Web Science cs532: Assignment #7

Due on Thursday, March 31, 2016

Dr.Michael.L.Nelson 4:20pm

Zetan Li

Contents

Problem 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?

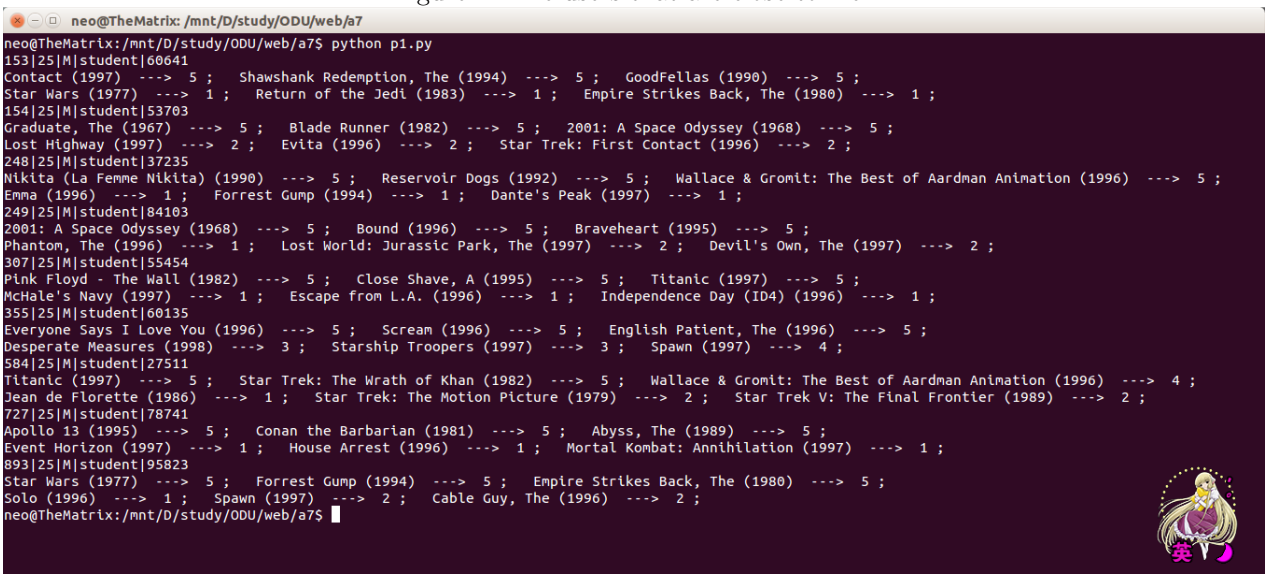
Based on the movie values in those 6 tables (3 users X (favorite + least)), choose a user that you feel is most like you. Feel free to note any outliers (e.g., "I mostly identify with user 123, except I did not like "Ghost" at ll").

This user is the "substitute you".

SOLUTION

To get users that are closest to me, first i have to go through all the user data and retrieve user that have the same gender, age and occupation as me.
Then in python, storing all the movie rating in dictionaries using user id for category. Sort the ratings and pick top 3 and bottom 3 as result.

Figure 1: The users that are close to me



Here I pick student 893 as "substitute you", because he has Star Wars and Forrest Gump in his favorite, which is the same as me.

Listing 1: Python code to get closet user and their movie preferences

```
users={}
movies={}
linktable={}
#parse rating file
linkfile=open('ml-100k/u.data')
strline=linkfile.readlines()
for line in strline:
    uid,itemid,rating,_=line.split('\t')
```

Problem 1 continued on next page...

```
     if uid not in linktable:
          linktable[uid]={}
     linktable[uid][itemid]=int(rating)
linkfile.close()
#parse movie file
moviefile=open('ml-100k/u.item')
strline=moviefile.readlines()
for line in strline:
     tuples=line.split('|')
     movies[tuples[0]]=tuples[1]
moviefile.close()
#parse user file and filter
datafile=open('ml-100k/u.user')
strline=datafile.readlines()
for line in strline:
     tuples=line.split('|')
     if tuples[1] == '25' and  tuples[2]=='M' and tuples[3]=='student' :
          print (line.strip())
          movielist=sorted(linktable[tuples[0]],key=linktable[tuples[0]].get,
              reverse=True)
          for i in  range(3) :
               print(movies[movielist[i]]+'  --->  '+str(linktable[tuples[0]][
                   movielist[i]])+' ;  '),
          print
          for j in range(-1,-4,-1):
               print(movies[movielist[j]]+'  --->  '+str(linktable[tuples[0]][
                   movielist[j]])+' ;  '),
          print
```

Problem 2

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

SOLUTION

As in problem 1, I already stored movie ratings in the category of user ids. Next step is to explore all the users' data, pick the rating of the movie that both substitute me and them have seen, then calculate similarity.

Here, as I choose the pearson r for similarity, sometimes it may face "divide by zero" warning. In that case, distance algorithm will be performed as alternative plan.

When all the similarity has been calculated, give them a sort, and then we can get top 5 and bottom 5 correlated users.

