INTRODUCTION TO WEB SCIENCES:
Assignment 7

Manoj Chandra Kompalli

31 March 2016

Contents

1 Question 1:

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".

1.1 Approach

Here my task was to find out a substitute of me.

- I have used the u.user file to collect the user data.
- I extracted all the users matching with my age, gender, occupation
- I extracted the top 3 from my list
- For each user, I stored the itemids and rating from u.data file in a dictionary
- I have sorted the dictionary based on the rating
- I have extracted the top three most rated and least rated three items
- I have extracted the movie names for the items using u.items
- I have observed that user 33 has two of my favourite movies like Titanic and Tomorrow never dies.
- Therefore, I chose user 33 to be my substitute

1.2 Code Listing

1.2.1 substitute.py to generate favorite and least favorite movies

```
import collections
import sys
from itertools import groupby
favmovies=[]
itemsdict ={}
itemsdict2={}
itemsdict3={}
leastdict={}
leastdict2={}
leastdict3={}
def userMovies(movilefile):
  c=0




  print 'top movies for user 33'
  print 'itemid  rating'
  for line in movilefile:
    (user_id, item_id, rating, timestamp ) = line.split('\t')

    rating = float(rating)
    try:
      if(user_id=='33'):
        itemsdict[item_id].append(rating)

    except KeyError:
      itemsdict[item_id] = list()
      itemsdict[item_id].append( rating )
  for key, value in sorted(itemsdict.iteritems(),reverse=True, key=lambda (k,v): (v,k)
   ):
    c=c+1
    if(c<4):
      print "%s: %s" % (key, value)
      favmovies.append(key)

  c=0
  moviefile.seek(0)
  print 'least favourite movies for user 33'
  print 'itemid  rating'
  for line in movilefile:
    (user_id, item_id, rating, timestamp ) = line.split('\t')

    rating = float(rating)
    try:
      if(user_id=='33'):
        leastdict[item_id].append(rating)

    except KeyError:
      leastdict[item_id] = list()
      leastdict[item_id].append( rating )
  for key, value in sorted(leastdict.iteritems(), key=lambda (k,v): (v,k)):
    c=c+1
```

```
    if(c<4):
      print "%s: %s" % (key, value)

  c=0
  moviefile.seek(0)
  print 'top movies for user 37'
  print 'itemid  rating'
  for line in movilefile:
    (user_id, item_id, rating, timestamp ) = line.split('\t')

    rating = float(rating)
    try:
      if(user_id=='37'):
        itemsdict2[item_id].append(rating)

    except KeyError:
      itemsdict2[item_id] = list()
      itemsdict2[item_id].append( rating )
  for key, value in sorted(itemsdict2.iteritems(),reverse=True, key=lambda (k,v): (v,k
   )):
    c=c+1
    if(c<4):
      print "%s: %s" % (key, value)
      favmovies.append(key)
  c=0
  moviefile.seek(0)
  print 'least favourite movies for user 37'
  print 'itemid  rating'
  for line in movilefile:
    (user_id, item_id, rating, timestamp ) = line.split('\t')

    rating = float(rating)
    try:
      if(user_id=='37'):
        leastdict2[item_id].append(rating)

    except KeyError:
      leastdict2[item_id] = list()
      leastdict2[item_id].append( rating )
  for key, value in sorted(leastdict2.iteritems(), key=lambda (k,v): (v,k)):
    c=c+1
    if(c<4):
      print "%s: %s" % (key, value)

  c=0
  moviefile.seek(0)
  print 'top movies for user 66'
  print 'itemid  rating'
  for line in movilefile:
    (user_id, item_id, rating, timestamp ) = line.split('\t')

    rating = float(rating)
    try:
      if(user_id=='66'):
        itemsdict3[item_id].append(rating)

    except KeyError:
```

```
109         itemsdict3[item_id] = list()
110         itemsdict3[item_id].append( rating )
111     for key, value in sorted(itemsdict3.iteritems(),reverse=True, key=lambda (k,v): (v,k
        )):
112       c=c+1
113       if(c<4):
114         print "%s: %s" % (key, value)
115         favmovies.append(key)
116     c=0
117     moviefile.seek(0)
118     print 'least favourite movies for user 66'
119     print 'itemid  rating'
120     for line in movilefile:
121       (user_id, item_id, rating, timestamp ) = line.split('\t')
122 
123       rating = float(rating)
124       try:
125         if(user_id=='66'):
126           leastdict3[item_id].append(rating)
127 
128       except KeyError:
129         leastdict3[item_id] = list()
130         leastdict3[item_id].append( rating )
131     for key, value in sorted(leastdict3.iteritems(), key=lambda (k,v): (v,k)):
132       c=c+1
133       if(c<4):
134         print "%s: %s" % (key, value)
135 
136     c=0
137     moviefile.seek(0)
138 def getUsers(userfile):
139     count=0
140     for line in userfile:
141       line=line.split("|")
142       id=line[0]
143       age=line[1]
144       gender=line[2]
145       occupation=line[3]
146 
147       if(age=='23' and gender=='M' and occupation=='student' and count<3):
148         count=count+1
149         print 'top 3 substitutes'
150         print 'id'+' '+ 'count'+' '+ 'occupation'+' '+' '+'gender'+ ' '+'age'
151         print id,count,occupation,gender,age
152 
153 
154 def getTitles(imdbfile):
155     ct=0
156     namesdict = {}
157     print 'top movies of my substitutes'
158     for line in imdbfile:
159       # print line
160       line  = line.split("|")
161       id    = line[0]
162       title = line[1]
163 
164       namesdict[ id ] = title
```

```
    for key,value in namesdict.items():
      try:
        if key==favmovies[ct] and ct<9:
          ct=ct+1
          print(key,value)
      except:
        pass
  print 'my final substitute is user 33 because i like 2 of his movies titanic and
    tomorrow never dies'
```

```
if __name__=='__main__':
  userfile=open("u.user","r")
  moviefile=open("u.data","r")
  imdbfile=open("u.item","r")
  users=getUsers(userfile)
  movies=userMovies(moviefile)
  titles=getTitles(imdbfile)
```

