Assignment 8

CS 595: Introduction to Web Science

Fall 2013

Shawn M. Jones

Finished on November 14, 2013

1

Question

`1. What 5 movies have the highest average ratings? Show the movies and their ratings sorted by their average ratings.`

Answer

Listing 2 on page 18 contains the source code for calculating the highest average ratings. It is run like so:

```
./highestratings.py 11  ../data/u.data ../data/u.item
```

The first argument is the number of movies to return. The second and third are the files to extract data from in order to calculate and produce the output.

Output looks like the following:

```
Great Day in Harlem, A (1994)  5.0
Entertaining Angels: The Dorothy Day Story (1996)  5.0
Someone Else's America (1995)  5.0
Aiqing wansui (1994)  5.0
Santa with Muscles (1996)  5.0
Saint of Fort Washington, The (1993)  5.0
Star Kid (1997) 5.0
Marlene Dietrich: Shadow and Light (1996)  5.0
Prefontaine (1997)  5.0
They Made Me a Criminal (1939)  5.0
Pather Panchali (1955)  4.625
```

As we can see, the top 10 all have an average rating of 5.0, and the 11th item is where the average rating starts to go down, so there are more than 5 with the highest average rating.

2

Question

2. What 5 movies received the most ratings? Show the movies and the number of ratings sorted by number of ratings.

Answer

Listing 3 on page 20 contains the source code for calculating the movies that received the most ratings.It is run like so:

```
./mostratings.py 5 ../data/u.data ../data/u.item
```

It's output looks like the following:

```
Star Wars (1977)         583
Contact (1997)  509
Fargo (1996)    508
Return of the Jedi (1983)       507
Liar Liar (1997)         485
```

3

Question

3. What 5 movies were rated the highest on average by women? Show the movies and their ratings sorted by ratings.

Answer

Listing 4 on page 22 contains the source code used to calculate the n movies that were rated highest on average by a given gender. For women, it is run like so:

```
./highestbygender.py 12 ../data/u.data ../data/u.item ../data/u.
    user "F"
```

The results are the following:

```
Year of the Horse (1997)          5.0
Telling Lies in America (1997)  5.0
Faster Pussycat! Kill! Kill! (1965)      5.0
Someone Else's America (1995)    5.0
Everest (1998)  5.0
Visitors, The (Visiteurs, Les) (1993)    5.0
Foreign Correspondent (1940)     5.0
Mina Tannenbaum (1994)  5.0
Stripes (1981)  5.0
Maya Lin: A Strong Clear Vision (1994)  5.0
Prefontaine (1997)       5.0
Schindler's List (1993) 4.63291139241
```

which show more than 5 movies rated with a score of 5.0 on average by women and the 12th item is where the average rating starts to go down, so there are more than 5 with the highest average rating among women.

4

Question

4. What 5 movies were rated the highest on average by men? Show the movies and their ratings sorted by ratings.

Answer

Listing 4 on page **??** contains the source code used to calculate the n movies that were rated highest on average by a given gender. For men, it is run like so:

```
./highestbygender.py 17 ../data/u.data ../data/u.item ../data/u.
    user "M"
```

The results are the following:

```
Great Day in Harlem, A (1994)  5.0
Little City (1998)  5.0
Entertaining Angels: The Dorothy Day Story (1996)  5.0
Leading Man, The (1996)  5.0
Love Serenade (1996)  5.0
Aiqing wansui (1994)  5.0
Santa with Muscles (1996)  5.0
Saint of Fort Washington, The (1993)  5.0
Delta of Venus (1994)  5.0
Star Kid (1997)  5.0
Marlene Dietrich: Shadow and Light (1996)  5.0
Letter From Death Row, A (1998)  5.0
Prefontaine (1997)  5.0
Hugo Pool (1997)  5.0
Quiet Room, The (1996)  5.0
They Made Me a Criminal (1939)  5.0
Two or Three Things I Know About Her (1966)  4.66666666667
```

which show more than 5 movies rated with a score of 5.0 on average by men and the 17th item is where the average rating starts to go down, so there are more than 5 with the highest average rating among men.

