Assignment 8

Fall 2014
CS595 Web Science
Dr. Michael Nelson

Mathew Chaney

November 12, 2014

Contents

Listings

1 Question 1

1.1 Question

```
The goal of this project is to use the basic recommendation principles
we have learned for user-collected data. You will modify the code
given to you which performs movie recommendations from the MovieLense
data sets.

The MovieLense data sets were collected by the GroupLens Research
Project at the University of Minnesota during the seven-month period
from September 19th, 1997 through April 22nd, 1998. It is available
for download from http://www.grouplens.org/node/73

The code for reading from the u.data and u.item files and creating
recommendations is described in the book Programming Collective
Intelligence (check email for more details). You are to modify
recommendations.py to answer the following questions. Each question
your program answers correctly will award you 10 points. You must
have the question answered completely correct; partial credit will
only be awarded if your answer is very close to the correct one.

Your output should clearly indicate the answers from the question
you answered.  Provide any relevant discussion.
```

1.2 Answer

Each question was answered by using some combination of the existing functions from `recommendations.py` and some functions that were added. All of the tables provided were created using the `tabulate` function (with some minor edits), which is shown in Listing 1. All of the code used can be found in Appendix A, which contains Listing 17.

```
236 def tabulate(tuples, caption, label, colnames, output):
237     output.write('\\begin{table}[h!]\n')
238     output.write('\\centering\n')
239     opts = '| ' + ' | '.join(['l' for i in xrange(len(tuples[0]))]) + ' |'
240     output.write('\\begin{{tabular}}{{{0}}}\n'.format(opts))
241     output.write('\\hline\n')
242     header = ' & '.join(['{}' for i in xrange(len(tuples[0]))]).format(*colnames)
243     output.write(header + ' \\\\\n\\hline\n')
244     for item in tuples:
245         temp = ' & '.join(['{}' for i in xrange(len(item))])
246         output.write(temp.format(*item) + ' \\\\\n')
247     output.write('\\hline\n\\end{tabular}\n')
248     output.write('\\caption{{{0}}}\n'.format(caption))
249     output.write('\\label{{tab:{0}}}\n'.format(label))
250     output.write('\\end{table}\n\n')
```

Listing 1: tabulate function

1. What 5 movies have the highest average ratings?

Question 1 was solved using the code in Listing 2, which utilizes the added `get_avg` and `get_top` functions, which are found in Listing 3. The `get_avg` function uses the `mean` function from the numpy python library [1].

```
259         question1 = '1. What 5 movies have the highest average ratings?'
260         averages_all = {movie: get_avg(prefs, mid) for mid, movie in movies.iteritems()}
261         sorted_avg_all = sorted(averages_all.items(), key=itemgetter(1), reverse=True)
262         top_all = get_top(sorted_avg_all)
263         tabulate(top_all, 'Question 1: Highest Average Rating', 'hiavgrat', ('Title', '
                Rating'),
264                  outfile)
265         print "done with 1"
```

Listing 2: Question 1 code

```
203 def get_avg(prefs, mid, user_filter=lambda x: True):
204     ratings = []
205     for user, user_ratings in prefs.iteritems():
206         if user_filter(users[user]) and user_ratings.has_key(movies[mid]):
207             ratings.append(user_ratings[movies[mid]])
208     if not ratings:
209         return 0.0
210     return mean(ratings)
211
212 def get_top(sorted_list, key=lambda x, i: x[i][1], n=5):
213     top = key(sorted_list, 0)
214     top_items = []
215     i = 0
216     while i < n or key(sorted_list, i) == top:
217         top_items.append(sorted_list[i])
218         if i < n and key(sorted_list, i) != top:
219             top = key(sorted_list, i)
220         i += 1
221     return top_items
```

Listing 3: get_avg and get_top functions

The results for Question 1 are shown in Table 1.

