reuse this data.
```

```
# Embedding function for the query.
query_embeddings = similarity_model(query_movie)

# Calculate similarity scores using cosine similarity.
similarity_scores = tf.linalg.matmul(query_embeddings, tf.transpose(similarity_model.movie_model.weights[1]))

# Get indices of most similar movies.
top_k = tf.math.top_k(similarity_scores, k=5)

# Convert movies dataset to a list of movie titles.
movie_titles_list = list(movies.as_numpy_iterator())
top_movie_indices = top_k.indices.numpy()[0]
# Extract movie titles from the list using indices.
top_movie_titles = [movie_titles_list[idx] for idx in top_movie_indices]

print(f"Top 5 similar movies to '{query_movie[0]}':")
for i, title in enumerate(top_movie_titles):
    print(f"{i+1}: {title}")
```

```
Top 5 similar movies to 'From Dusk Till Dawn (1996)':
1: b'Blown Away (1994)'
2: b"Some Mother's Son (1996)"
3: b'Toy Story (1995)'
4: b'Deer Hunter, The (1978)'
5: b'Screamers (1995)'
```

Recommendation System 2: Data Exploration and Visualization

```
[ ]
    !pip install -q matplotlib pandas tensorflow-datasets

    import matplotlib.pyplot as plt
    import pandas as pd
    import tensorflow_datasets as tfds

    # Load the movie metadata
    movies_data = tfds.load('movielens/100k-movies', split="train")
    ratings_data = tfds.load('movielens/100k-ratings', split="train")
```

Converting into tensorflow dataframe

```
[ ]  movies_df = tfds.as_dataframe(movies_data)
     ratings_df = tfds.as_dataframe(ratings_data)