The men seem to agree more than the women, though, with 16 rather than 11 shared average scores.

5

Question

`5. What movie received ratings most like Top Gun? Which movie received ratings that were least like Top Gun (negative correlation)?`

Answer

Listing 5 on page 24 shows the source code to calculate which movie received ratings most like the given movie.

To give Pearson's Correlation Coefficients, the `calculateSimilarItems` function was altered to use the `sim_pearson` function instead of the default `sim_distance` function that already existed. The modified `calculateSimilarItems` function is shown in Listing 1.

```
124 def calculateSimilarItems(prefs,n=10):
125   # Create a dictionary of items showing which other items they
126   # are most similar to.
127   result={}
128   # Invert the preference matrix to be item-centric
129   itemPrefs=transformPrefs(prefs)
130   c=0
131   for item in itemPrefs:
132     # Status updates for large datasets
133     c+=1
134     if c%100==0: print "%d / %d" % (c,len(itemPrefs))
135     # Find the most similar items to this one
136     scores=topMatches(itemPrefs,item,n=n,similarity=sim_pearson)
137     result[item]=scores
138   return result
```

Listing 1: changed version of `recommendations.py` showing modified similarity function choice on line 136

To get the ratings most like a given move, the script is run like so:

```
./getFilmsLike.py 'Top Gun (1986)' 62 'most'
```

Which produces the following output consisting of movie title followed by Pearson score in parentheses:

```
100 / 1664
200 / 1664
300 / 1664
400 / 1664
500 / 1664
600 / 1664
```

```
700 / 1664
800 / 1664
900 / 1664
1000 / 1664
1100 / 1664
1200 / 1664
1300 / 1664
1400 / 1664
1500 / 1664
1600 / 1664
Movies most like 'Top Gun (1986)': '
Shiloh (1997) (1.0)
King of the Hill (1993) (1.0)
Bhaji on the Beach (1993) (1.0)
Wild America (1997) (1.0)
Wedding Gift, The (1994) (1.0)
Underground (1995) (1.0)
Two or Three Things I Know About Her (1966) (1.0)
Two Bits (1995) (1.0)
Total Eclipse (1995) (1.0)
The Innocent (1994) (1.0)
That Old Feeling (1997) (1.0)
Stars Fell on Henrietta, The (1995) (1.0)
Stalker (1979) (1.0)
Spirits of the Dead (Tre passi nel delirio) (1968) (1.0)
Show, The (1995) (1.0)
Shooter, The (1995) (1.0)
Selena (1997) (1.0)
Schizopolis (1996) (1.0)
Scarlet Letter, The (1926) (1.0)
Run of the Country, The (1995) (1.0)
Ponette (1996) (1.0)
Perfect Candidate, A (1996) (1.0)
Outlaw, The (1943) (1.0)
Old Lady Who Walked in the Sea, The (Vieille qui marchait dans
    la mer, La) (1991) (1.0)
Nothing to Lose (1994) (1.0)
New Jersey Drive (1995) (1.0)
Mr. Jones (1993) (1.0)
Metisse (Caf? au Lait) (1993) (1.0)
Maybe, Maybe Not (Bewegte Mann, Der) (1994) (1.0)
Manny & Lo (1996) (1.0)
Man of the Year (1995) (1.0)
Love Serenade (1996) (1.0)
Last Time I Saw Paris, The (1954) (1.0)
Killer (Bulletproof Heart) (1994) (1.0)
Jerky Boys, The (1994) (1.0)
I Like It Like That (1994) (1.0)
Horse Whisperer, The (1998) (1.0)
```

```
Hear My Song (1991) (1.0)
Grosse Fatigue (1994) (1.0)
Gone Fishin' (1997) (1.0)
Glass Shield, The (1994) (1.0)
Germinal (1993) (1.0)
Gabbeh (1996) (1.0)
Four Days in September (1997) (1.0)
Flower of My Secret, The (Flor de mi secreto, La) (1995) (1.0)
Fausto (1993) (1.0)
Even Cowgirls Get the Blues (1993) (1.0)
Enfer, L' (1994) (1.0)
Dream With the Fishes (1997) (1.0)
Dream Man (1995) (1.0)
Dangerous Ground (1997) (1.0)
Collectionneuse, La (1967) (1.0)
Clean Slate (Coup de Torchon) (1981) (1.0)
Calendar Girl (1993) (1.0)
Blood For Dracula (Andy Warhol's Dracula) (1974) (1.0)
Bliss (1997) (1.0)
Best Men (1997) (1.0)
American Dream (1990) (1.0)
Albino Alligator (1996) (1.0)
8 Seconds (1994) (1.0)
Aparajito (1956) (1.0)
Scarlet Letter, The (1995) (0.995870594886)
```