Listing 2: Python code to get correlated users

```
import math
from math import *
import scipy
from scipy import stats
from scipy.spatial import distance
users={}
movies={}
linktable={}

correlation={}
#parse rating file
linkfile=open('ml-100k/u.data')
strline=linkfile.readlines()
for line in strline:
     uid,itemid,rating,_=line.split('\t')
     if uid not in linktable:
          linktable[uid]={}
     linktable[uid][itemid]=int(rating)
linkfile.close()
#parse movie file
moviefile=open('ml-100k/u.item')
strline=moviefile.readlines()
for line in strline:
     tuples=line.split('|')
     movies[tuples[0]]=tuples[1]
moviefile.close()
#calculate correlation
pickedId='893'
pickedUser=linktable[pickedId]
for uid in linktable :
     if uid==pickedId:
          continue
     pickedUserRating=[]
     currentUserRating=[]
     for mid in pickedUser:
          if linktable[uid].has_key(mid) :
               pickedUserRating.append(pickedUser[mid])
               currentUserRating.append(linktable[uid][mid])
     if len(currentUserRating)==0 :
```

```
            correlation[uid]=0
        else:
            correlation[uid]=scipy.stats.pearsonr(pickedUserRating,currentUserRating
                )[0]
            if not correlation[uid] or math.isnan(correlation[uid]) :
                correlation[uid]=float(1)/(float(1)+scipy.spatial.distance.
                    euclidean(pickedUserRating,currentUserRating))

correlationArray=sorted(correlation,key=correlation.get,reverse=True)
print('5 users are most correlated to the substitute you:')
for i in  range(5) :
    print(correlationArray[i]+'  ( correlation: '+str(correlation[
        correlationArray[i]])+' ) ')

print('5 users are least correlated to the substitute you:')
for j in range(-1,-6,-1):
    print(correlationArray[j]+' ( correlation: '+str(correlation[correlationArray
        [j]])+' ) ')
```

Here's the result:
5 users are most correlated to the substitute you:
420 (correlation: 1.0)
191 (correlation: 1.0)
440 (correlation: 1.0)
333 (correlation: 1.0)
858 (correlation: 1.0)
5 users are least correlated to the substitute you:
604 (correlation: -1.0)
309 (correlation: -1.0)
212 (correlation: -1.0)
469 (correlation: -1.0)
319 (correlation: -1.0)

Problem 3

Compute ratings for all the films that the substitute you hasn't 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).

SOLUTION

To get most likely rating for the film that one hasn't seen, we have to calculate the weighted average rating of all other users.

Note that here we ignored the user that have zero and negative similarity. (If those data are included, the result will be weird, with average rating that more than 5)

Sorting the weighted average rating of all the film that substitute me haven't seen, and pick top 5 and bottom 5 as result.

Listing 3: Python code to get estimated rating of unseen movies

```
import math
from math import *
import scipy
from scipy import stats
from scipy.spatial import distance

movies={}
linktable={}

correlation={}
#parse rating file
linkfile=open('ml-100k/u.data')
strline=linkfile.readlines()
for line in strline:
     uid,itemid,rating,_=line.split('\t')
     if uid not in linktable:
          linktable[uid]={}
     linktable[uid][itemid]=float(rating)
linkfile.close()
#parse movie file
moviefile=open('ml-100k/u.item')
strline=moviefile.readlines()
for line in strline:
     tuples=line.split('|')
     movies[tuples[0]]={'name' : tuples[1], 'wtotal':0, 'stoal':0, 'erate' : 0}
moviefile.close()

#calculate correlation
pickedId='893'
pickedUser=linktable[pickedId]
for uid in linktable :
     if uid==pickedId:
          continue
     pickedUserRating=[]
     currentUserRating=[]
     for mid in pickedUser:
          if linktable[uid].has_key(mid) :
               pickedUserRating.append(pickedUser[mid])
               currentUserRating.append(linktable[uid][mid])
```

Problem 3 continued on next page...

```
     if len(currentUserRating)==0 :
          correlation[uid]=0
     else:
          correlation[uid]=scipy.stats.pearsonr(pickedUserRating,currentUserRating
              )[0]
          if not correlation[uid] or math.isnan(correlation[uid]) :
               correlation[uid]=float(1)/(float(1)+scipy.spatial.distance.
                   euclidean(pickedUserRating,currentUserRating))
     #calculate estimated rating
     for mid in linktable[uid]:
          # ignore scores of zero or lower and only score movies I haven't seen
              yet
          if mid not in pickedUser and  correlation[uid]>0:
               movies[mid]['wtotal']+=linktable[uid][mid]*correlation[uid]
               movies[mid]['stoal']+=correlation[uid]
#calculate rating
for m in movies:
     if  movies[m]['stoal']!= 0:
          movies[m]['erate']=float(movies[m]['wtotal'])/float(movies[m]['stoal'])
     # if movies[m]['erate']>5 or movies[m]['erate']< -5:
     #    print movies[m]


movieList=sorted(movies.values(),key=lambda v  : v['erate'],reverse=True)
print ('Top 5 recommendations for films')
for mv in movieList[:5]:
     print(mv['name']+'      Most likely rating:  '+ str(mv['erate']))
print ('\nBottom 5 recommendations for films')
for mv in movieList[-5:] :
     print(mv['name']+'      Most likely rating:  '+ str(mv['erate']))
```

