Srividya Majeti

Assignment 6

CS 532: Introduction to Web Science
Dr. Michael Nelson
Spring 2016

April 1, 2016

Contents

1

Question 1

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

- what are their top 3 favorite films?
- bottom 3 least favorite films?

Based on the movie values in those 6 tables (3 users X (favorite + least)), 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").

This user is the "substitute you".

Following are the steps I have taken to the solve the problem:

- First I got all the data arranged in a JSON structure. I wrote the rating data, movie data in two JSON files and got aggregated user data with 'user_id', 'user_details' and 'movie_details'. This is explained in the function getData() which is illustrated in Listing 1.1
- To find three users who are closest to me in terms of age, gender and occupation I wrote three functions getFemaleUsers(), getFemaleUsersWithAgeCloserto24(), getFemaleUsersWithAgeCloserto24AndStudent() which are listed in Listings 1.2, 1.3, 1.4respectively.
- Then I got the top three and bottom three favorite films of the users. This is illustrated in Listing 1.5
- The table with three users with their top and bottom favorite films is illustrated below.
- I selected user with 'user_id' 711 as my subsitute because all our top favorite movies are similar. I like to watch animate films. The reason for not considering the remaining 2 users as my substitute because they ranked my favorite films with least score.

Table 1.1. User 1

User 1	Top Favorite Films	Least Favorite Films
711	Winnie the Pooh and the Blustery Day (1968)	Independence Day (ID4) (1996)
	Alice in Wonderland (1951)	Before Sunrise (1995)
	Good Will Hunting (1997)	Dragonheart (1996)

Table 1.2. User 2

User 2	Top Favorite Films	Least Favorite Films
917	City Hall (1996)	Four Rooms (1995)
	Dead Man Walking (1995)	Independence Day (ID4) (1996)
	Leaving Las Vegas (1995)	Tomorrow Never Dies (1997)

Table 1.3. User 3

User 3	Top Favorite Films	Least Favorite Films
875	Shawshank Redemption, The (1994)	Lion King, The (1994)
	Star Wars (1977)	American President, The (1995)
	Wings of Desire (1987)	Liar Liar (1997)

Code Listing

```
1
2  def getData(path='./data/movielens/'):
3    # DONE
4    ratingDataFile = open("ratingData.json","w")
5    ratingData ={}
6    for line in open(path + 'u.data'):
7      (user, movieid, rating, ts) = line.split('\t')
8      ratingData['user'] = user.strip()
9      ratingData['movieid'] = movieid.strip()
10     ratingData['rating'] = rating.strip()
11     # ratingData['ts'] = ts.strip()
12     ratingDataFile.write(json.dumps(ratingData)+",\n")
13
14   # DONE
15   moviesDataFile = open("movieData.json","w")
16   movies = {}
17   for line in open(path + 'u.item'):
18     movie_id = line.split('|')[0:1][0]
19     movie_name = line.split('|')[1:2][0]
20     movies['movie_id'] = re.sub(r'[^\x00-\x7F]',' ',
           movie_id)
21     movies['movie_name'] = re.sub(r'[^\x00-\x7F]',' ',
           movie_name)
```

```
22      moviesDataFile.write(json.dumps(movies) + ',\n')
23
24
25    # DONE
26    userDataFile = open("userData.json","w")
27    UserData = {}
28    for line in open(path + 'u.user'):
29      UserData['user_id'] = line.split('|')[0]
30      userDetails = {}
31      userDetails['age'] = line.split('|')[1]
32      userDetails['occupation'] = line.split('|')[3]
33      userDetails['gender'] = line.split('|')[2]
34      UserData['user_details']= userDetails
35      ratingDataFile = open("ratingData.json","r")
36      ratingData = json.load(ratingDataFile)
37      movieDetails_list = []
38      for user1 in ratingData:
39        r_id = user1['user']
40        if UserData['user_id'] == r_id:
41          movieDetails= {}
42          movieDetails['movie_id'] = user1['movieid']
43          movieDetails['movie_rating'] = user1['rating']
44          moviesDataFile = open("movieData.json","r")
45          movieData = json.load(moviesDataFile)
46          for user2 in movieData:
47            movie_id = user2['movie_id']
48            if user1['movieid'] == movie_id:
49              movieDetails['movie_name'] = user2['movie_name']
50              break
51          moviesDataFile.close()
52          movieDetails_list.append(movieDetails)
53      ratingDataFile.close()
54      UserData['movie_details']= movieDetails_list
55      userDataFile.write(json.dumps(UserData) + ',\n')
```