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

Table 1: Question 1: Highest Average Rating

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

Question 2 was solved using the `count_movie_ratings` and `get_top` functions. The code used is shown in Listings 4, 5 and 3.

```
267         question2 = "2. What 5 movies received the most ratings? Show the movies and the
                number of ratings sorted by number of ratings."
268         movie_counts = {movie: count_movie_ratings(prefs, mid) for mid, movie in movies.
                iteritems()}
269         sorted_counts = sorted(movie_counts.items(), key=itemgetter(1), reverse=True)
270         top_movie_counts = get_top(sorted_counts)
271         tabulate(top_movie_counts, 'Question 2: Most Ratings', 'mratings', ('Title', '
                Ratings Count'), outfile)
272         print "done with 2"
```

Listing 4: Question 2 code

```
223 def count_movie_ratings(prefs, mid, transform=False):
224     num = 0
225     for user, user_ratings in prefs.iteritems():
226         if user_ratings.has_key(movies[mid]):
227             num += 1
228     return num
```

Listing 5: count_movie_ratings function

The results for Question 2 are shown in Table 2.

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

Table 2: Question 2: Most Ratings

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

Question 3 was solved using the code in Listing 6 and the `get_avg` and `get_top` functions from Listing 3.

```
275         averages_w = {movie: get_avg(prefs, mid, user_filter=lambda x: x['gender']=='F') for
                mid, movie in movies.iteritems()}
276         sorted_avg_w = sorted(averages_w.items(), key=itemgetter(1), reverse=True)
277         top_avg_w = get_top(sorted_avg_w)
278         tabulate(top_avg_w, 'Question 3: Highest Ratings by Women', 'hwratings', ('Title', '
                Rating'), outfile)
279         print "done with 3"
```

Listing 6: question 3 code

The results of Question 3 are shown in Table 3.

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

Table 3: Question 3: Highest Ratings by Women

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

Question 4 was solved using the code in Listing 7 and the `get_avg` and `get_top` functions from Listing 3.

```
281         question4 = "4. What 5 movies were rated the highest on average by men? Show the
                movies and their ratings sorted by ratings."
282         averages_m = {movie: get_avg(prefs, mid, user_filter=lambda x: x['gender']=='M') for
                 mid, movie in movies.iteritems()}
283         sorted_avg_m = sorted(averages_m.items(), key=itemgetter(1), reverse=True)
284         top_avg_m = get_top(sorted_avg_m)
285         tabulate(top_avg_m, 'Question 4: Highest Ratings by Men', 'hmratings', ('Title', '
                Rating'), outfile)
286         print "done with 4"
```

Listing 7: question 4 code

The results of Question 4 are shown in Table 4.

Title	Rating
They Made Me a Criminal (1939)	5.0
Letter From Death Row, A (1998)	5.0
Saint of Fort Washington, The (1993)	5.0
Quiet Room, The (1996)	5.0
Entertaining Angels: The Dorothy Day Story (1996)	5.0
Marlene Dietrich: Shadow and Light (1996)	5.0
Star Kid (1997)	5.0
Little City (1998)	5.0
Aiqing wansui (1994)	5.0
Prefontaine (1997)	5.0
Love Serenade (1996)	5.0
Leading Man, The (1996)	5.0
Great Day in Harlem, A (1994)	5.0
Santa with Muscles (1996)	5.0
Delta of Venus (1994)	5.0

Table 4: Question 4: Highest Ratings by Men

5. What movie received ratings most like Top Gun?

Question 5 was solved using the code from Listing 8 and the `get_sim_ratings` function from Listing 9.

```
288         question5 = "5. What movie received ratings most like Top Gun?"
289         sim_top_gun = get_sim_ratings("Top Gun (1986)", similar=True, top_key=lambda x, i: x
                [i][0])
290         tabulate(sim_top_gun, 'Question 5: Most like Top Gun', 'mltg', ('Pearson\'s r', '
                Title'), outfile)
291         print "done with 5"
292
293         question55 = "5 cont'd. Which movie received ratings that were least like Top Gun (
                negative correlation)"
294         dissim_top_gun = get_sim_ratings("Top Gun (1986)", similar=False, top_key=lambda x,
                 i: x[i][0])
295         tabulate(dissim_top_gun, 'Question 5.5: Least like Top Gun', 'lltg', ('Pearson\'s r'
                , 'Title'), outfile)
296         print "done with 5.5"
```

Listing 8: question 5 code

```
230 def get_sim_ratings(title, similar, top_key=lambda x, i: x[i][1], n=2000):
231     itemPrefs = transformPrefs(prefs)
232     matches = topMatches(itemPrefs, title, n=n, similarity=sim_pearson)
233     sorted_m = sorted(matches, key=itemgetter(0), reverse=similar)
234     return get_top(sorted_m, key=top_key, n=20)
```

Listing 9: get_sim_ratings function

The results for Question 5 are shown in Tables 5, 6 and 7.