It sorts them by reverse alphabetical order, except for the first 3. This is because the first 3 do not actually have a calculated Pearson's score of 1.0.

If I change the code to just `pprint.pprint(result['Top Gun (1986)'])`, then I can see the actual values stored in the resulting dictionary, producing output like so:

```
(1.0000000000000027, 'Shiloh (1997)'),
(1.0000000000000027, 'King of the Hill (1993)'),
(1.0000000000000007, 'Bhaji on the Beach (1993)'),
(1.0, 'Wild America (1997)'),
(1.0, 'Wedding Gift, The (1994)'),
(1.0, 'Underground (1995)'),
```

Invoking the `str` function in order to convert the float for printing causes Python to convert its internal representation of values such as 1.0000000000000027 into 1.0.

To get the ratings least like a given movie, the script is run like so:

```
./getFilmsLike.py 'Top Gun (1986)' 32 'least'
```

Which produces the following output consisting of movie title followed by Pearson score in parentheses:

```
100 / 1664
200 / 1664
300 / 1664
400 / 1664
500 / 1664
600 / 1664
700 / 1664
800 / 1664
900 / 1664
1000 / 1664
1100 / 1664
1200 / 1664
1300 / 1664
1400 / 1664
1500 / 1664
1600 / 1664
Movies least like 'Top Gun (1986)': '
Babysitter, The (1995) (-1.0)
Telling Lies in America (1997) (-1.0)
Bad Moon (1996) (-1.0)
Beat the Devil (1954) (-1.0)
Bewegte Mann, Der (1994) (-1.0)
Bitter Sugar (Azucar Amargo) (1996) (-1.0)
Broken English (1996) (-1.0)
Caro Diario (Dear Diary) (1994) (-1.0)
Carpool (1996) (-1.0)
Carried Away (1996) (-1.0)
Everest (1998) (-1.0)
Frisk (1995) (-1.0)
Heidi Fleiss: Hollywood Madam (1995) (-1.0)
Joy Luck Club, The (1993) (-1.0)
Lamerica (1994) (-1.0)
Loch Ness (1995) (-1.0)
Love and Death on Long Island (1997) (-1.0)
Lover's Knot (1996) (-1.0)
Meet Wally Sparks (1997) (-1.0)
Midnight Dancers (Sibak) (1994) (-1.0)
Naked in New York (1994) (-1.0)
Nico Icon (1995) (-1.0)
Nil By Mouth (1997) (-1.0)
Romper Stomper (1992) (-1.0)
Roseanna's Grave (For Roseanna) (1997) (-1.0)
Safe Passage (1994) (-1.0)
Switchback (1997) (-1.0)
Tetsuo II: Body Hammer (1992) (-1.0)
Two Much (1996) (-1.0)
World of Apu, The (Apur Sansar) (1959) (-1.0)
Year of the Horse (1997) (-1.0)
```

```
Alphaville (1965) (-0.946729262406)
```

Just like the with *most like*, we see this output in alphabetical order, except for the first two items. The first two items, 'Babysitter, The (1995)' and 'Telling Lies in America (1997)', do not actually have values of -1.0. Again, using `pprint.pprint` on the dictionary returned gives values like so:

```
(-1.0000000000000007, 'Babysitter, The (1995)'),
(-1.0000000000000004, 'Telling Lies in America (1997)')
```

Again, the `str` function annoyingly converts the values to -1.0.