Listing 1.1. Function for getting ratingData movieData and aggregated user data

Code Listing

```
1  def getFemaleUsers():
2    f = open('listOfFemaleUsers','w')
3    userDataFile = open("userData.json","r")
4    userData = json.load(userDataFile)
5    # femaleUsers=[]
6    count = 0
7    countF = 0
8    for user in userData:
9      count += 1
10     if user['user_details']['gender'] =='F':     #gives only
           female users
```

```
11          countF += 1
12          f.write(json.dumps(user) + ',\n')
13      print "Total users:" +str(count)
14      print "number of female users:" +str(countF)
```

Listing 1.2. Function for getting female users

Code Listing

```
1  def getFemaleUsersWithAgeCloserto24():
2      f = open('listOfFemaleUsers','r')
3      userData = json.load(f)
4      f1 = open('listOfFemaleUsersWithAgeCloserTo24','w')
5      # femaleUsersWithAge24=[]
6      count24 = 0
7      countF = 0
8      for user in userData:
9          countF += 1
10         if user['user_details']['age'] < '26' and user['
               user_details']['age'] >'21':
11             count24 += 1
12             f1.write(json.dumps(user) + ',\n')
13     print "number of female users:" +str(countF)
14     print "number of female users with age 24:" +str(count24)
```

Listing 1.3. Function for getting female users who are close to age 24

Code Listing

```
1  def getFemaleUsersWithAgeCloserto24AndStudent():
2      f = open('listOfFemaleUsersWithAgeCloserTo24','r')
3      userData = json.load(f)
4      f1 = open('listOfFemaleUsersWithAge24AndStudent','w')
5      count24 = 0
6      countStudent = 0
7      for user in userData:
8          count24 += 1
9          if user['user_details']['occupation']=='student':
10             countStudent+= 1
11             f1.write(json.dumps(user) + ',\n')
12     print "number of female users with age 24:" +str(count24)
13     print "number of female users with age 24 and Student:" +
           str(countStudent)
```

Listing 1.4. Function for getting female users whose age is close to 24 and who are students

Code Listing

```
1  def topRatedAndLeastRated():
2      f = open('myList.json','r')
```

```
3   f1 = open('myListWithMovies.json','w')
4   userData = json.load(f)
5   dict= {}
6   for user in userData:
7     dict['user_id'] = user['user_id']
8     for movie in user['movie_details']:
9       if movie['movie_rating'] =='5':
10        dict['type'] ='Top Rated'
11        dict['movie_id']=movie['movie_id']
12        dict['movie_name']=movie['movie_name']
13        dict['movie_rating']=movie['movie_rating']
14        f1.write(json.dumps(dict) + ',\n')
15      if movie['movie_rating'] =='1' or movie['movie_rating'
            ] =='2' :
16        dict['type'] ='Least Rated'
17        dict['movie_id']=movie['movie_id']
18        dict['movie_name']=movie['movie_name']
19        dict['movie_rating']=movie['movie_rating']
20        f1.write(json.dumps(dict) + ',\n')
```

Listing 1.5. Function for getting top three and bottom three favorite films of the users who are closer to me

2

Question 2

Which 5 users are most correlated to the substitute you? Which 5 users are least correlated (i.e., negative correlation)?

Following are the steps I have taken to solve the problem:

- To find the most correlated and least correlated for the substitute me, I calculated the pearson score.
- The pearson score ranges from -1 to 1. Users who have correlation coefficient equal to or greater than one are categorized as most correlated and those with pearson score closer to -1 are categorised as least correlated (i.e., negative correlation).
- This is illustrated in the Listing 2.1
- The list of users with their pearson score is illustrated in the Table 2.1 2.2

Table 2.1. Most correlated users

UserId	Correlation Value
725	1.0
732	1.0
359	1.0
358	1.028
4	1.133

Table 2.2. Least correlated users

UserId	Correlation Value
434	-0.707
818	-0.738
27	-0.755
792	-0.866
281	-0.948

Code Listing

```
def sim_pearson(pref,p1,p2):
  # Get the list of mutually rated items
  si={}
  for item in pref[p1]:
    if item in pref[p2]: si[item]=1

  # if they are no ratings in common, return 0
  if len(si)==0: return 0

  # Sum calculations
  n=len(si)

  sum1 = 0
  for it in si:
  sum1 =  sum1 + int(pref[p1][it])

  sum2 = 0
  for it in si:
  sum2 =  sum2 + int(pref[p2][it])

  sum1Sq = 0
  for it in si:
  sum1Sq =  sum1Sq + pow(int(pref[p1][it]),2)

  sum2Sq = 0
  for it in si:
  sum2Sq =  sum2Sq + pow(int(pref[p2][it]),2)


  pSum = 0
  for it in si:
  pSum =  pSum + ( int(pref[p2][it]) * int(pref[p1][it]) )


  # Calculate r (Pearson score)
  num=pSum-(sum1*sum2/n)
  den=sqrt((sum1Sq-pow(sum1,2)/n)*(sum2Sq-pow(sum2,2)/n))
  if den==0: return 0

  r=num/den

  return r
```