Pearson's r	Title
1.0	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 (CafÃľ au Lait) (1993)
1.0	Maybe, Maybe Not (Bewegte Mann, Der) (1994)
1.0	Manny & Lo (1996)

Table 5: Question 5: Most like Top Gun

Pearson's r	Title
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)

Table 6: Question 5 cont'd: Most like Top Gun

5. cont'd. Which movie received ratings that were least like Top Gun (negative correlation)

Pearson's r	Title
-1.0	Babysitter, The (1995)
-1.0	Telling Lies in America (1997)
-1.0	Year of the Horse (1997)
-1.0	World of Apu, The (Apur Sansar) (1959)
-1.0	Two Much (1996)
-1.0	Tetsuo II: Body Hammer (1992)
-1.0	Switchback (1997)
-1.0	Safe Passage (1994)
-1.0	Roseanna's Grave (For Roseanna) (1997)
-1.0	Romper Stomper (1992)
-1.0	Nil By Mouth (1997)
-1.0	Nico Icon (1995)
-1.0	Naked in New York (1994)
-1.0	Midnight Dancers (Sibak) (1994)
-1.0	Meet Wally Sparks (1997)
-1.0	Lover's Knot (1996)
-1.0	Love and Death on Long Island (1997)
-1.0	Loch Ness (1995)
-1.0	Lamerica (1994)
-1.0	Joy Luck Club, The (1993)
-1.0	Heidi Fleiss: Hollywood Madam (1995)
-1.0	Frisk (1995)
-1.0	Everest (1998)
-1.0	Carried Away (1996)
-1.0	Carpool (1996)
-1.0	Caro Diario (Dear Diary) (1994)
-1.0	Broken English (1996)
-1.0	Bitter Sugar (Azucar Amargo) (1996)
-1.0	Bewegte Mann, Der (1994)
-1.0	Beat the Devil (1954)
-1.0	Bad Moon (1996)

Table 7: Question 5.5: Least like Top Gun

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

Question 6 was solved using the code from Listing 10 and the `get_top` function shown in Listing 3.

```
298         question6 = "6. Which 5 raters rated the most films? Show the raters' IDs and the
                number of films each rated."
299         counts = {user: len(user_ratings) for user, user_ratings in prefs.iteritems()}
300         sorted_counts = sorted(counts.items(), key=itemgetter(1), reverse=True)
301         top_rater_counts = get_top(sorted_counts)
302         tabulate(top_rater_counts, 'Question 6: Most Opinionated Viewers', 'opinion', ('
                Rater ID', 'Ratings Count'), outfile)
303         print "done with 6"
```

Listing 10: question 6 code

The results of Question 6 are shown in Table 8.

Rater ID	Ratings Count
405	736
655	678
13	632
450	538
276	516

Table 8: Question 6: Most Opinionated Viewers

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

Question 7 was solved using the code shown in Listing 11, with the `calcSimilarUsers` and the `get_top` functions from Listings 12 and 3, respectively. The `flatten` function from Listing 13 was used to flatted the oddly arranged tuple that was created from the `calcSimUsers` function.

```
305         question7 = "7. Which 5 raters most agreed with each other? Show the raters' IDs and
                Pearson's r, sorted by r."
306         raters_sim = calcSimilarUsers(prefs, n=1, similarity=sim_pearson)
307         sorted_sim = sorted(raters_sim.items(), key=lambda x: x[1][0], reverse=True)
308         top_sim_raters = get_top(sorted_sim, key=lambda x,i: x[i][1][0])
309         top_sim_raters = [flatten(rater) for rater in top_sim_raters]
310         tabulate(top_sim_raters, 'Question 7: Bandwagoners', 'band', ('User 1 ID', 'User 2
                ID', 'Pearson\'s r'), outfile)
311         print "done with 7"
```

Listing 11: question 7 code

```
144 def calcSimilarUsers(prefs, n=10, similarity=sim_distance):
145     result = {}
146     itemPrefs = prefs
147     c=0
148     for item in itemPrefs:
149         c+=1
150         if c%100==0: print "%d / %d" % (c, len(itemPrefs))
151         scores = topMatches(itemPrefs, item, n=n, similarity=similarity)
152         result[item]=scores
153     return result
```

Listing 12: calcSimilarUsers function

```
254 def flatten(tup, f=lambda x: (x[0], x[1][0][1], x[1][0][0])):
255     return f(tup)
```

Listing 13: flatten function

The results for Question 7 are shown in Table 9.

