Ryan Frank

DSC630

05/18/2025

Week 10 Assignment: Recommender System

The goal is to create a recommender system based on the small MovieLens data set, where a movie will be provided to the recommender and based on that it will recommend 10 other movies to watch. As we do not have a user profile to work with (the only information we have on the user is a single selected movie) I will be using item-to-item Collaborative Filtering to build my recommender system. I want to leverage both the movie genres and the user ratings to make the recommendations.

Resources used:
https://www.geeksforgeeks.org/item-to-item-based-collaborative-filtering/
https://analyticsindiamag.com/deep-tech/how-to-build-your-first-recommender-system-using-python-movielens-dataset/
https://towardsdatascience.com/using-cosine-similarity-to-build-a-movie-recommendation-system-ae7f20842599/

In [...

```
# Import modules used in code
import pandas
from sklearn.metrics.pairwise import cosine_similarity
```

In [...
```
# load movie data
movieData = pandas.read_csv('movies.csv')
movieData.head()
```

Out[...

	movieId	title	ger
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fan
1	2	Jumanji (1995)	Adventure|Children|Fan
2	3	Grumpier Old Men (1995)	Comedy|Roma
3	4	Waiting to Exhale (1995)	Comedy|Drama|Roma
4	5	Father of the Bride Part II (1995)	Com

In [...
```
# get distinct list of genres from the loaded data
distinctGenres = []
for i, row in movieData.iterrows():
    # do nothing if no genre data
    if row['genres'] == '(no genres listed)':
        continue
    # split genres by | delimited, and if not already
    genreList = str(row['genres']).split("|")
    for item in genreList:
        if item not in distinctGenres:
            distinctGenres.append(item)
```

In […

```
# turn genres into a 0 or 1 flag and add as columns to
def checkGenre(movieGenres, genre):
    if genre in movieGenres:
        return 1
    return 0

# for each distinct genre discovered create a column a
for genre in distinctGenres:
    movieData[genre] = movieData['genres'].apply(lambd
```

In […

```
# create a new column that is the count of the number
# Need this to test for movies with no genre data in t
movieData['numGenres'] = movieData[distinctGenres].sum
```

In […

```
# check resulting dataframe
movieData.head(10)
```

Out[...

	movieId	title	ge
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fa
1	2	Jumanji (1995)	Adventure|Children|Fa
2	3	Grumpier Old Men (1995)	Comedy|Rom
3	4	Waiting to Exhale (1995)	Comedy|Drama|Rom
4	5	Father of the Bride Part II (1995)	Cor
5	6	Heat (1995)	Action|Crime|Th
6	7	Sabrina (1995)	Comedy|Rom
7	8	Tom and Huck (1995)	Adventure|Chi
8	9	Sudden Death (1995)	A
9	10	GoldenEye (1995)	Action|Adventure|Th

10 rows × 23 columns

In [...
```
# create a matrix of genres
genreMatrix = movieData[['movieId', 'Adventure', 'Anim
                         'Horror', 'Mystery', 'Sci-Fi'
movieIDlist = genreMatrix['movieId'].values.tolist()
genreMatrix.set_index('movieId', inplace=True)
```

In [...
```
# create a matrix of similiarity of genres for movies
# by sorting on a movie ID we can get a list of movies
genreSimilarityMatrix = pandas.DataFrame(cosine_simila
genreSimilarityMatrix.head(10)
```

Out[...

	1	2	3	4	5	6
1	1.000000	0.774597	0.316228	0.258199	0.447214	0.000000
2	0.774597	1.000000	0.000000	0.000000	0.000000	0.000000
3	0.316228	0.000000	1.000000	0.816497	0.707107	0.000000
4	0.258199	0.000000	0.816497	1.000000	0.577350	0.000000
5	0.447214	0.000000	0.707107	0.577350	1.000000	0.000000
6	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
7	0.316228	0.000000	1.000000	0.816497	0.707107	0.000000
8	0.632456	0.816497	0.000000	0.000000	0.000000	0.000000
9	0.000000	0.000000	0.000000	0.000000	0.000000	0.577350
10	0.258199	0.333333	0.000000	0.000000	0.000000	0.666667

10 rows × 9742 columns

In [...
```
# test pull - sort movies based on similarity to movie
genreSimilarityMatrix.sort_values(by=2,ascending=False
```

Out[…

	1	2	3	4	5	6	7	8	9	
50601	0.774597	1.0	0.0	0.0	0.0	0.0	0.0	0.816497	0.0	0.3333
2043	0.774597	1.0	0.0	0.0	0.0	0.0	0.0	0.816497	0.0	0.3333
59501	0.774597	1.0	0.0	0.0	0.0	0.0	0.0	0.816497	0.0	0.3333
104074	0.774597	1.0	0.0	0.0	0.0	0.0	0.0	0.816497	0.0	0.3333
173873	0.774597	1.0	0.0	0.0	0.0	0.0	0.0	0.816497	0.0	0.3333
56915	0.774597	1.0	0.0	0.0	0.0	0.0	0.0	0.816497	0.0	0.3333
1009	0.774597	1.0	0.0	0.0	0.0	0.0	0.0	0.816497	0.0	0.3333
160573	0.774597	1.0	0.0	0.0	0.0	0.0	0.0	0.816497	0.0	0.3333
56171	0.774597	1.0	0.0	0.0	0.0	0.0	0.0	0.816497	0.0	0.3333
4896	0.774597	1.0	0.0	0.0	0.0	0.0	0.0	0.816497	0.0	0.3333

10 rows × 9742 columns

In […

```
# build a lookup dictionary to get movieId from movie
movieLookup = {}
for i, row in movieData.iterrows():
    movieLookup[row['title']] = {'movieID': row['movie
# test lookup
movieLookup['Father of the Bride Part II (1995)']
# also build lookup for movieId to title
titleLookup = {}
for i, row in movieData.iterrows():
    titleLookup[row['movieId']] = row['title']
```

In […

```
# load movie data
ratingData = pandas.read_csv('ratings.csv')
ratingData.head(5)
```

Out[...