     # Inspect the column names
     print(movies_df.columns)
     print(ratings_df.columns)
```

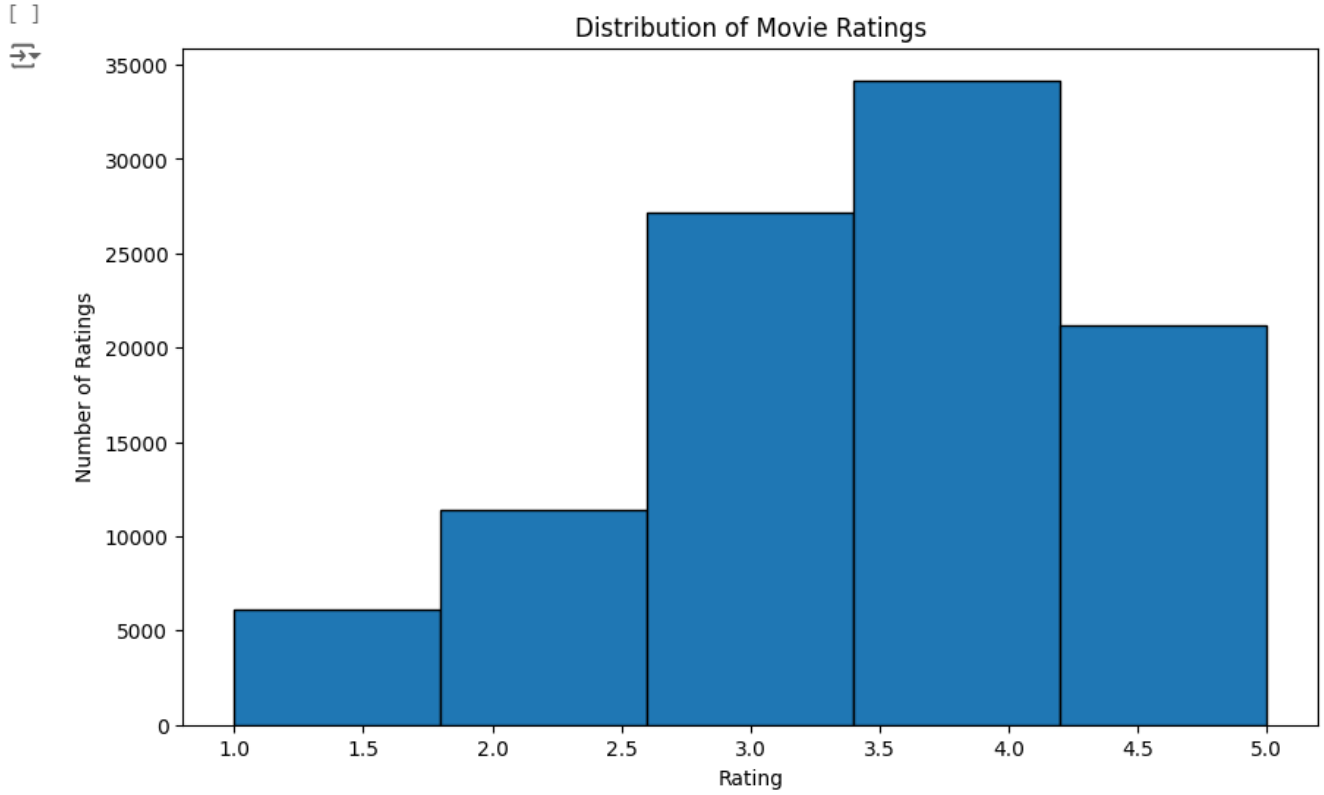
```
Index(['movie_genres', 'movie_id', 'movie_title'], dtype='object')
Index(['bucketized_user_age', 'movie_genres', 'movie_id', 'movie_title',
       'raw_user_age', 'timestamp', 'user_gender', 'user_id',
       'user_occupation_label', 'user_occupation_text', 'user_rating',
       'user_zip_code'],
      dtype='object')
```

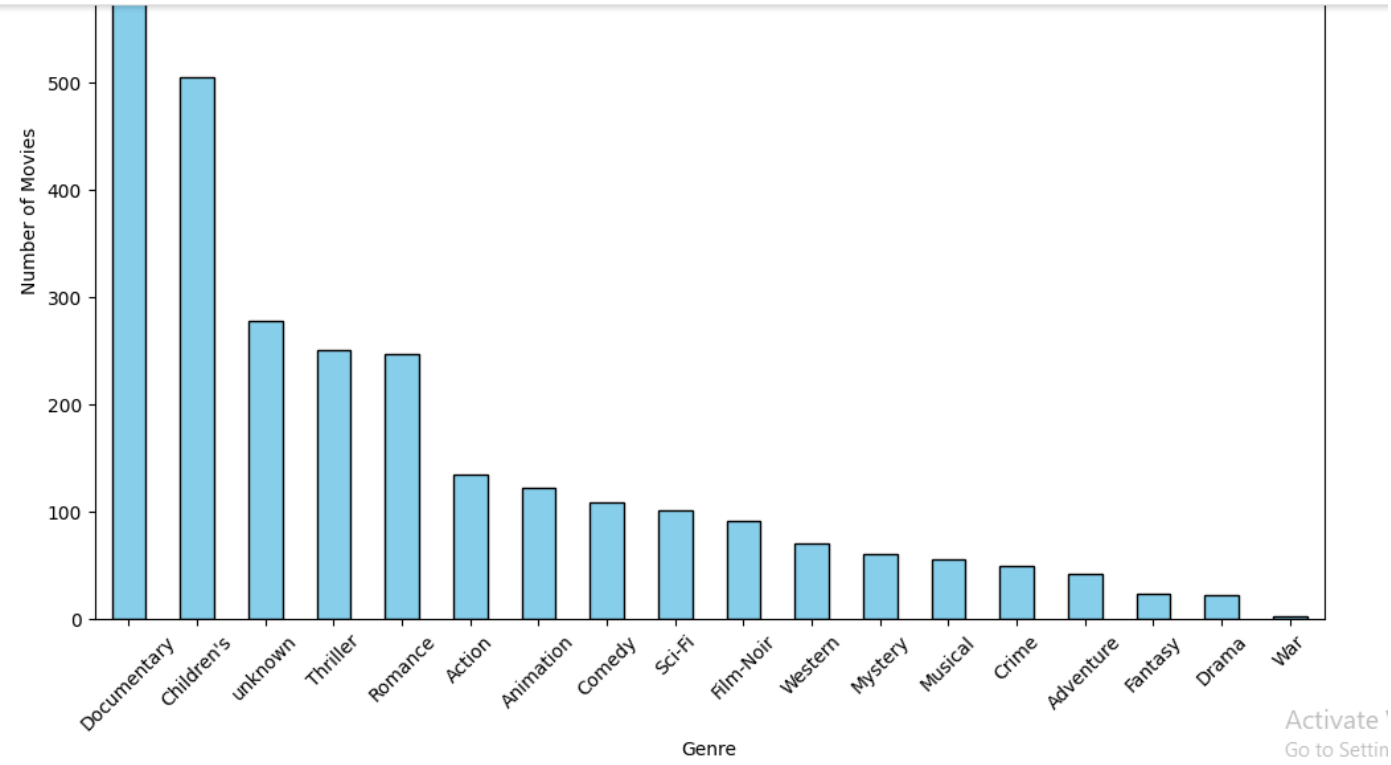
Mapping

```
[ ]
    # Decode bytes to string where applicable
    movies_df['movie_title'] = movies_df['movie_title'].apply(lambda x: x.decode('utf-8') if isinstance(x, bytes) else x)

    # Check the data type of 'movie_genres' column
    print(movies_df['movie_genres'].head())

    # If 'movie_genres' is already in a usable format, no decoding is needed
    # Otherwise, apply decoding if necessary
    # Assuming 'movie_genres' is a list of integers representing genre IDs, we can map them to genre names
    genre_map = {
        0: 'unknown', 1: 'Action', 2: 'Adventure', 3: 'Animation', 4: "Children's",
        5: 'Comedy', 6: 'Crime', 7: 'Documentary', 8: 'Drama', 9: 'Fantasy',
        10: 'Film-Noir', 11: 'Horror', 12: 'Musical', 13: 'Mystery', 14: 'Romance',
        15: 'Sci-Fi', 16: 'Thriller', 17: 'War', 18: 'Western'
    }
```

```
0      [4]
1    [4, 7]
2    [1, 3]
3       [0]
4       [7]
Name: movie_genres, dtype: object
```

[ ]





Recommendation System 3: Average Movie Ratings Visualization

Plot distribution

[9]

```
# Plot line graph of average ratings over movie IDs
plt.figure(figsize=(12, 6))
plt.plot(top_rated_movies['movie_id'], top_rated_movies['user_rating'], marker='o', linestyle='-', color='b')
plt.title('Average Movie Ratings (Top 20 Highest Rated)')
plt.xlabel('Movie ID')
plt.ylabel('Average Rating')
plt.grid(True)
plt.tight_layout()
plt.show()
```

[9]