User 1 ID	User 2 ID	Pearson's r
772	889	1.0
889	772	1.0
828	98	1.0
231	942	1.0
846	941	1.0

Table 9: Question 7: Bandwagoners

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

Question 8 was solved using the code from Listing 14 and the `get_top` function from Listing 3.

```
313         question8 = "8. Which 5 raters most disagreed with each other (negative correlation)
                ? Show the raters' IDs and Pearson's r, sorted by r"
314         sorted_dissim = sorted(raters_sim.items(), key=lambda x: x[1][0])
315         top_dissim_raters = get_top(sorted_dissim, key=lambda x,i: x[i][1][0])
316         top_dissim_raters = [flatten(rater) for rater in top_dissim_raters]
317         tabulate(top_dissim_raters, 'Question 8: Nemeses', 'cont', ('User 1 ID', 'User 2 ID'
                , 'Pearson\'s r'), outfile)
318         print "done with 8"
```

Listing 14: question 8 code

The results for Question 8 are shown in Table 10.

User 1 ID	User 2 ID	r Value
655	384	0.683130051064
13	46	0.687746384953
130	511	0.725423370905
327	816	0.77151674981
796	205	0.791384015377

Table 10: Question 8: Nemeses

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

Question 9 was solved using the code from Listing 15 with the `get_avg` and `get_top` functions from Listing 3.

```
320         question9 = "9.  What movie was rated highest on average by men over 40?"
321         averages_mo = {movie: get_avg(prefs, mid, user_filter=lambda x: x['gender']=='M' and
                int(x['age'])>40) for mid, movie in movies.iteritems()}
322         sorted_avg_mo = sorted(averages_mo.items(), key=itemgetter(1), reverse=True)
323         top_avg_mo = get_top(sorted_avg_mo)
324         tabulate(top_avg_mo, 'Question 9: Highest Ratings by Men aged > 40', 'hrbom', ('
                Title', 'Rating'), outfile)
325         print "done with 9"
326
327         question95 = "9 cont'd. By men under 40?"
328         averages_mu = {movie: get_avg(prefs, mid, user_filter=lambda x: x['gender']=='M' and
                int(x['age'])<40) for mid, movie in movies.iteritems()}
329         sorted_avg_mu = sorted(averages_mu.items(), key=itemgetter(1), reverse=True)
330         top_avg_mu = get_top(sorted_avg_mu)
331         tabulate(top_avg_mu, 'Question 9.5: Highest Ratings by Men aged < 40', 'hrbaom', ('
                Title', 'Rating'), outfile)
332         print "done with 9.5"
```

Listing 15: question 9 code

The results for Question 9 are shown in Tables 11 and 12.

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

Table 11: Question 9: Highest Ratings by Men aged > 40

9. cont'd. By men under 40?

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

Table 12: Question 9.5: Highest Ratings by Men aged < 40

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

Question 10 was solved using the code from Listing 16 with the `get_avg` and `get_top` functions from Listing 3.

```
334         question10 = "10. What movie was rated highest on average by women over 40?"
335         averages_wo = {movie: get_avg(prefs, mid, user_filter=lambda x: x['gender']=='F' and
                int(x['age'])>40) for mid, movie in movies.iteritems()}
336         sorted_avg_wo = sorted(averages_wo.items(), key=itemgetter(1), reverse=True)
337         top_avg_wo = get_top(sorted_avg_wo)
338         tabulate(top_avg_wo, 'Question 10: Highest Ratings by Women aged > 40', 'hrbow', ('
                Title', 'Rating'), outfile)
339         print "done with 10"
340
341         question105 = "10. cont'd. By women under 40?"
342         averages_wu = {movie: get_avg(prefs, mid, user_filter=lambda x: x['gender']=='F' and
                int(x['age'])<40) for mid, movie in movies.iteritems()}
343         sorted_avg_wu = sorted(averages_wu.items(), key=itemgetter(1), reverse=True)
344         top_avg_wu = get_top(sorted_avg_wu)
345         tabulate(top_avg_wu, 'Question 10.5: Highest Ratings by Women aged < 40', 'hrbaow',
                ('Title', 'Rating'), outfile)
346         print "done with 10.5"
```

Listing 16: question 10 code

The results for Question 10 are shown in Tables 13 and 14.