6

Question

6. Which 5 raters rated the most films? Show the raters' IDs and the number of films each rated.

Answer

Listing 6 on page 25 computes the list of raters and the number of films they rated.

It is run like so:

```
./ratedMost.py ../data/u.data 5
```

And returns output like so, consisting of rater ID followed by number of films:

```
405     737
655     685
13      636
450     540
276     518
```

As we see, rater 405 comes in on top with 737 ratings.

7

Question

7. Which 5 raters most agreed with each other? Show the raters' IDs and Pearson's r, sorted by r.

Answer

Listing 7 on page 26 attempts to answer this question.

8

Question

8. Which 5 raters most disagreed with each other (negative correlation)? Show the raters' IDs and Pearson's r, sorted by r.

Answer

Listing 7 on page 26 attempts to answer this question.

9

Question

9. What movie was rated highest on average by men over 40? By men under 40?

Answer

Listing 8 on page 28 shows the source for calculating the movie that was rated highest on average by a given gender, given a pivot age, and a direction.

To calculate the movies rated highest on average by men over 40:

```
./highestbygenderagepivot.py 26 ../data/u.data ../data/u.item
    ../data/u.user 'M' 40 'greater'
```

The output for men over 40 looks like so:

```
Solo (1996)      5.0
Grateful Dead (1995)     5.0
Unstrung Heroes (1995)  5.0
Hearts and Minds (1996) 5.0
Two or Three Things I Know About Her (1966)      5.0
Great Day in Harlem, A (1994)    5.0
Boxing Helena (1993)     5.0
Ace Ventura: When Nature Calls (1995)    5.0
Spice World (1997)       5.0
Little City (1998)       5.0
Leading Man, The (1996) 5.0
Aparajito (1956)         5.0
World of Apu, The (Apur Sansar) (1959)  5.0
Little Princess, The (1939)      5.0
Late Bloomers (1996)     5.0
Indian Summer (1996)     5.0
Star Kid (1997) 5.0
Poison Ivy II (1995)     5.0
Marlene Dietrich: Shadow and Light (1996)        5.0
Strawberry and Chocolate (Fresa y chocolate) (1993)      5.0
Prefontaine (1997)       5.0
Rendezvous in Paris (Rendez-vous de Paris, Les) (1995)  5.0
They Made Me a Criminal (1939)  5.0
Faithful (1996) 5.0
Double Happiness (1994) 5.0
Pather Panchali (1955)  4.8
```

which shows that men over 40 agree that 25 movies in this set deserve a 5.0 on average.

To calculate the movies rated highest on average by men under 40:

```
./highestbygenderagepivot.py 19 ../data/u.data ../data/u.item
    ../data/u.user 'M' 40 'less'
```

```
Perfect Candidate, A (1996)      5.0
Entertaining Angels: The Dorothy Day Story (1996)        5.0
Angel Baby (1995)        5.0
Leading Man, The (1996)  5.0
Love Serenade (1996)     5.0
Magic Hour, The (1998)   5.0
Aiqing wansui (1994)     5.0
Santa with Muscles (1996)        5.0
Saint of Fort Washington, The (1993)     5.0
Delta of Venus (1994)    5.0
Star Kid (1997) 5.0
Love in the Afternoon (1957)     5.0
Letter From Death Row, A (1998)  5.0
Maya Lin: A Strong Clear Vision (1994)   5.0
Prefontaine (1997)       5.0
Hugo Pool (1997)         5.0
Quiet Room, The (1996)   5.0
Crossfire (1947)         5.0
Winter Guest, The (1997)         4.5
```

which shows that men under 40 agree that 18 movies in this set deserve a 5.0 on average.

10

Question

10. What movie was rated highest on average by women over 40? By women under 40?

Answer

Listing 8 on page 28 shows the source for calculating the movie that was rated highest on average by a given gender, given a pivot age, and a direction.