1.3 Generated output on shell

```
sirius:~/wsprograms/a7new/a7/q1> python substitute.py
top movies for user 33
itemid  rating
313: [5.0]
751: [4.0]
682: [4.0]
least favourite movies for user 33
itemid  rating
245: [3.0]
294: [3.0]
307: [3.0]
top movies for user 37
itemid  rating
68: [5.0]
62: [5.0]
597: [5.0]
least favourite movies for user 37
itemid  rating
82: [1.0]
118: [2.0]
121: [2.0]
top movies for user 66
itemid  rating
742: [5.0]
50: [5.0]
471: [5.0]
least favourite movies for user 66
itemid  rating
21: [1.0]
286: [1.0]
877: [1.0]
top movies of my substitutes
('313', 'Titanic (1997)')
('751', 'Tomorrow Never Dies (1997)')
('682', 'I Know What You Did Last Summer (1997)')
('68', 'Crow, The (1994)')
('62', 'Stargate (1994)')
('597', 'Eraser (1996)')
('742', 'Ransom (1996)')
('50', 'Star Wars (1977)')
('471', 'Courage Under Fire (1996)')
my final substitute is user 33 because i like 2 of his movies titanic and tomorrow never dies
sirius:~/wsprograms/a7new/a7/q1>
```

Figure 1: Top and least rated movies and their ratings of user 33

2 Question 2:

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

2.1 Approach

Here, I used the recommendations.py file mentioned in the book Programming Collective Intelligence .