Listing 2.1. Function for calculating pearson score between substitute me and all other users

3

Question 3

Compute ratings for all the films that the substitute you have 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).

Following are the steps I have taken to solve the problem:

- I used the getRecommendations() function from recommendations.py to get top 5 recommendations of films that the substitute me should see. This is illustrated in Listing 3.1
- The table for top recommended movies and least recommended movies for substitute me with Movie_id, movie_name and rating are in Table 3.1 3.2 respectively.

Table 3.1. Top Recommended movies for substitute me

Movie_Id	Movie_name	Rating
814	Great Day in Harlem, A (1994)	5.0
1653	Entertaining Angels: The Dorothy Day Story (1996)	5.0
1599	Someone Else's America (1995)	5.0
1536	Aiqing wansui (1994)	5.0
1500	Santa with Muscles (1996)	5.0

Table 3.2. Least Recommended movies for substitute me

Movie_Id	Movie_name	Rating
1374	Country Life (1994)	1.0
1354	Babyfever (1994)	1.0
1309	Very Natural Thing, A (1974)	1.0
1308	Venice/Venice (1992)	1.0
1290	Falling in Love Again (1980)	1.0

Code Listing

```
1  def getRecommendations(prefs,person,similarity=sim_pearson):
2    totals={}
3    simSums={}
4    for other in prefs:
5      # don't compare me to myself
6      if other==person: continue
7      sim=similarity(prefs,person,other)
8
9      # ignore scores of zero or lower
10     if sim<=0: continue
11     for item in prefs[other]:
12
13       # only score movies I haven't seen yet
14       if item not in prefs[person] or prefs[person][item
             ]==0:
15         # Similarity * Score
16         totals.setdefault(item,0)
17         totals[item] += int(prefs[other][item])*sim
18         # Sum of similarities
19         simSums.setdefault(item,0)
20         simSums[item]+=sim
21
22   # Create the normalized list
23   rankings=[(total/simSums[item],item) for item,total in
         totals.items()]
24
25   # Return the sorted list
```

```
26      rankings.sort()
27      rankings.reverse()
28      return rankings
```

Listing 3.1. Function for getting top recommended movies and least recommended movies for substitute me

4

Question 4

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. 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?

Following are the steps I have taken to solve the problem:

- I selected my favorite film as 'Lion King (1994)' Movie_Id '771' and least favorite movie as ' Johnny Mnemonic (1995)' with Movie_Id '71'.
- To get the top 5 most correlated movies and bottom 5 least correlated movies I have used the function calculateSimilarItems() from recommendations.py.
- This is illustrated in the Listing 4.1
- The output for this function is in the Figure 4.1
- I personally like 'Being Human (1993)' and 'Across the Sea of Time (1995)' movies. I did not see the other 3 movies, I read the summary in Wikipedia and IMDb, I liked '187 (1997)' , 'Underneath, The (1995)' but I did not like 'Go Fish (1994)'.

```
% python recommendations.py
Top five correlated films for:  Lion King, The (1994)
(1.1547005383792517, 'Being Human (1993)')
(1.0606601717798212, '187 (1997)')
(1.0434983894999017, 'Across the Sea of Time (1995)')
(1.0079052613579391, 'Go Fish (1994)')
(1.0, 'Underneath, The (1995)')
Bottom five correlated films for:  Lion King, The (1994)
(-0.8164965809277261, 'Spice World (1997)')
(-0.8164965809277261, 'Above the Rim (1994)')
(-0.8660254037844387, 'Mr. Magoo (1997)')
(-1.0, 'Welcome To Sarajevo (1997)')
(-1.0, 'Thousand Acres, A (1997)')
Top five correlated films for:  Johnny Mnemonic (1995)
(1.1547005383792517, 'Street Fighter (1994)')
(1.1547005383792517, 'Simple Twist of Fate, A (1994)')
(1.1547005383792517, 'Low Down Dirty Shame, A (1994)')
(1.1547005383792517, 'In Love and War (1996)')
(1.1547005383792517, 'Being Human (1993)')
Bottom five correlated films for:  Johnny Mnemonic (1995)
(-0.8164965809277261, 'Friday (1995)')
(-0.8164965809277261, 'Flirting With Disaster (1996)')
(-0.8164965809277261, 'Denise Calls Up (1995)')
(-0.8944271909999159, 'Richard III (1995)')
(-1.0, 'Gay Divorcee, The (1934)')
```