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

Table 13: Question 10: Highest Ratings by Women aged $>$ 40

10. cont'd. By women under 40?

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

Table 14: Question 10.5: Highest Ratings by Women aged < 40

2 Appendix A

```
# A dictionary of movie critics and their ratings of a small
# set of movies
critics={'Lisa Rose': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.5,
 'Just My Luck': 3.0, 'Superman Returns': 3.5, 'You, Me and Dupree': 2.5,
 'The Night Listener': 3.0},
'Gene Seymour': {'Lady in the Water': 3.0, 'Snakes on a Plane': 3.5,
 'Just My Luck': 1.5, 'Superman Returns': 5.0, 'The Night Listener': 3.0,
 'You, Me and Dupree': 3.5},
'Michael Phillips': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.0,
 'Superman Returns': 3.5, 'The Night Listener': 4.0},
'Claudia Puig': {'Snakes on a Plane': 3.5, 'Just My Luck': 3.0,
 'The Night Listener': 4.5, 'Superman Returns': 4.0,
 'You, Me and Dupree': 2.5},
'Mick LaSalle': {'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0,
 'Just My Luck': 2.0, 'Superman Returns': 3.0, 'The Night Listener': 3.0,
 'You, Me and Dupree': 2.0},
'Jack Matthews': {'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0,
 'The Night Listener': 3.0, 'Superman Returns': 5.0, 'You, Me and Dupree': 3.5},
'Toby': {'Snakes on a Plane':4.5,'You, Me and Dupree':1.0,'Superman Returns':4.0}}


from math import sqrt
from numpy import mean
from operator import itemgetter
from pprint import pprint
from sys import stdout

# Returns a distance-based similarity score for person1 and person2
def sim_distance(prefs,person1,person2):
  # Get the list of shared_items
  si={}
  for item in prefs[person1]:
    if item in prefs[person2]: si[item]=1

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

  # Add up the squares of all the differences
  sum_of_squares=sum([pow(prefs[person1][item]-prefs[person2][item],2)
                      for item in prefs[person1] if item in prefs[person2]])

  return 1/(1+sum_of_squares)

# Returns the Pearson correlation coefficient for p1 and p2
def sim_pearson(prefs, p1, p2):
    # Get the list of mutually rated items
    si={}
    for item in prefs[p1]:
        if item in prefs[p2]:
            si[item]=1

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

    # Sum calculations
    n=len(si)

    # Sums of all the preferences
    sum1=sum([prefs[p1][it] for it in si])
    sum2=sum([prefs[p2][it] for it in si])

    # Sums of the squares
    sum1Sq=sum([pow(prefs[p1][it],2) for it in si])
    sum2Sq=sum([pow(prefs[p2][it],2) for it in si])

    # Sum of the products
    pSum=sum([prefs[p1][it]*prefs[p2][it] for it in si])

    # 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
```

```
75
76      return r
77
78  # Returns the best matches for person from the prefs dictionary.
79  # Number of results and similarity function are optional params.
80  def topMatches(prefs,person,n=5,similarity=sim_pearson):
81    scores=[(similarity(prefs,person,other),other)
82                    for other in prefs if other!=person]
83    scores.sort()
84    scores.reverse()
85    return scores[0:n]
86
87  # Gets recommendations for a person by using a weighted average
88  # of every other user's rankings
89  def getRecommendations(prefs,person,similarity=sim_pearson):
90    totals={}
91    simSums={}
92    for other in prefs:
93      # don't compare me to myself
94      if other==person: continue
95      sim=similarity(prefs,person,other)
96
97      # ignore scores of zero or lower
98      if sim<=0: continue
99      for item in prefs[other]:
100
101       # only score movies I haven't seen yet
102       if item not in prefs[person] or prefs[person][item]==0:
103         # Similarity * Score
104         totals.setdefault(item,0)
105         totals[item]+=prefs[other][item]*sim
106         # Sum of similarities
107         simSums.setdefault(item,0)
108         simSums[item]+=sim
109
110   # Create the normalized list
