HW 7 - Recommendation Systems

Swathi Venkatesh

11/29/2021

Q1

Find 3 users who are closest to you in terms of age, gender, and occupation. For each of those 3 users:

- *Q. What are their top 3 (favorite) films?*
- *Q. What are their bottom 3 (least favorite) films?*

Based on the movie values in those 6 tables (3 users X (favorite + least favorite)), choose a user that you feel is most like you. Feel free to note any outliers (e.g., "I mostly identify with user 123, except I did not like "Ghost" at all"). You can investigate more than just the top 3 and bottom 3 movies to find your best match.

This user is the substitute you.

Answer

```
#!/usr/local/bin/python3
import pandas as pd

def funcAgeGendOccupation(item,data):

    #load movies data into variables
    movies = {}
    for line in open(str(item)):
        (id, title) = line.split('|')[0:2]
        movies[id] = title
    # Load data
    prefs = {}
    for line in open(str(data)):
        (user, movieid, rating, ts) = line.split('\t')
        prefs.setdefault(user, {})
        prefs[user][movies[movieid]] = float(rating)
    #Load in users
    users = []
    for line in open('movie/u.user'):
        (user, age, gender, occupation, zipcode) = line.split('|')
```

```
21          dictA = {'user': user, 'age':age, 'gender': gender, 'occupation
    ' :occupation }
22          users.append(dictA)
23
24      count = 1
25      #hold 3 similar user in term of age gender and category
26      similarUser =[]
27      #Find all users that have things common with me
28      for a in users:
29          """
30          passed in my age gender and occupation
31          to find users of my age gender and occupation category.
32          I take only 3 users
33          """
34          if a['age'] == '23' and a['gender'] == 'F' and a['occupation']
    == 'student' and count < 4 :
35              similarUser.append(a['user'])
36              count += 1
37      #print user movie likes
38      for a in similarUser:
39          """
40          Based on similar users like me,
41          I obtained these similar users top 3 favorite films
42          """
43          if a in prefs:
44              #this gets the movie names, and the movie rating per user
45              p = pd.DataFrame(list(prefs[a].items()),columns=["Movie
    Names","Rating"])
46              #this sort the users movie list based on rating in
    decending order
47              p.sort_values("Rating",ascending=False, inplace=True)
48              #gets the top3 films
49              top3 = p.head(3)
50              #gets the bottom 3 films
51              bottom3 = p.tail(3)
52              top3.to_csv("one/"+a+"top3.csv",index=False)
53              bottom3.to_csv("one/"+a+"bottom.csv",index=False)
54  funcAgeGendOccupation('movie/u.item','movie/u.data')
```

Listing 1: one.py

Table 1: Top 3 movies for user 49

Movie Names	Rating
Monty Python and the Holy Grail (1974)	5.0
Paradise Lost: The Child Murders at Robin Hood Hills (1996)	5.0
Clockwork Orange, A (1971)	5.0

Table 2: Bottom 3 movies for user 49

Movie Names	Rating
Firm, The (1993)	1.0
Long Kiss Goodnight, The (1996)	1.0
Terminator 2: Judgment Day (1991)	1.0

Table 3: Top 3 movies for user 159

Movie Names	Rating
Grumpier Old Men (1995)	5.0
That Old Feeling (1997)	5.0
Volcano (1997)	5.0

Table 4: Bottom 3 movies for user 159

Movie Names	Rating
Grumpier Old Men (1995)	1.0
Soul Food (1997)	1.0
Ace Ventura: Pet Detective (1994)	1.0

Table 5: Top 3 movies for user 477

Movie Names	Rating
One Fine Day (1996)	5.0
Evita (1996)	5.0
Grease (1978)	5.0

Table 6: Bottom 3 movies for user 477

Movie Names	Rating
Corrina, Corrina (1994)	4.0
Black Sheep (1996)	4.0
Don Juan DeMarco (1995)	4.0

Discussion

- *Q. What are their top 3 (favorite) films?*
- *Q. What are their bottom 3 (least favorite) films?*

The users top 3 and bottom 3 films are shown in the above tables.

I created a function called funcAgeGendOccupation() ,in line 4. This function takes two parameters of the location of u.item and u.data path. u.user was not going through as a parameter so it is passed directly.

From lines 6-22 the data from each file is read and stored into dictionary variables. In line 24 -36, I found users that were just as similar to me in age, gender and occupation. I limited this search to find only just 3 similar users to me on line 30. These users where stored in a list variable called similarUser. The line 26 is the variable declaration, while line 35 appends new users to the list in the created variable.