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

In [...

```
# create a matrix of userID x movieID with the value b
ratingMatrix = ratingData.pivot_table(index='userId',c
ratingMatrix.head(10)
```

Out[...

movieId	1	2	3	4	5	6	7	8	9	1
userId										
1	4.0	NaN	4.0	NaN	NaN	4.0	NaN	NaN	NaN	Na
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
5	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
6	NaN	4.0	5.0	3.0	5.0	4.0	4.0	3.0	NaN	3
7	4.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
8	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

10 rows × 9724 columns

In […

```
def makeRecommendations(movieTitle):
    # get the movieID of the movie we are basing recom
    movieID = movieLookup[movieTitle]['movieID']
    # get the top 500 movies with similar genres based
    similarGenres = genreSimilarityMatrix.sort_values(
    # get a list of the movieIDs with similar genres
    similarGenresList = similarGenres.index.values.tol
    # if the original movieID is in the list, remove i
    similarGenresList.remove(movieID)
    # I'm using corrwith instead of cosine similarity
    # this creates correlation between movies based on
    correlations = ratingMatrix.corrwith(ratingMatrix[
    # turn correlations into a dataframe
    recommendations = pandas.DataFrame(correlations,co
    # remove entries with NaN correlation value
    recommendations.dropna(inplace=True)
    if movieLookup[movieTitle]['numGenres'] != 0:
        # only do this if the number of genres in the
        # otherwise we would prioritizing other movies
        finalRecommendation = recommendations[recommen
    else:
        # when there is no genre data, make recommenda
        finalRecommendation = recommendations.sort_val
    # report recommendations
    print(f"Based on your interest in {movieTitle} we
    i = 1
    for index, row in finalRecommendation.itertuples()
        print(f"{i}) {titleLookup[index]}")
        i += 1
```

In […

```
makeRecommendations("Grumpier Old Men (1995)")
```

```
C:\Users\Ryan\AppData\Roaming\Python\Python311\site-packages\numpy\lib\function_base.py:2889: RuntimeWarning: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
C:\Users\Ryan\AppData\Roaming\Python\Python311\site-packages\numpy\lib\function_base.py:2748: RuntimeWarning: divide by zero encountered in divide
  c *= np.true_divide(1, fact)
C:\Users\Ryan\AppData\Roaming\Python\Python311\site-packages\numpy\lib\function_base.py:2748: RuntimeWarning: invalid value encountered in multiply
  c *= np.true_divide(1, fact)
C:\Users\Ryan\AppData\Roaming\Python\Python311\site-packages\numpy\lib\function_base.py:2897: RuntimeWarning: invalid value encountered in divide
  c /= stddev[:, None]
C:\Users\Ryan\AppData\Roaming\Python\Python311\site-packages\numpy\lib\function_base.py:2898: RuntimeWarning: invalid value encountered in divide
  c /= stddev[None, :]
```

```
Based on your interest in Grumpier Old Men (1995) we recommend:
1) Blame It on Rio (1984)
2) Down with Love (2003)
3) World According to Garp, The (1982)
4) Lost & Found (1999)
5) The Big Sick (2017)
6) Booty Call (1997)
7) Sweetest Thing, The (2002)
8) Love Potion #9 (1992)
9) Mr. Baseball (1992)
10) Heartbreakers (2001)
```

In […

```
makeRecommendations("Toy Story (1995)")
```

```
C:\Users\Ryan\AppData\Roaming\Python\Python311\site-packages\numpy\lib\function_base.py:2889: RuntimeWarning: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
C:\Users\Ryan\AppData\Roaming\Python\Python311\site-packages\numpy\lib\function_base.py:2748: RuntimeWarning: divide by zero encountered in divide
  c *= np.true_divide(1, fact)
C:\Users\Ryan\AppData\Roaming\Python\Python311\site-packages\numpy\lib\function_base.py:2748: RuntimeWarning: invalid value encountered in multiply
  c *= np.true_divide(1, fact)
C:\Users\Ryan\AppData\Roaming\Python\Python311\site-packages\numpy\lib\function_base.py:2897: RuntimeWarning: invalid value encountered in divide
  c /= stddev[:, None]
C:\Users\Ryan\AppData\Roaming\Python\Python311\site-packages\numpy\lib\function_base.py:2898: RuntimeWarning: invalid value encountered in divide
  c /= stddev[None, :]
```

```
Based on your interest in Toy Story (1995) we recommend:
1) Mind Game (2004)
2) Great Yokai War, The (Yôkai daisensô) (2005)
3) Land Before Time III: The Time of the Great Giving (1995)
4) Wizard, The (1989)
5) Rio 2 (2014)
6) It's a Very Merry Muppet Christmas Movie (2002)
7) Zathura (2005)
8) For the Birds (2000)
9) Ewok Adventure, The (a.k.a. Caravan of Courage: An Ewok Adventure) (1984)
10) Planes: Fire & Rescue (2014)
```

My final version of the recommender uses the similiarity of genres to filter movies before comparing the user rating profiles. It seems to produce recommendations that make sense, although due to a lack of incorperating a minimum number of reviews it tends to return some obscure titles along with some more obvious ones. If I were to continue to refine this I would probably also add a filter for movies to have a minimum number of reviews to make it on the recommendation list as well.