To calculate the movies rated highest on average by women over 40:

```
./highestbygenderagepivot.py 27 ../data/u.data ../data/u.item
    ../data/u.user 'F' 40 'greater'
```

The output for this run of the program is shown below:

```
Funny Face (1957)        5.0
Nightmare Before Christmas, The (1993)  5.0
Ma vie en rose (My Life in Pink) (1997) 5.0
In the Bleak Midwinter (1995)    5.0
Bride of Frankenstein (1935)     5.0
Mary Shelley's Frankenstein (1994)       5.0
Pocahontas (1995)        5.0
Great Dictator, The (1940)       5.0
Tombstone (1993)         5.0
Grand Day Out, A (1992) 5.0
Wrong Trousers, The (1993)       5.0
Angel Baby (1995)        5.0
Gold Diggers: The Secret of Bear Mountain (1995)         5.0
Visitors, The (Visiteurs, Les) (1993)    5.0
Foreign Correspondent (1940)     5.0
Swept from the Sea (1997)        5.0
Mina Tannenbaum (1994)  5.0
Band Wagon, The (1953)  5.0
Shall We Dance? (1937)  5.0
Top Hat (1935)  5.0
Letter From Death Row, A (1998) 5.0
Best Men (1997) 5.0
Safe (1995)      5.0
Shallow Grave (1994)     5.0
Balto (1995)     5.0
Quest, The (1996)        5.0
Once Were Warriors (1994)        4.8
```

which shows that women over 40 agree that 26 movies in this set deserve a 5.0 on average.

To calculate the movies rated highest on average by women under 40:

```
./highestbygenderagepivot.py 18 ../data/u.data ../data/u.item
    ../data/u.user 'F' 40 'less'
```

The output for this run of the program is shown below:

```
Heaven's Prisoners (1996)        5.0
Year of the Horse (1997)         5.0
Telling Lies in America (1997)  5.0
Faster Pussycat! Kill! Kill! (1965)      5.0
Nico Icon (1995)         5.0
Someone Else's America (1995)    5.0
Everest (1998)  5.0
Wedding Gift, The (1994)         5.0
Grace of My Heart (1996)         5.0
Mina Tannenbaum (1994)  5.0
Stripes (1981)  5.0
Maya Lin: A Strong Clear Vision (1994)  5.0
Prefontaine (1997)       5.0
Backbeat (1993) 5.0
Horseman on the Roof, The (Hussard sur le toit, Le) (1995)
    5.0
Umbrellas of Cherbourg, The (Parapluies de Cherbourg, Les)
    (1964)        5.0
Don't Be a Menace to South Central While Drinking Your Juice in
    the Hood (1996)5.0
Wallace & Gromit: The Best of Aardman Animation (1996)
    4.81818181818
```

which shows that women under 40 agree that 17 movies in this set deserve a 5.0 on average.

A Source for Question 1

```
1  #!/usr/local/bin/python3
2
3  import sys
4  import numpy
5  import codecs
6
7  def getRatingsFromFile(ratingsfile):
8
9      ratingsdict = {}
10
11     f = open(ratingsfile)
12
13     for line in f:
14         (user_id, item_id, rating, timestamp) = line.split('\t')
15
16         # deal with new items
17         if item_id not in ratingsdict:
18             ratingsdict[item_id] = []
19
20         ratingsdict[item_id].append(int(rating))
21
22     f.close()
23
24     return ratingsdict
25
26 def getMovieNames(namesfile):
27
28     namesdict = {}
29
30     f = codecs.open(namesfile, 'r', 'iso-8859-1')
31
32     for line in f:
33         (id, name) = line.split('|')[0:2]
34         namesdict[id] = name
35
36     f.close()
37
38     return namesdict
39
40 def getAverageRatings(ratingsdict):
41
42     averagelist = []
43
44     for key in ratingsdict:
45         averagelist.append( ( numpy.mean(ratingsdict[key]), key
               ) )
```

```
46
47      return sorted(averagelist, reverse=True)
48
49  def getTopN(averagelist, n):
50
51      return averagelist[0:n]
52
53  if __name__ == '__main__':
54      topratingsCount = int(sys.argv[1])
55      ratingsfile = sys.argv[2]
56      namesfile = sys.argv[3]
57
58      ratingsdict = getRatingsFromFile(ratingsfile)
59      averagelist = getAverageRatings(ratingsdict)
60      topN = getTopN(averagelist, topratingsCount)
61
62      namesdict = getMovieNames(namesfile)
63
64      for i in topN:
65          print(namesdict[i[1]] + '\t' + str(i[0]))
```