The result is:
Top 5 recommendations for films
Entertaining Angels: The Dorothy Day Story (1996) — Most likely rating: 5.0
Great Day in Harlem, A (1994) — Most likely rating: 5.0
They Made Me a Criminal (1939) — Most likely rating: 5.0
Someone Else's America (1995) — Most likely rating: 5.0
Saint of Fort Washington, The (1993) — Most likely rating: 5.0

Bottom 5 recommendations for films
Event Horizon (1997) — Most likely rating: 0
Mimic (1997) — Most likely rating: 0
Rock, The (1996) — Most likely rating: 0
Twister (1996) — Most likely rating: 0
Circle of Friends (1995) — Most likely rating: 0

Problem 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?

SOLUTION

Exploring all the films in the file, I choose the film below as my input data:
Favorite movie:
121—Independence Day (ID4) (1996)

Least favorate movie:
870—Touch (1997)

To get recommendations on movies instead of users, we switch the row and column and calculate similarity of the movies as we did with users.
This time, we store the user rating in the category of movie id instead of user id.
Then pick the top 5 and bottom 5 movie from movie table that sorted by similarity.

Listing 4: Python code to calculate similarity of the given movie

```
import math
from math import *
import scipy
from scipy import stats
from scipy.spatial import distance
users={}
movies={}
linktable={}

correlation={}
def recommandate(pid):
    correlation={}
    pickedMovie=linktable[pid]
    for mid in linktable :
        if mid==pid:
            continue
        pickedMovieRating=[]
        currentMovieRating=[]
        for uid in pickedMovie:
            if  uid in linktable[mid] :
                pickedMovieRating.append(pickedMovie[uid])
                currentMovieRating.append(linktable[mid][uid])
        if len(currentMovieRating)==0 :
            correlation[mid]=0
        else:
            correlation[mid]=scipy.stats.pearsonr(pickedMovieRating,
                currentMovieRating)[0]
            if not correlation[mid] or math.isnan(correlation[mid]) :
                correlation[mid]=float(1)/(float(1)+scipy.spatial.distance.
                    euclidean(pickedMovieRating,currentMovieRating))
    correlationArray=sorted(correlation,key=correlation.get,reverse=True)
```

```
     print('Top 5 most correlated movies:')
     for m in  correlationArray[:5] :
          print(movies[m] +'  ( correlation: '+str(correlation[m])+' ) ')

     print('Bottom 5 least correlated movies:')
     for m in correlationArray[-5:] :
          print(movies[m]+' ( correlation: '+str(correlation[m])+' ) ')

#parse rating file
linkfile=open('ml-100k/u.data')
strline=linkfile.readlines()
for line in strline:
     uid,itemid,rating,_=line.split('\t')
     if itemid not in linktable:
          linktable[itemid]={}
     linktable[itemid][uid]=float(rating)
linkfile.close()
#parse movie file
moviefile=open('ml-100k/u.item')
strline=moviefile.readlines()
for line in strline:
     tuples=line.split('|')
     movies[tuples[0]]=tuples[1]
moviefile.close()

#calculate correlation
pickedId='121'
print('favorate move: 121|Independence Day (ID4) (1996)')
recommandate(pickedId)
print
print('Least favorate move: 870|Touch (1997)')
pickedId='870'
recommandate(pickedId)
```

The result is:

Favorate movie: 121—Independence Day (ID4) (1996)
Top 5 most correlated movies:
Wife, The (1995) (correlation: 1.0)
Savage Nights (Nuits fauves, Les) (1992) (correlation: 1.0)
Collectionneuse, La (1967) (correlation: 1.0)
Truth or Consequences, N.M. (1997) (correlation: 1.0)
Intimate Relations (1996) (correlation: 1.0)

Bottom 5 least correlated movies:
Crows and Sparrows (1949) (correlation: -1.0)
Kicked in the Head (1997) (correlation: -1.0)
Underworld (1997) (correlation: -1.0)
Johnny 100 Pesos (1993) (correlation: -1.0)
Forbidden Christ, The (Cristo proibito, Il) (1950) (correlation: -1.0)

==========

Least favorate movie: 870—Touch (1997)
Top 5 most correlated movies:
Hoodlum (1997) (correlation: 1.0)
Ulee's Gold (1997) (correlation: 1.0)
Rosewood (1997) (correlation: 1.0)
Good Will Hunting (1997) (correlation: 1.0)
Restoration (1995) (correlation: 1.0)

Bottom 5 least correlated movies:
I Know What You Did Last Summer (1997) (correlation: -1.0)
Smilla's Sense of Snow (1997) (correlation: -1.0)
Halloween: The Curse of Michael Myers (1995) (correlation: -1.0)
Amistad (1997) (correlation: -1.0)
She's So Lovely (1997) (correlation: -1.0)

For me this result seems a little weird to me. I don't know the movie listed above, and when i try to get some information about them though google, some of them seems interesting but in totally different category.