The lines 38 to 53, process the outcomes for top 3 movies and bottom 3 movies for each users found. Using a for loop to retrieve each user, parse this result to ensure that the said user is in prefs dictionary. If present, retrieve the row of the dictionary of that particular user. Using a pandas dataframe in Line 45, parse a list of the row item, so that we can easily sort the data based on values in Rating column in descending order. Retrieve the first three data using pandas .head(3) and the last three data using .tail(3). Save each result as a pandas dataFrame and convert the result to a saved csv file in one/.

My best substitute user is 159, the movies user hates I hate and the movies the user enjoys I too find them interesting.

Q2

Based on the ratings that users have given to the movies, answer the following questions:

- *Q: Which 5 users are most correlated to the substitute you (i.e., which 5 users rate movies most similarly to the substitute you?)*
- *Q: Which 5 users are least correlated (i.e., negative correlation)?*

Answer

```
1 #!/usr/local/bin/python3
2 from math import sqrt
```

```
3
4  def sim_pearson(prefs, p1, p2):
5      """
6      Returns the Pearson correlation coefficient for p1 and p2.
7      """
8
9  # Get the list of mutually rated items
10     si = {}
11     for item in prefs[p1]:
12         if item in prefs[p2]:
13             si[item] = 1
14
15 # If they are no ratings in common, return 0
16     if len(si) == 0:
17         return 0
18
19 # Sum calculations
20     n = len(si)
21
22  # Sums of all the preferences
23     sum1 = sum([prefs[p1][it] for it in si])
24     sum2 = sum([prefs[p2][it] for it in si])
25
26 # Sums of the squares
27     sum1Sq = sum([pow(prefs[p1][it], 2) for it in si])
28     sum2Sq = sum([pow(prefs[p2][it], 2) for it in si])
29
30  # Sum of the products
31     pSum = sum([prefs[p1][it] * prefs[p2][it] for it in si])
32
33  # Calculate r (Pearson score)
34     num = pSum - sum1 * sum2 / n
35     den = sqrt((sum1Sq - pow(sum1, 2) / n) * (sum2Sq - pow(sum2, 2) / n
36     ))
37     if den == 0:
38         return 0
39     r = num / den
40     return r
41
42 def getRecommendations(prefs, person, similarity=sim_pearson):
43     """
44      Gets recommendations for a person by using a weighted average
45      of every other users rankings
46     """
47     totals = {}
48     simSums = {}
49     for other in prefs:
```

```
        if other == person:
            continue
        sim = similarity(prefs, person, other)
        #Ignore scores of zero or lower
        if sim <= 0:
            continue
        for item in prefs[other]:
            #Only score movies I havent seen yet
            if item not in prefs[person] or prefs[person][item] == 0:
                # Similarity * Score
                totals.setdefault(item, 0)
                # The final score is calculated by multiplying each item by the
                # similarity and adding these products together
                totals[item] += prefs[other][item] * sim
                # Sum of similarities
                simSums.setdefault(item, 0)
                simSums[item] += sim
                # Create the normalized list
        rankings = [(total / simSums[item], item) for (item, total) in
                            totals.items()]
        # Return the sorted list
        rankings.sort()
        rankings.reverse()
        return rankings

def topMatches(prefs,person,n=5,similarity=sim_pearson,):
    '''
    Returns the best matches for person from the prefs dictionary.
    Number of results and similarity function are optional params.
    bottomMatches
    '''

    scores = [(similarity(prefs, person, other), other) for other in prefs
              if other != person]
    scores.sort()
    scores.reverse()
    return scores[0:n]

def bottomMatches(prefs,person,n=5,similarity=sim_pearson,):
    """
    Returns the lowest matches for person from the prefs dictionary.
    Number of results and similarity function are optional params.
    """
    scores = [(similarity(prefs, person, other), other) for other in prefs
```

```
93      if other != person]
94      scores.sort()
95      scores.reverse()
96      return scores[len(scores)-n: len(scores)]
97
98  def transformPrefs(prefs):
99      '''
100     Transform the recommendations into a mapping where persons are
        described with interest scores for a given title e.g. {title: person
        } instead of {person: title}
101     '''
102
103     result = {}
104     for person in prefs:
105         for item in prefs[person]:
106             result.setdefault(item, {})
107             # Flip item and person
108             result[item][person] = prefs[person][item]
109     return result
110
111 def loadMovieLens():
112     # Get movie titles
113     movies = {}
114     for line in open("movie/u.item"):
115         (id, title) = line.split("|")[0:2]
116         movies[id] = title
117     # Load data
118     #prefs = {}
119     for line in open("movie/u.data"):
120         (user, movieid, rating, ts) = line.split("\t")
121         prefs.setdefault(user, {})
122         prefs[user][movies[movieid]] = float(rating)
123     #Load in users
124     #users = []
125     for line in open("movie/u.user"):
126         (user, age, gender, occupation, zipcode) = line.split("|")
127         dictA = {"user": user, "age":age, "gender": gender, "occupation
        " :occupation }
128     users.append(dictA)
129
130 prefs ={}
131 users =[]
132 loadMovieLens()
133
134 top5 = topMatches(prefs, "159", n = 5, similarity = sim_pearson)
135 bottom5 = bottomMatches(prefs, "928", n = 5, similarity = sim_pearson)
136 print("5 users that  most correlate with my substitute me ")
```

```
print(*top5, sep='\n')
print("\n\n")
print("5 users that least correlate with my substitute me ")
print(*bottom5, sep='\n')
"""
question2:
(1.000000000000004, '604')
(1.000000000000001, '713')
(1.0, '914')
(1.0, '80')
(1.0, '237')

(-1.000000000000004, '760')
(-1.000000000000004, '547')
(-1.000000000000004, '432')
(-1.000000000000004, '317')
(-1.000000000000004, '112')
"""
recommend = getRecommendations(prefs,"159")
print("\nTop 5 movies recommendations for substitute me: ")
print(*recommend[0:5], sep="\n")
print("\nBottom 5 movies recommendations for substitute me:")
print(*recommend[len(recommend)-5: len(recommend)], sep="\n")
"""
question3:
(5.0, 'Unforgiven (1992)')
(5.0, 'Unforgettable (1996)')
(5.0, 'Tombstone (1993)')
(5.0, 'Silence of the Lambs, The (1991)')
(5.0, 'Net, The (1995)')

(1.0, 'Usual Suspects, The (1995)')
(1.0, 'Thinner (1996)')
(1.0, 'Natural Born Killers (1994)')
(1.0, 'Full Monty, The (1997)')
(1.0, 'Albino Alligator (1996)')
"""

print("\n\n")
movies = transformPrefs(prefs)
mybestmovie = "Vampire in Brooklyn (1995)"
topFive = topMatches(movies, mybestmovie)
print("My best 5 recommended movies")
print(topFive)
print("\n")
print("My worst 5 recommended movies")
worstFive = bottomMatches(movies,mybestmovie)
```

```
184 print(worstFive)
185 """
186 question4:
187 [(1.000000000000004, 'Young Guns II (1990)'), (1.000000000000004, '
      Reality Bites (1994)'), (1.000000000000001, 'Ran (1985)'),
      (1.000000000000001, 'Butch Cassidy and the Sundance Kid (1969)'),
      (1.0000000000000009, 'Chinatown (1974)')]
188
189 [(-1.0, 'Big Night (1996)'), (-1.0, 'Barbarella (1968)'), (-1.0, '
      Another Stakeout (1993)'), (-1.0000000000000007, "Fathers' Day
      (1997)"), (-1.000000000000004, "Devil's Own, The (1997)")]
190 """
```