- I have used topMatches() method to extract the users correlated with my substitute user(user 33).
- I have stored the results in a dictionary and extracted the top five users from it for the users with most correlation
- I have extracted the bottom five users from the same dictionary to get the bottom five for the users with least correlation

2.2 Code Listing

2.2.1 correlation.py

```
#!/usr/local/bin/python

import sys
import recommendations
import operator
def getRatings(ratingsfile):
  itemsdict = {}
  count=0
  for line in ratingsfile:
    (user_id, item_id, rating, timestamp) = line.split('\t')
    if user_id in itemsdict:
      itemsdict[user_id][item_id] = float(rating)

    else:
      count=count+1
      itemsdict[user_id]={}
  output= recommendations.topMatches(itemsdict,'33',n=count)
  top = output[:5]
  print 'top 5 correlated users to user 33 are'
  print top
  bot = output[-5:]
  print 'bottom 5 correlated users to user 33 are'
  print bot
if __name__ == '__main__':


    ratingsfile = open ("../ml-100k/u.data", "r")
    ratings     = getRatings(ratingsfile)
```

2.3 Output

2.3.1 Response showing most and least correlated users to my substitute

```
sirius:~/wsprograms/a7new/a7/q2> python correlation.py
top 5 correlated users to user 33 are
[(1.000000000000004, '890'), (1.0000000000000018, '937'), (1.0000000000000018, '718'), (1.0, '95'), (1.0, '935')]
bottom 5 correlated users to user 33 are
[(-1.0, '136'), (-1.0, '103'), (-1.000000000000001, '736'), (-1.0000000000000018, '657'), (-1.0000000000000027, '797'
)]
sirius:~/wsprograms/a7new/a7/q2>
```

Figure 2: Pearson Coefficients and Correlated users to user 33

3 Question 3:

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).

3.1 Approach

My substitute you has seen and liked a fair bit of movies. Based on this dataset I have to find the most recommended movies and least recommended movies for my substitute.

- I used the getRecommendations method from recommendations.py which returned me all the recommended movies for a particular user.
- I fetched the movie titles and their respective ratings for all the resulting movie titles
- The resulting movies are sorted based on their ratings

3.2 Code Listing

3.2.1 recsub.py

```
import recommendations
import sys
from itertools import groupby
itemsdict ={}
def getTitles(imdbfile, movie_list):
  ct=0
  namesdict = {}
  movie_id = []
  ratings=[]
  for i in movie_list:
    movie_id.append(i[1])
    ratings.append(i[0])
  movie_titles = []

  for line in imdbfile:
    # print line
    line  = line.split("|")
    id    = line[0]
    title = line[1]

    namesdict[ id ] = title
    if id in movie_id:
      movie_titles.append(title)

  return movie_titles,ratings

if __name__=='__main__':
  moviefile   = open("u.data","r")

  pref_dic = {}
  rec=[]
  for line in moviefile:
```

```
    (user_id, item_id, rating, timestamp ) = line.split('\t')
    if user_id in pref_dic:
      pref_dic[user_id][item_id] = rating
    else:
      pref_dic[user_id] = {}
  result = recommendations.getRecommendations(pref_dic, '33')
  rec.append(result)
```

```
  bottom5= result[-5:]

  top5 = result[:5]
  imdbfile=open("u.item","r")

  top_titles,rate=getTitles(imdbfile, top5)
  print "Top Movies are"
  for i in range(0,len(top_titles)):
    print top_titles[i],rate[i]

  imdbfile=open("u.item","r")
  bottom_titles,rate = getTitles(imdbfile, bottom5)
  print "Bottom Movies are"
  for j in range(0,len(bottom_titles)):
    print bottom_titles[j],rate[j]
```

3.3 Output

3.3.1 Top movies along with their ratings

```
sirius:~/wsprograms/a7new/a7/q3> python recsub.py
Top Movies are
Great Day in Harlem, A (1994) 5.0
Two or Three Things I Know About Her (1966) 5.0
Hearts and Minds (1996) 5.0
Prefontaine (1997) 5.0
Santa with Muscles (1996) 5.0
Bottom Movies are
Theodore Rex (1995) 1.0
Country Life (1994) 1.0
New York Cop (1996) 1.0
Catwalk (1995) 1.0
Homage (1995) 1.0
sirius:~/wsprograms/a7new/a7/q3>
```

Figure 3: Recommended movies sorted based on ratings

4 Question 4:

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?

4.1 Approach

My favorite movie is Shawshank Redemption by a long way. Although there are many movies which belong to least favorite movie, I chose Smile like yours. Its rated 4.2 in IMDB right now. Now, my next job is to get the most correlated movies and least correlated movies for both my favorite and least favorite movies.

