Exercise 9.2 - Recommender System

August 6, 2023

[3]:
```python
# Import libraries
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
import warnings
warnings.filterwarnings('ignore')
from fuzzywuzzy import process
```

[4]:
```python
# Load ratings and movies excel's into Dataframe
ratings = pd.read_csv('ratings.csv')
movies = pd.read_csv('movies.csv')
```

[5]:
```python
# Merge ratings and movie data into one dataframe
df = ratings.merge(movies, on='movieId')
```

[6]:
```python
# Display the new dataframe
df.head()
```

[6]:
```
   userId  movieId  rating   timestamp             title  \
0       1        1     4.0   964982703  Toy Story (1995)
1       5        1     4.0   847434962  Toy Story (1995)
2       7        1     4.5  1106635946  Toy Story (1995)
3      15        1     2.5  1510577970  Toy Story (1995)
4      17        1     4.5  1305696483  Toy Story (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1  Adventure|Animation|Children|Comedy|Fantasy
2  Adventure|Animation|Children|Comedy|Fantasy
3  Adventure|Animation|Children|Comedy|Fantasy
4  Adventure|Animation|Children|Comedy|Fantasy
```

[7]:
```python
# Create a pivot table with movie titles as index and user ids as columns
pivot_table = df.pivot_table(index='title', columns='userId', values='rating')
```

[8]:
```python
pivot_table.head()
```

[8]:
```
userId                                    1    2    3    4    5    6    7   \
title
'71 (2014)                              NaN  NaN  NaN  NaN  NaN  NaN  NaN
'Hellboy': The Seeds of Creation (2004) NaN  NaN  NaN  NaN  NaN  NaN  NaN
'Round Midnight (1986)                  NaN  NaN  NaN  NaN  NaN  NaN  NaN
'Salem's Lot (2004)                     NaN  NaN  NaN  NaN  NaN  NaN  NaN
'Til There Was You (1997)               NaN  NaN  NaN  NaN  NaN  NaN  NaN

userId                                    8    9    10   …  601  602  603  \
title                                                    …
'71 (2014)                              NaN  NaN  NaN  …  NaN  NaN  NaN
'Hellboy': The Seeds of Creation (2004) NaN  NaN  NaN  …  NaN  NaN  NaN
'Round Midnight (1986)                  NaN  NaN  NaN  …  NaN  NaN  NaN
'Salem's Lot (2004)                     NaN  NaN  NaN  …  NaN  NaN  NaN
'Til There Was You (1997)               NaN  NaN  NaN  …  NaN  NaN  NaN

userId                                  604  605  606  607  608  609  610
title
'71 (2014)                              NaN  NaN  NaN  NaN  NaN  NaN  4.0
'Hellboy': The Seeds of Creation (2004) NaN  NaN  NaN  NaN  NaN  NaN  NaN
'Round Midnight (1986)                  NaN  NaN  NaN  NaN  NaN  NaN  NaN
'Salem's Lot (2004)                     NaN  NaN  NaN  NaN  NaN  NaN  NaN
'Til There Was You (1997)               NaN  NaN  NaN  NaN  NaN  NaN  NaN

[5 rows x 610 columns]
```

[9]:
```
# Fill NaN values with 0
pivot_table = pivot_table.fillna(0)
```

[10]:
```
# Create TfidfVectorizer object
tfidf = TfidfVectorizer(stop_words='english')

# Generate matrix of TF-IDF features
tfidf_matrix = tfidf.fit_transform(pivot_table.index)

# Generate cosine similarity matrix
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

[11]:
```
# List of all movie titles
all_titles = list(pivot_table.index)
```

[12]:
```
# Function to get movie match
# Using fuzzy matching to find the closest match, so the user doesn't have to␣
↪enter the exact title

def get_movie_match(user_input):
```

```python
    # Get match with fuzzy matching
    match = process.extractOne(user_input, all_titles)[0]

    return match
```

[13]:
```python
# Helper function to get movie recommendations
def get_recommendations(title, top_n):

    # Get user input
    user_input = input("What movie do you like? ")

    # Get closest match to input
    title = get_movie_match(user_input)

    # Make sure input is valid
    while title not in all_titles:
        print("Movie not found. Please enter a valid movie title.")
        title = input("What movie do you like? ")

    # Get index of movie title
    idx = pivot_table.index.get_loc(title)

    # Get pairwise similarity scores
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort movies based on similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get top n most similar movies
    sim_scores = sim_scores[1:top_n+1]

    # Get movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return top n movie recommendations
    return pivot_table.index[movie_indices]
```

[14]:
```python
# Generate Recommendation based on User Input
movie = get_recommendations('title', 10)

print("Recommendations: ", movie)
```

```
What movie do you like? Toy Story
Recommendations:  Index(['Toy Story 2 (1999)', 'Toy Story 3 (2010)', 'Toy, The
(1982)',
       'Toy Soldiers (1991)', 'Now and Then (1995)', 'Two Much (1995)',
       'Story of Us, The (1999)', 'L.A. Story (1991)',
```

```
       'Pyromaniac's Love Story, A (1995)', 'Kid's Story (2003)'],
      dtype='object', name='title')
```

Here is a summary of the key steps in the full movie recommender code:

1. Load and merge the MovieLens ratings and movies datasets
2. Pivot the dataframe to have movies as rows and users as columns
3. Create a list of all movie titles for matching
4. Vectorize the movie titles using TF-IDF
5. Calculate a cosine similarity matrix between the TF-IDF vectors
6. Define a function to get fuzzy match for user input
7. Define a function to generate recommendations:

- Take in user input and match to a movie
- Get the index of the matched movie
- Find similar movies based on cosine similarity
- Return top N similar movie titles

8. Call recommendation function with user input
9. Print the recommended movies

In summary, the key steps are:

- Preprocessing data into a movie vs user matrix
- Vectorizing movie titles for similarity
- Fuzzy matching user input
- Generating recommendations via cosine similarity

The program allows the user to simply input a movie name, matches it to a valid title, and outputs personalized movie recommendations.