Listing 2: highestratings.py source, listing the movies with the highest average ratings

A Source for Question 2

```
1  #!/usr/local/bin/python3
2
3  import sys
4  import codecs
5
6  def getRatingsFromFile(ratingsfile):
7
8      ratingsdict = {}
9
10     f = open(ratingsfile)
11
12     for line in f:
13         (user_id, item_id, rating, timestamp) = line.split('\t')
14
15         # deal with new items
16         if item_id not in ratingsdict:
17             ratingsdict[item_id] = []
18
19         ratingsdict[item_id].append(int(rating))
20
21     f.close()
22
23     return ratingsdict
24
25 def getMovieNames(namesfile):
26
27     namesdict = {}
28
29     f = codecs.open(namesfile, 'r', 'iso-8859-1')
30
31     for line in f:
32         (id, name) = line.split('|')[0:2]
33         namesdict[id] = name
34
35     f.close()
36
37     return namesdict
38
39 def getRatingsCount(ratingsdict):
40
41     countlist = []
42
43     for key in ratingsdict:
44         countlist.append( ( len(ratingsdict[key]), key ) )
45
46     return sorted(countlist, reverse=True)
```

```
47
48 def getTopN(countlist, n):
49
50     return countlist[0:n]
51
52 if __name__ == '__main__':
53     topratingsCount = int(sys.argv[1])
54     ratingsfile = sys.argv[2]
55     namesfile = sys.argv[3]
56
57     ratingsdict = getRatingsFromFile(ratingsfile)
58     averagelist = getRatingsCount(ratingsdict)
59     topN = getTopN(averagelist, topratingsCount)
60
61     namesdict = getMovieNames(namesfile)
62
63     for i in topN:
64         print(namesdict[i[1]] + '\t' + str(i[0]))
```

Listing 3: mostratings.py source, listing the movies with the most ratings

A Source for Questions 3 and 4

```
1  #!/usr/local/bin/python3
2
3  import sys
4  import numpy
5  import codecs
6
7  def getUsersByGender(userfile, selectedGender):
8
9      f = open(userfile)
10
11     userdict = {}
12
13     for line in f:
14         (userid, age, gender) = line.split('|')[0:3]
15
16         if gender == selectedGender:
17             userdict[userid] = age
18
19     f.close()
20
21     return userdict
22
23
24 def getRatingsFromFileForUsers(ratingsfile, userlist):
25
26     ratingsdict = {}
27
28     f = open(ratingsfile)
29
30     for line in f:
31         (user_id, item_id, rating, timestamp) = line.split('\t')
32
33         if user_id in userlist:
34             # deal with new items
35             if item_id not in ratingsdict:
36                 ratingsdict[item_id] = []
37
38             ratingsdict[item_id].append(int(rating))
39
40     f.close()
41
42     return ratingsdict
43
44 def getMovieNames(namesfile):
45
46     namesdict = {}
```

```
47
48      f = codecs.open(namesfile, 'r', 'iso-8859-1')
49
50      for line in f:
51          (id, name) = line.split('|')[0:2]
52          namesdict[id] = name
53
54      f.close()
55
56      return namesdict
57
58  def getAverageRatings(ratingsdict):
59
60      averagelist = []
61
62      for key in ratingsdict:
63          averagelist.append( ( numpy.mean(ratingsdict[key]), key
                ) )
64
65      return sorted(averagelist, reverse=True)
66
67  def getTopN(averagelist, n):
68
69      return averagelist[0:n]
70
71  if __name__ == '__main__':
72      topratingsCount = int(sys.argv[1])
73      ratingsfile = sys.argv[2]
74      namesfile = sys.argv[3]
75      userfile = sys.argv[4]
76      gender = sys.argv[5]
77
78      userlist = getUsersByGender(userfile, gender)
79      ratingsdict = getRatingsFromFileForUsers(ratingsfile,
            userlist)
80      averagelist = getAverageRatings(ratingsdict)
81      topN = getTopN(averagelist, topratingsCount)
82
83      namesdict = getMovieNames(namesfile)
84
85      for i in topN:
86          print(namesdict[i[1]] + '\t' + str(i[0]))
```

