ACasaccioDSC630 - 10.2 Recommender System

May 16, 2024

Author: Alysen Casaccio

DSC630 - Predictive Analytics

Assignment 10.2: Recommender Systems

Due Date: 5-19-24

Narrative Summary I developed a movie recommender system for this assignment using the small MovieLens dataset and Python for data processing and model development. This appears to be a common data science project with a few different viable approaches. The code I wrote allows users to input a film title (which includes the year of production) they enjoy, and the user receives recommendations for ten similar films.

First, I loaded and cleaned the data, converting timestamp columns to datetime format in case that became leverageable. I checked for missing values and removed any possible duplicates. I then created a user-item matrix, where rows represented users, columns represented movies, and the values were user ratings.

Next, I computed a movie similarity matrix using cosine similarity to measure how similar each pair of movies was based on those user ratings. This matrix formed the core of the recommendation engine.

To recommend movies, I designed a function that takes a movie title as input, finds the movie ID, and retrieves similarity scores for all other movies. By sorting these scores, the function identifies the top ten most similar movies, which are then recommended to the user.

The recommender system provides a straightforward yet effective approach to movie recommendations, leveraging collaborative filtering to suggest movies based on user preferences. This project demonstrates the practical application of data cleaning, matrix manipulation, and similarity calculations in building a recommendation engine.

[1]:
```
# import statements
import pandas as pd
import warnings
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
```

[2]:
```
# suppressing future warnings for this assignment
warnings.filterwarnings('ignore')
```

[3]:
```python
# loading and reading
movies_file_path = 'C:/Users/alyse/OneDrive/Documents/Bellevue University/DSC␣
↪630 - Predictive Analytics/Week 10 Movie Data/movies.csv'
links_file_path = 'C:/Users/alyse/OneDrive/Documents/Bellevue University/DSC␣
↪630 - Predictive Analytics/Week 10 Movie Data/links.csv'
ratings_file_path = 'C:/Users/alyse/OneDrive/Documents/Bellevue University/DSC␣
↪630 - Predictive Analytics/Week 10 Movie Data/ratings.csv'
tags_file_path = 'C:/Users/alyse/OneDrive/Documents/Bellevue University/DSC 630␣
↪- Predictive Analytics/Week 10 Movie Data/tags.csv'
movies = pd.read_csv(movies_file_path)
links = pd.read_csv(links_file_path)
ratings = pd.read_csv(ratings_file_path)
tags = pd.read_csv(tags_file_path)

movies.info()
links.info()
ratings.info()
tags.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9742 entries, 0 to 9741
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   movieId  9742 non-null   int64
 1   title    9742 non-null   object
 2   genres   9742 non-null   object
dtypes: int64(1), object(2)
memory usage: 228.5+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9742 entries, 0 to 9741
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   movieId  9742 non-null   int64
 1   imdbId   9742 non-null   int64
 2   tmdbId   9734 non-null   float64
dtypes: float64(1), int64(2)
memory usage: 228.5 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100836 entries, 0 to 100835
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   userId     100836 non-null  int64
 1   movieId    100836 non-null  int64
 2   rating     100836 non-null  float64
```

```
 3   timestamp  100836 non-null  int64
dtypes: float64(1), int64(3)
memory usage: 3.1 MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3683 entries, 0 to 3682
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   userId     3683 non-null   int64
 1   movieId    3683 non-null   int64
 2   tag        3683 non-null   object
 3   timestamp  3683 non-null   int64
dtypes: int64(3), object(1)
memory usage: 115.2+ KB
```

Data Cleaning

[4]:
```
# converting timestamp columns to datetime
ratings['timestamp'] = pd.to_datetime(ratings['timestamp'], unit='s')
tags['timestamp'] = pd.to_datetime(tags['timestamp'], unit='s')

# checking for missing values
print(movies.isnull().sum())
print(links.isnull().sum())
print(ratings.isnull().sum())
print(tags.isnull().sum())
```

```
movieId    0
title      0
genres     0
dtype: int64
movieId    0
imdbId     0
tmdbId     8
dtype: int64
userId       0
movieId      0
rating       0
timestamp    0
dtype: int64
userId       0
movieId      0
tag          0
timestamp    0
dtype: int64
```