- To show correlation, again I have used Pearsons Coefficients.
- Some of the key functions I used from recommendations.py to extract the data are loadMovieLens which loads the movie lens data, topMatches which sorts the dictionary containing movie items and pearson coefficient.
- TopMatches again calls a method "simpearson" which finds the preferences and ultimately Pearson score.
- A Pearson score closer to 1 shows highly correlated movies and score close to -1 shows least correlated movies.
- I have calculated all twenty scores for my favorite and least favorite films.
- Unfortunately though I cannot comment on whether I like or dislike the resulting movies because I have hardly watched any of them

4.2 Code Listing

4.2.1 myfav.py

```
#!/usr/local/bin/python
import sys
import recommendationsedited

if __name__ == '__main__':

    outlier   = sys.argv[1]
    movie   = 'Shawshank Redemption, The (1994)'
    movie2='Smile Like Yours, A (1997)'

    count       = 5

    if outlier not in [ "top5", "bottom5"]:
        print "argument  1 must be either top5 or bottom5"
        sys.exit(1)


    prefs = recommendationsedited.loadMovieLens('../data')
    itemPrefs = recommendationsedited.transformPrefs(prefs)
    results   = recommendationsedited.topMatches(itemPrefs,movie,2000)
    results2= recommendationsedited.topMatches(itemPrefs,movie2,2000)

    if outlier == "top5":
    print 'for favourite movie Shawshank Redemption most correlated movies are'
    for i in results[0:count]:

      print i[0],i[1]
    print 'for least favourite movie A Smile like yours most correlated movies are'
    for i in results2[0:count]:

      print i[0],i[1]


    if outlier == "bottom5":


        results.reverse()
        results2.reverse()
        print 'for favourite movie Shawshank Redemption least correlated movies are'
        for i in results[0:count]:
            #print i[0],i[1]
      print i
        print 'for least favourite movie A Smile like yours bottom5 least correlated
   movies are'
        for i in results2[0:count]:
            #print i[0],i[1]
      print i
```

4.3 Output

4.3.1 Response showing Pearson's Coefficient along with correlated movies for Shawshank Redemption and A Smile like yours

```
sirius:~/wsprograms/a7new/a7/q4> python myfav.py "myfav.py"
argument  1 must be either top5 or bottom5
sirius:~/wsprograms/a7new/a7/q4> python myfav.py "top5"
for favourite movie Shawshank Redemption most correlated movies are
1.0 Penny Serenade (1941)
1.0 Newton Boys, The (1998)
1.0 Oscar & Lucinda (1997)
1.0 Wedding Gift, The (1994)
1.0 Search for One-eye Jimmy, The (1996)
for least favourite movie A Smile like yours most correlated movies are
1.0 City of Angels (1998)
1.0 Truth About Cats & Dogs, The (1996)
1.0 Silence of the Lambs, The (1991)
1.0 Othello (1995)
1.0 Fargo (1996)
sirius:~/wsprograms/a7new/a7/q4> python myfav.py "bottom5"
for favourite movie Shawshank Redemption least correlated movies are
(-1.0000000000000004, 'Clean Slate (Coup de Torchon) (1981)')
(-1.0, '1-900 (1994)')
(-1.0, 'American Dream (1990)')
(-1.0, 'Collectionneuse, La (1967)')
(-1.0, 'Colonel Chabert, Le (1994)')
for least favourite movie A Smile like yours bottom5 least correlated movies are
(-1.0000000000000007, 'Andre (1994)')
(-1.0, 'Flubber (1997)')
(-1.0, 'Nightwatch (1997)')
(-1.0, 'Vegas Vacation (1997)')
(-1.0, 'While You Were Sleeping (1995)')
sirius:~/wsprograms/a7new/a7/q4>
```

Figure 4: Pearson's Coefficient and movie name

References

[1] Movie Lens dataset http://grouplens.org/datasets/movielens/100k/,

[2] Programming Collective Intelligence http://forum.myquant.cn,

[3] Recommendations.py https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter2/recommendations.py

[]