111   rankings=[(total/simSums[item],item) for item,total in totals.items()]
112
113   # Return the sorted list
114   rankings.sort()
115   rankings.reverse()
116   return rankings
117
118 def transformPrefs(prefs):
119   result={}
120   for person in prefs:
121     for item in prefs[person]:
122       result.setdefault(item,{})
123
124       # Flip item and person
125       result[item][person]=prefs[person][item]
126   return result
127
128 def calculateSimilarItems(prefs,n=10,similarity=sim_distance):
129   # Create a dictionary of items showing which other items they
130   # are most similar to.
131   result={}
132   # Invert the preference matrix to be item-centric
133   itemPrefs=transformPrefs(prefs)
134   c=0
135   for item in itemPrefs:
136     # Status updates for large datasets
137     c+=1
138     if c%100==0: print "%d / %d" % (c,len(itemPrefs))
139     # Find the most similar items to this one
140     scores=topMatches(itemPrefs,item,n=n,similarity=similarity)
141     result[item]=scores
142   return result
143
144 def calcSimilarUsers(prefs, n=10, similarity=sim_distance):
145     result = {}
146     itemPrefs = prefs
147     c=0
148     for item in itemPrefs:
149         c+=1
150         if c%100==0: print "%d / %d" % (c, len(itemPrefs))
151         scores = topMatches(itemPrefs, item, n=n, similarity=similarity)
```

```
152         result[item]=scores
153     return result
154
155 def getRecommendedItems(prefs,itemMatch,user):
156   userRatings=prefs[user]
157   scores={}
158   totalSim={}
159   # Loop over items rated by this user
160   for (item,rating) in userRatings.items( ):
161
162     # Loop over items similar to this one
163     for (similarity,item2) in itemMatch[item]:
164
165       # Ignore if this user has already rated this item
166       if item2 in userRatings: continue
167       # Weighted sum of rating times similarity
168       scores.setdefault(item2,0)
169       scores[item2]+=similarity*rating
170       # Sum of all the similarities
171       totalSim.setdefault(item2,0)
172       totalSim[item2]+=similarity
173
174   # Divide each total score by total weighting to get an average
175   rankings=[(score/totalSim[item],item) for item,score in scores.items( )]
176
177   # Return the rankings from highest to lowest
178   rankings.sort( )
179   rankings.reverse( )
180   return rankings
181
182 def loadMovieLens():
183     # Get movie titles
184     movies={}
185     for line in open('u.item'):
186         (id,title)=line.split('|')[0:2]
187         movies[id]=title
188
189     # Load data
190     prefs={}
191     for line in open('u.data'):
192         (user,movieid,rating,ts)=line.split('\t')
193         prefs.setdefault(user,{})
194         prefs[user][movies[movieid]]=float(rating)
195
196     users={}
197     for line in open('u.user'):
198         (user, age, gender, job, zipcode) = line.split('|')
199         users.setdefault(user, {})
200         users[user] = {'age': age, 'gender': gender, 'job': job, 'zipcode': zipcode}
201     return prefs, movies, users
202
203 def get_avg(prefs, mid, user_filter=lambda x: True):
204     ratings = []
205     for user, user_ratings in prefs.iteritems():
206         if user_filter(users[user]) and user_ratings.has_key(movies[mid]):
207             ratings.append(user_ratings[movies[mid]])
208     if not ratings:
209         return 0.0
210     return mean(ratings)
211
212 def get_top(sorted_list, key=lambda x, i: x[i][1], n=5):
213     top = key(sorted_list, 0)
214     top_items = []
215     i = 0
216     while i < n or key(sorted_list, i) == top:
217         top_items.append(sorted_list[i])
218         if i < n and key(sorted_list, i) != top:
219             top = key(sorted_list, i)
220         i += 1
221     return top_items
222
223 def count_movie_ratings(prefs, mid, transform=False):
224     num = 0
225     for user, user_ratings in prefs.iteritems():
226         if user_ratings.has_key(movies[mid]):
227             num += 1
228     return num
