Untitled9

August 4, 2024

DSC 630

Week 8 Assignment

Said Moussadeq

Movies Recommender System

This project involves creating a movie recommender system using the MovieLens dataset. The system allows users to input a movie they like and recommends ten other movies for them to watch. The recommender system is built using collaborative filtering with Singular Value Decomposition (SVD), a matrix factorization technique. The code is divided into multiple chunks for better readability and modularity.

Import Libraries and Set Working Directory

[18]:
```
import os
import pandas as pd
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate
import warnings
warnings.filterwarnings('ignore')
```

Load and Merge Data

[19]:
```
# Step 2: Load the Data
df_movies = pd.read_csv('movies.csv')
df_ratings = pd.read_csv('ratings.csv')

# Merge the dataframes
df = pd.merge(df_ratings, df_movies, on='movieId')
df.head()
```

[19]:

	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
```

Prepare Data for Surprise Library

[20]:
```
# Step 3: Prepare Data for Surprise Library
reader = Reader(rating_scale=(0.5, 5.0))
data = Dataset.load_from_df(df[['userId', 'movieId', 'rating']], reader)
```

Train the SVD Model

[21]:
```
# Step 4: Train the SVD Model
svd = SVD()
cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
trainset = data.build_full_trainset()
svd.fit(trainset)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8784  0.8672  0.8731  0.8744  0.8740  0.8734  0.0036
MAE (testset)     0.6714  0.6657  0.6690  0.6744  0.6731  0.6707  0.0031
Fit time          0.96    1.11    0.95    0.96    0.97    0.99    0.06
Test time         0.09    0.09    0.16    0.08    0.08    0.10    0.03
```

[21]:
```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x2a8ad715150>
```

Display Sample Movies

[22]:
```
# Display few movies from the database
print("Here are some movies available you can pick from:")
print(df_movies['title'].sample(10).to_string(index=False))
print("\n")
```

```
Here are some movies available you can pick from:
                Green Mile, The (1999)
          Welcome to Mooseport (2004)
Captain Horatio Hornblower R.N. (1951)
              Human Stain, The (2003)
          Doom Generation, The (1995)
                        Neds (2010)
            Closer to the Moon (2013)
```

```
Rapture-Palooza (2013)
     Moonwalker (1988)
        Gintama (2017)
```

Define Helper Functions

[23]:
```python
# Step 5: Build Recommender Function
def get_movie_id(title):
    # Perform a case-insensitive match and allow partial matches
    result = df_movies[df_movies['title'].str.contains(title, case=False,␣
 ↪na=False, regex=False)]
    if len(result) > 0:
        print(f"Potential matches for '{title}':")
        print(result['title'].to_string(index=False))
        return result.iloc[0]['movieId']
    else:
        return None

def get_movie_title(movie_id):
    return df_movies[df_movies['movieId'] == movie_id]['title'].values[0]

def recommend_movies(movie_title, n_recommendations=10):
    movie_id = get_movie_id(movie_title)
    if not movie_id:
        print("Sorry, that movie is not in our database.")
        return []

    user_id = df['userId'].sample(1).iloc[0]  # Use a random user ID

    # Get the list of movie IDs that the user has not rated yet
    movie_ids = df_movies['movieId'].unique()
    watched_movie_ids = df[df['userId'] == user_id]['movieId'].values
    movie_ids_to_predict = [movie_id for movie_id in movie_ids if movie_id not␣
 ↪in watched_movie_ids]

    # Predict ratings for the movies the user has not seen
    predictions = [svd.predict(user_id, movie_id) for movie_id in␣
 ↪movie_ids_to_predict]

    # Sort predictions by estimated rating
    predictions.sort(key=lambda x: x.est, reverse=True)

    # Get the top n recommendations
    top_predictions = predictions[:n_recommendations]
```

```
    recommended_movie_titles = [get_movie_title(pred.iid) for pred in␣
↪top_predictions]

    return recommended_movie_titles
```

Get Recommendations

[24]:
```
# Step 6: Get Recommendations
movie_title = input("Enter a movie title: ")
recommendations = recommend_movies(movie_title)

if recommendations:
    print("\nRecommended movies based on your choice:")
    for i, title in enumerate(recommendations, 1):
        print(f"{i}. {title}")
```

```
Enter a movie title:  No Strings Attached (2011)

Potential matches for 'No Strings Attached (2011)':
No Strings Attached (2011)

Recommended movies based on your choice:
1. Inglourious Basterds (2009)
2. Guess Who's Coming to Dinner (1967)
3. Sweet Hereafter, The (1997)
4. Lives of Others, The (Das leben der Anderen) (2006)
5. Casablanca (1942)
6. Rear Window (1954)
7. Glory (1989)
8. To Catch a Thief (1955)
9. Monty Python's And Now for Something Completely Different (1971)
10. North by Northwest (1959)
```

1 References

- Hug, N. (2020). Surprise: A Python library for recommender systems. Journal of Open Source Software, 5(52), 2174. https://doi.org/10.21105/joss.02174
- Surprise Library Documentation: http://surpriselib.com/