Fig. 4.1. Output with top 5 most correlated and bottom 5 least correlated movies for my most favorite and least favorite movie

Code Listing

```
1  def getRecommendations(prefs,person,similarity=sim_pearson):
2    totals={}
3    simSums={}
4    for other in prefs:
5      # don't compare me to myself
6      if other==person: continue
7      sim=similarity(prefs,person,other)
8
9      # ignore scores of zero or lower
10     if sim<=0: continue
11     for item in prefs[other]:
12
13       # only score movies I haven't seen yet
14       if item not in prefs[person] or prefs[person][item
             ]==0:
15         # Similarity * Score
16         totals.setdefault(item,0)
17         totals[item] += int(prefs[other][item])*sim
18         # Sum of similarities
19         simSums.setdefault(item,0)
20         simSums[item]+=sim
21
22   # Create the normalized list
23   rankings=[(total/simSums[item],item) for item,total in
         totals.items()]
24
25   # Return the sorted list
26   rankings.sort()
```

```
27    rankings.reverse()
28    return rankings
```

Listing 4.1. Function for getting top 5 correlated movies and bottom 5 correlated movies for my favorite and least favorite movies.

5

Extra-Credit Question-3

Rank the 1,682 movies according to the 1997/1998 MovieLense data. Now rank the same 1,682 movies according to todays (March 2016) IMDB data (break ties based on number of users, for example: 7.2 with 10,000 raters > 7.2 with 9,000 raters).
Draw a graph, where each dot is a film (i.e., 1,682 dots). The x-axis is the MovieLense ranking and the y-axis is today's IMDB ranking. What is Pearon's r for the two lists (along w/ the p-value)? Assuming the two user bases are interchangable (which might not be a good assumption), what does this say about the attitudes about the films after nearly 20 years?

Following are the steps I have taken to solve the problem:

- I ranked all the movies based on MovieLense. The python code for this is illustrated in 5.1
- Futhermore I ranked all the movies based on todays IMDB data. The python code for this is illustrated in 5.2
- For IMDB data I got the ranking out of 10.0, I normalized the ranking to 5.0.

- The output file for MovieLense with rating and movie names are in Figure 5.1.

```
3.51 Wag the Dog (1997)
3.59 Boogie Nights (1997)
3.09 Critical Care (1997)
2.92 Man Who Knew Too Little, The (1997)
3.1 Alien: Resurrection (1997)
3.3 Desperate Measures (1998)
2.9 Hard Rain (1998)
3.77 Face/Off (1997)
2.93 Hoodlum (1997)
3.33 Promesse, La (1996)
3.96 Ulee's Gold (1997)
3.16 Liar Liar (1997)
3.4 Breakdown (1997)
3.67 Rosewood (1997)
3.8 Donnie Brasco (1997)
3.11 Fierce Creatures (1997)
3.37 Absolute Power (1997)
3.6 Gattaca (1997)
3.23 Starship Troopers (1997)
4.26 Good Will Hunting (1997)
3.57 Heat (1995)
3.5 Sabrina (1995)
4.01 Sense and Sensibility (1995)
3.7 Leaving Las Vegas (1995)
3.46 Restoration (1995)
3.23 Bed of Roses (1996)
3.29 Once Upon a Time... When We Were Colored (1995)
3.48 Grifters, The (1990)
3.11 Geronimo: An American Legend (1993)
3.08 Kids in the Hall: Brain Candy (1996)
3.43 Mystery Science Theater 3000: The Movie (1996)
3.13 Aristocats, The (1970)
1.87 All Dogs Go to Heaven 2 (1996)
4.16 Fargo (1996)
3.26 Heavy Metal (1981)
```