Listing 4: highestbygender.py source, listing the movies with highest ratings by the given gender

A Source for Question 5

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

import sys
import pprint

sys.path.insert(0, '../starter-code')

import recommendations

if __name__ == '__main__':

    film = sys.argv[1]
    threshold = int(sys.argv[2])
    direction = sys.argv[3]

    prefs = recommendations.loadMovieLens('../data')

    result = recommendations.calculateSimilarItems(prefs)

    if direction == 'most':
        print "Movies most like '" + film + "': '"
        for i in range(0, threshold):
            print result[film][i][1] + ' (' + str(result[film][i
                ][0]) + ')'
    else:
        print "Movies least like '" + film + "': '"
        for i in range(1, threshold):
            print result[film][-i][1] + ' (' + str(result[film
                ][-i][0]) + ')'
```

Listing 5: getFilmsLike.py source, listing the movies with ratings like the given film

A Source for Question 6

```
1  #!/usr/local/bin/python3
2
3  import sys
4  import pprint
5
6  def getRatingsFromFile(ratingsfile):
7
8      f = open(ratingsfile)
9
10     userlist = []
11
12     for line in f:
13         (user_id, item_id, rating, timestamp) = line.split('\t')
14
15         userlist.append(user_id)
16
17     return userlist
18
19
20 if __name__ == '__main__':
21
22     ratingsFile = sys.argv[1]
23     n = int(sys.argv[2])
24
25     userlist = getRatingsFromFile(ratingsFile)
26
27     users = set(userlist)
28
29     countdict = {}
30
31     for user in users:
32         countdict[user] = userlist.count(user)
33
34     for user in sorted(countdict, key=countdict.get, reverse=
           True)[0:n]:
35         print(user + '\t' + str(countdict[user]))
```

Listing 6: ratedMost.py source, listing the movies with ratings like the given film

A Source for Questions 7 and 8

```
1  #!/usr/local/bin/python
2
3  import pprint
4  import sys
5
6  sys.path.insert(0, '../starter-code')
7
8  import recommendations
9
10 def getRatingsFromFile(ratingsfile):
11
12     ratingsdict = {}
13
14     f = open(ratingsfile)
15
16     for line in f:
17         (user_id, item_id, rating, timestamp) = line.split('\t')
18
19         if user_id not in ratingsdict:
20             ratingsdict[user_id] = {}
21
22         ratingsdict[user_id][item_id] = float(rating)
23
24     f.close()
25
26     return ratingsdict
27
28 def getMovieNames(namesfile):
29
30     namesdict = {}
31
32     f = codecs.open(namesfile, 'r', 'iso-8859-1')
33
34     for line in f:
35         (id, name) = line.split('|')[0:2]
36         namesdict[id] = name
37
38     f.close()
39
40     return namesdict
41
42 if __name__ == '__main__':
43
44     ratingsfile = sys.argv[1]
45     namesfile = sys.argv[2]
46     n = int(sys.argv[3])
```