[5]:
```
# checking for duplicates and remove them
movies.drop_duplicates(inplace=True)
links.drop_duplicates(inplace=True)
```

```python
ratings.drop_duplicates(inplace=True)
tags.drop_duplicates(inplace=True)
```

[6]:
```python
# verifying data types
print(movies.dtypes)
print(links.dtypes)
print(ratings.dtypes)
print(tags.dtypes)
```

```
movieId     int64
title      object
genres     object
dtype: object
movieId      int64
imdbId       int64
tmdbId     float64
dtype: object
userId                int64
movieId               int64
rating              float64
timestamp    datetime64[ns]
dtype: object
userId                int64
movieId               int64
tag                  object
timestamp    datetime64[ns]
dtype: object
```

0.0.1 Creating a Similarity Matrix

[7]:
```python
# creating a user-item matrix
user_movie_ratings = ratings.pivot(index='userId', columns='movieId',␣
↪values='rating').fillna(0)

# computing a similarity matrix
movie_similarity = cosine_similarity(user_movie_ratings.T)

# converting the similarity matrix to a DataFrame
movie_similarity_df = pd.DataFrame(movie_similarity, index=user_movie_ratings.
↪columns, columns=user_movie_ratings.columns)
```

Establishing the Recommender Function

[8]:
```python
# creating a function to recommend movies
def recommend_movies(movie_title, n_recommendations=10):
    # finding the movie ID for the input movie title
    movie_id = movies[movies['title'] == movie_title]['movieId'].values[0]
```

```
    # getting the similarity scores for the input movie
    similarity_scores = movie_similarity_df[movie_id]

    # sorting the movies based on similarity scores
    similar_movies = similarity_scores.sort_values(ascending=False)

    # getting the top N similar movies
    top_n_movies = similar_movies.index[1:n_recommendations + 1]

    # getting the movie titles for the top N similar movies
    recommended_movie_titles = movies[movies['movieId'].
↪isin(top_n_movies)]['title']

    return recommended_movie_titles
```

Code for Practice Usage and Download

[10]:
```
# providing an example of usage
input_movie = "Wicker Man, The (1973)" ## change the movie title as desired
recommended_movies = recommend_movies(input_movie, 10)
print("Movies similar to", input_movie, ":\n", recommended_movies)
```

```
Movies similar to Wicker Man, The (1973) :
 3755         Return of the Secaucus 7 (1980)
3843                   Monkey Business (1952)
4635    Unvanquished, The (Aparajito) (1957)
4673                 Ordet (Word, The) (1955)
4729              What's New, Pussycat (1965)
4779     Diabolique (Les diaboliques) (1955)
4970                 Fountainhead, The (1949)
5427       Day of Wrath (Vredens dag) (1943)
5462               Au Hasard Balthazar (1966)
6510                       TV Set, The (2006)
Name: title, dtype: object
```

[ ]:
```
# option to save cleaned data and model to csv
movies.to_csv('cleaned_movies.csv', index=False)
links.to_csv('cleaned_links.csv', index=False)
ratings.to_csv('cleaned_ratings.csv', index=False)
tags.to_csv('cleaned_tags.csv', index=False)

# option to save movie similarity matrix
movie_similarity_df.to_csv('movie_similarity.csv')
```

References Sucky, R. N. (2020, October 23). A complete recommendation system algorithm using Python's scikit-learn library: Step by step guide. Towards Data Science. Retrieved from https://towardsdatascience.com/a-complete-recommendation-system-algorithm-using-pythons-scikit-learn-library-step-by-step-guide-9d563c4db6b2

Adhikari, S. (2019, February 27). Building a movie recommendation engine in Python using scikit-learn. Medium. Retrieved from https://medium.com/@sumanadhikari/building-a-movie-recommendation-engine-using-scikit-learn-8dbb11c5aa4b

Kurka, B. (2019, May 7). Building a movie recommender system with Python. Medium. Retrieved from https://medium.com/@bkexcel2014/building-movie-recommender-systems-using-cosine-similarity-in-python-eff2d4e60d24

Python Software Foundation. (2023). Python Language Reference, version 3.10. Available at https://www.python.org/

[ ]:
```

```