Fig. 5.1. MovieLense rating

- The output file for IMDB with rating and movie names are in Figure 5.2

```
3.6 GoldenEye (1995)
3.35 Four Rooms (1995)
3.45 Get Shorty (1995)
3.3 Copycat (1995)
3.6 Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)
4.05 Twelve Monkeys (1995)
3.4 Babe (1995)
3.8 Dead Man Walking (1995)
3.75 Richard III (1995)
3.85 Seven (Se7en) (1995)
4.3 Usual Suspects, The (1995)
3.55 Mighty Aphrodite (1995)
4.15 Postino, Il (1994)
3.2 French Twist (Gazon maudit) (1995)
3.65 From Dusk Till Dawn (1996)
3.9 White Balloon, The (1995)
3.75 Antonia's Line (1995)
3.5 Angels and Insects (1995)
3.45 Muppet Treasure Island (1996)
4.2 Braveheart (1995)
4.15 Taxi Driver (1976)
3.35 Rumble in the Bronx (1995)
3.45 Birdcage, The (1996)
3.3 Brothers McMullen, The (1995)
3.4 Bad Boys (1995)
3.8 Apollo 13 (1995)
2.7 Batman Forever (1995)
3.9 Belle de jour (1967)
3.65 Crimson Tide (1995)
4.0 Crumb (1994)
```

Fig. 5.2. IMDB rating

Code Listing

```
1  import json
2
3  def getMovieName(id):
4    moviesDataFile = open("movieData.json","r")
5    movieData = json.load(moviesDataFile)
6    for user in movieData:
7      if user['movie_id'] == id:
8        return user['movie_name']
9
10 def rankBasedOnMovieLense():
11   f = open('averageRank','w')
12   ratingDataFile = open("ratingData.json","r")
13   ratingData = json.load(ratingDataFile)
14   count = 0
15   siri_dic = {}
16   for user in ratingData:
17     rating = user['rating']
18     movie_id = user['movieid']
19     user_id = user['user']
20     if movie_id in siri_dic:
21       siri_dic[movie_id]['rating'] = siri_dic[movie_id]['
             rating'] + int(rating)
22       siri_dic[movie_id]['count'] = siri_dic[movie_id]['
             count'] + 1
23       siri_dic[movie_id]['average'] = siri_dic[movie_id]['
             rating'] * 1.0 / siri_dic[movie_id]['count']
24     else:
25       siri_dic[movie_id] = {'rating': int(rating), 'count':
             1, 'average': rating, 'movie_name': getMovieName(
             movie_id)}
26   f.write(json.dumps(siri_dic) + ',\n')
27 rankBasedOnMovieLense()
```

Listing 5.1. Python code for ranking movies based on MovieLense data.

Code Listing

```
1  import requests
2  import json
3
4  BASE_URL = 'http://www.omdbapi.com/'
5  MAX_RATING = 5
6
7  #removes all the junk and returns the name of the movie in
       correct format
8  def parseMovieName(movieName):
9    movie = movieName.strip().split('(')
10   name = movie[0].split(',')
11   try:
12     movieName = name[1].strip() + ' ' + name[0].strip()
13   except IndexError:
14     movieName = name[0].strip()
15   return movieName
16
17 #api call returns json with movie rating
18 def getRatingFromIMDb(movieName):
19   movie = parseMovieName(movieName)
20   params = dict(i='',t=movie.lower())
21   r = requests.get(BASE_URL, params=params)
22   # print r.content
23   data = json.loads(r.content)
24   dictionary = {}
25   if data['Response'] == 'True':
26     dictionary['rating'] = data['imdbRating']
27     dictionary['votes'] = data['imdbVotes']
28     dictionary['name'] = movieName.replace('"',"'")
29     return dictionary, 0
30   else:
31     dictionary['rating'] = -1
32     return dictionary, 1
33
34 #highest = maximum rating in the list.
35 #lowest = minimum rating in the list.
36 #currentRating = rating to be normalized
37 #temp is multiplied by MAX_RATING for normalizing it to
       scale of MAX_RATING.
38 def normalizeRatings(highest, lowest, currentRating):
39   temp = float(currentRating - lowest)/float(highest -
         lowest)
40   return temp * MAX_RATING
41
42 COUNT = 0
43 w = open('imdbRating','a')
44 f = open('u.item','r')
```

```
45 ratingList = []
46 for line in f:
47   data = line.split('|')
48   ratingData, count = getRatingFromIMDb(data[1])
49   if count == 0:
50     ratingList.append(ratingData)
51     w.write(json.dumps(ratingData) + ',\n')
52   else:
53     COUNT += 1
54     print 'MovieLens Name:' + data[1], ', Processed Name:' +
           parseMovieName(data[1]) + '--\n'
55
56 print COUNT
```

Listing 5.2. Python code for ranking movies based on IMDB data.

References

1. 100k dataset: http://grouplens.org/datasets/movielens/100k/, 2016, GroupLens