Listing 2: two.py

```
 5 users that  most correlate with my substitute me
(1.000000000000004, '604')
(1.000000000000001, '713')
(1.0, '914')
(1.0, '80')
(1.0, '237')


 5 users that least correlate with my substitute me
(-1.000000000000004, '760')
(-1.000000000000004, '547')
(-1.000000000000004, '432')
(-1.000000000000004, '317')
(-1.000000000000004, '112')
```

Figure 1: Output for Q2

Discussion

- *Q: Which 5 users are most correlated to the substitute you (i.e., which 5 users rate movies most similarly to the substitute you?)*
- *Q: Which 5 users are least correlated (i.e., negative correlation)?*

The final output for the top 5 users that were most correlated to my substitute me were 604,713,914,80, and 237 with their respective correlating score of 1.000000000000004,1.000000000000001,1.0,1.0 and 1.0.

For the bottom 5 users that were most correlated to my substitute me were, 760,547,432,317,and 112. Their correlative score was the same, -1.000000000000004.

For the driver lines 130 to 159 generated the resulting output. First I loaded the u.items, u.data and u.user in this function and stored them in a dictionary, dictionary and list variables respectively. Then called the topMatches function which I passed in my substitute me user id in.

Q3

Compute ratings for all the films that the substitute you has not seen.

Provide a list of the top 5 recommendations for films that the substitute you should see.

Provide a list of the bottom 5 recommendations (i.e., films the substitute you is almost certain to hate).

Answer

```
Top 5 movies recommendations for substitute me:
(5.0, 'Unforgiven (1992)')
(5.0, 'Unforgettable (1996)')
(5.0, 'Tombstone (1993)')
(5.0, 'Silence of the Lambs, The (1991)')
(5.0, 'Net, The (1995)')

Bottom 5 movies recommendations for substitute me:
(1.0, 'Usual Suspects, The (1995)')
(1.0, 'Thinner (1996)')
(1.0, 'Natural Born Killers (1994)')
(1.0, 'Full Monty, The (1997)')
(1.0, 'Albino Alligator (1996)')
```

Figure 2: Output for Q3

Discussion

Q. Provide a list of the top 5 recommendations for films that the substitute you should see.Provide a list of the bottom 5 recommendations (i.e., films the substitute you is almost certain to hate).