```
47      correlation = sys.argv[4]
48
49      ratingsdict = getRatingsFromFile(ratingsfile)
50
51      raters = ratingsdict.keys()
52
53      # list of tuples
54      comparedRaters = {}
55
56      # this whole thing runs in O(n^3) time, which leads it to
57      # take a minute with 100,000 on my quad-core MacBook Pro
58      for i in range(0, len(raters)): # O(n)
59
60          for j in range(0, len(raters)): # O(n)
61
62              if i != j:
63                  # strip out dupes!!!
64                  if (raters[j], raters[i]) not in comparedRaters:
65                      # it looks like sim_pearson runs in O(n)
66                      r = recommendations.sim_pearson(
67                          ratingsdict, raters[i], raters[j]
68                          )
69
70                      comparedRaters[(raters[i], raters[j])] = r
71
72      if correlation == 'agreed':
73          reversesort = True
74      else:
75          reversesort = False
76
77      for item in sorted(
78          comparedRaters, key=comparedRaters.get, reverse=
                reversesort)[0:n]:
79          print(
80              str(comparedRaters[item]) +
81                  '\t' + str(item[0]) + '\t' + str(item[1])
82                  )
83
84      pprint.pprint(comparedRaters)
```

Listing 7: ratersMostCorrelated.py source, listing the raters that are most/least in agreement with each other

A Source for Questions 9 and 10

```
1  #!/usr/local/bin/python3
2
3  import sys
4  import numpy
5  import codecs
6
7  def getUsersByGender(userfile, selectedGender):
8
9      f = open(userfile)
10
11     userdict = {}
12
13     for line in f:
14         (userid, age, gender) = line.split('|')[0:3]
15
16         if gender == selectedGender:
17             userdict[userid] = int(age)
18
19     f.close()
20
21     return userdict
22
23 def getUsersByAgeRange(userdict, pivot, direction):
24
25     newuserdict = {}
26
27     for user in userdict:
28
29         if direction == 'less':
30             # add to new dict because they're < pivot
31             if userdict[user] < pivot:
32                 newuserdict[user] = userdict[user]
33
34         if direction == 'greater':
35             # add to new dict because they're > pivot
36             if userdict[user] > pivot:
37                 newuserdict[user] = userdict[user]
38
39     return newuserdict
40
41
42 def getRatingsFromFileForUsers(ratingsfile, userlist):
43
44     ratingsdict = {}
45
46     f = open(ratingsfile)
```

```
47
48      for line in f:
49          (user_id, item_id, rating, timestamp) = line.split('\t')
50
51          if user_id in userlist:
52              # deal with new items
53              if item_id not in ratingsdict:
54                  ratingsdict[item_id] = []
55
56              ratingsdict[item_id].append(int(rating))
57
58      f.close()
59
60      return ratingsdict
61
62  def getMovieNames(namesfile):
63
64      namesdict = {}
65
66      f = codecs.open(namesfile, 'r', 'iso-8859-1')
67
68      for line in f:
69          (id, name) = line.split('|')[0:2]
70          namesdict[id] = name
71
72      f.close()
73
74      return namesdict
75
76  def getAverageRatings(ratingsdict):
77
78      averagelist = []
79
80      for key in ratingsdict:
81          averagelist.append( ( numpy.mean(ratingsdict[key]), key
                ) )
82
83      return sorted(averagelist, reverse=True)
84
85  def getTopN(averagelist, n):
86
87      return averagelist[0:n]
88
89  if __name__ == '__main__':
90      topratingsCount = int(sys.argv[1])
91      ratingsfile = sys.argv[2]
92      namesfile = sys.argv[3]
93      userfile = sys.argv[4]
94      gender = sys.argv[5]
```

```
95      agepivot = int(sys.argv[6])
96      agedirection = sys.argv[7]
97
98      userdict = getUsersByGender(userfile, gender)
99      userdict = getUsersByAgeRange(userdict, agepivot,
            agedirection)
100     ratingsdict = getRatingsFromFileForUsers(ratingsfile,
            userdict)
101     averagelist = getAverageRatings(ratingsdict)
102     topN = getTopN(averagelist, topratingsCount)
103
104     namesdict = getMovieNames(namesfile)
105
106     for i in topN:
107         print(namesdict[i[1]] + '\t' + str(i[0]))
```

Listing 8: highestbygenderagepivot.py source, listing the movies aged highest by the given gender, age pivot, and direction of pivot