The question 2's two.py in Listing 2 is used to answer this question.

In line 155 to 159, function getRecommendations() was to get my substiture user recommended list, I Used *recommend[0:5] to get first 5 recommendation. This code was gotten from `https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter2/recommendations.py`.

Top 5 movies recommendations for substitute me, Unforgiven (1992), Unforgettable (1996), Tombstone (1993), Silence of the Lambs, The (1991), Net, The (1995). Bottom 5 movies recommendations for substitute me,in line 159 *recommend[len(recommend)-5:len(recommend)] used it to extract the bottom 5 movie recommendations for substitute me Usual Suspects, The (1995), Thinner (1996), Natural Born Killers (1994), Full Monty, Albino Alligator (1996).

Q4

Choose your (the real you, not the substitute you) favorite and least favorite film from the data.

For each film, generate a list of the top 5 most correlated and bottom 5 least correlated films.

Q: Based on your knowledge of the resulting films, do you agree with the results? In other words, do you personally like/dislike the resulting films?

If you have not heard of the recommended movies, search for the movie's trailer on YouTube and watch it before you answer. If you do this, include the link to the trailer in your report. For example, the trailer for "Top Gun (1986)" was found by searching for "top gun 1986 trailer" on Google.

Answer

```
My best 5 recommended movies
[(1.000000000000004, 'Young Guns II (1990)'), (1.000000000000004, 'Reality Bites
 (1994)'), (1.000000000000001, 'Ran (1985)'), (1.000000000000001, 'Butch Cassidy
 and the Sundance Kid (1969)'), (1.0000000000000009, 'Chinatown (1974)')]


My worst 5 recommended movies
[(-1.0, 'Big Night (1996)'), (-1.0, 'Barbarella (1968)'), (-1.0, 'Another Stakeo
ut (1993)'), (-1.0000000000000007, "Fathers' Day (1997)"), (-1.000000000000004,
"Devil's Own, The (1997)")]
>>>
```

Figure 3: Output for Q4

Discussion

The question 2's two.py in Listing 2 is used to answer this question.

I first transformed the prefs dictionary to movies. I then selected my best (Vampire in Brooklyn (1995)) movie from the whole list in u.data. The function topMatches and bottomMatch handled the result of the questions 3.

Q: Based on your knowledge of the resulting films, do you agree with the results? In other-words, do you personally like/dislike the resulting films?

Top 5 movies recommend from (Vampire in Brooklyn (1995): *Young Guns II (1990)* `https://www.youtube.com/watch?v=r-FmfxLy7fo` *I like this good old movie.*
Reality Bites (1994) `https://www.youtube.com/watch?v=xDYGo0UgIVM` *It seem interesting and I would watch it. Just an average movie for me.*
Ran (1985) `https://www.youtube.com/watch?v=YwP_kXyd-Rw` *I love this one also, war action my favorite.*
Butch Cassidy and the Sundance Kid (1969) `https://www.youtube.com/watch?v=YdJW2UxvSFQ` *I love cow boys like movies*
Chinatown (1974) `https://www.youtube.com/watch?v=20FfiP7g4tU` *Quite boring but I would enjoy the detective part to movie*

Worst 5 movies recommended from (Vampire in Brooklyn (1995)

Big Night (1996) `https://www.youtube.com/watch?v=Yd8gK6EgpLM` *It is little boring to me because its a chef movie*
Barbarella (1968) `https://www.youtube.com/watch?v=M-fJg08wBKw` *Very old movie and I dont find it interesting at all*
Another Stakeout (1993) `https://www.youtube.com/watch?v=Gpm4lGyOVYc` *I like this one action, detective kind of movies*
Fathers' Day(1997) `https://www.youtube.com/watch?v=xsQfKt08X1k` *Not an inter-*

esting movie, too boring.
Devil's Own, The (1997)`https://www.youtube.com/watch?v=xZiWSY2SjAM` *I completely love the action in this movie just the first second of seeing it. As usual war movies gets me all the time.*

References

- `https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter2/recommendations.py`
- `https://www.example.com/reallyreallyreally-extra-long-URI/`
- `https://gis.stackexchange.com/questions/180933/how-to-slice-all-elements-in-a-list-to-a-certain-length`
- `https://stackoverflow.com/questions/32448414/what-does-colon-at-assignment-for-list-do-in-python/32448477`
- `http://www.hpc-carpentry.org/hpc-python/04-dicts/index.html`
- `https://www.kite.com/python/answers/how-to-append-an-element-to-a-key-in-a-dictionary-with-python`