```

```
229
230 def get_sim_ratings(title, similar, top_key=lambda x, i: x[i][1], n=2000):
231     itemPrefs = transformPrefs(prefs)
232     matches = topMatches(itemPrefs, title, n=n, similarity=sim_pearson)
233     sorted_m = sorted(matches, key=itemgetter(0), reverse=similar)
234     return get_top(sorted_m, key=top_key, n=20)
235
236 def tabulate(tuples, caption, label, colnames, output):
237     output.write('\\begin{table}[h!]\n')
238     output.write('\\centering\n')
239     opts = '| ' + ' | '.join(['l' for i in xrange(len(tuples[0]))]) + ' |'
240     output.write('\\begin{{tabular}}{{{0}}}\n'.format(opts))
241     output.write('\\hline\n')
242     header = ' & '.join(['{}' for i in xrange(len(tuples[0]))]).format(*colnames)
243     output.write(header + ' \\\\\n\\hline\n')
244     for item in tuples:
245         temp = ' & '.join(['{}' for i in xrange(len(item))])
246         output.write(temp.format(*item) + ' \\\\\n')
247     output.write('\\hline\n\\end{tabular}\n')
248     output.write('\\caption{{{0}}}\n'.format(caption))
249     output.write('\\label{{tab:{0}}}\n'.format(label))
250     output.write('\\end{table}\n\n')
251 print "Parsing data"
252 prefs, movies, users = loadMovieLens()
253
254 def flatten(tup, f=lambda x: (x[0], x[1][0][1], x[1][0][0])):
255     return f(tup)
256
257 if __name__ == '__main__':
258     with open('output.tex', 'w') as outfile:
259         question1 = '1. What 5 movies have the highest average ratings?'
260         averages_all = {movie: get_avg(prefs, mid) for mid, movie in movies.iteritems()}
261         sorted_avg_all = sorted(averages_all.items(), key=itemgetter(1), reverse=True)
262         top_all = get_top(sorted_avg_all)
263         tabulate(top_all, 'Question 1: Highest Average Rating', 'hiavgrat', ('Title', '
                Rating'),
264                  outfile)
265         print "done with 1"
266
267         question2 = "2. What 5 movies received the most ratings? Show the movies and the
                number of ratings sorted by number of ratings."
268         movie_counts = {movie: count_movie_ratings(prefs, mid) for mid, movie in movies.
                iteritems()}
269         sorted_counts = sorted(movie_counts.items(), key=itemgetter(1), reverse=True)
270         top_movie_counts = get_top(sorted_counts)
271         tabulate(top_movie_counts, 'Question 2: Most Ratings', 'mratings', ('Title', '
                Ratings Count'), outfile)
272         print "done with 2"
273
274         question3 = "3. What 5 movies were rated the highest on average by women? Show the
                movies and their ratings sorted by ratings."
275         averages_w = {movie: get_avg(prefs, mid, user_filter=lambda x: x['gender']=='F') for
                 mid, movie in movies.iteritems()}
276         sorted_avg_w = sorted(averages_w.items(), key=itemgetter(1), reverse=True)
277         top_avg_w = get_top(sorted_avg_w)
278         tabulate(top_avg_w, 'Question 3: Highest Ratings by Women', 'hwratings', ('Title', '
                Rating'), outfile)
279         print "done with 3"
280
281         question4 = "4. What 5 movies were rated the highest on average by men? Show the
                movies and their ratings sorted by ratings."
282         averages_m = {movie: get_avg(prefs, mid, user_filter=lambda x: x['gender']=='M') for
                 mid, movie in movies.iteritems()}
283         sorted_avg_m = sorted(averages_m.items(), key=itemgetter(1), reverse=True)
284         top_avg_m = get_top(sorted_avg_m)
285         tabulate(top_avg_m, 'Question 4: Highest Ratings by Men', 'hmratings', ('Title', '
                Rating'), outfile)
286         print "done with 4"
287
288         question5 = "5. What movie received ratings most like Top Gun?"
289         sim_top_gun = get_sim_ratings("Top Gun (1986)", similar=True, top_key=lambda x, i: x
                [i][0])
290         tabulate(sim_top_gun, 'Question 5: Most like Top Gun', 'mltg', ('Pearson\'s r', '
                Title'), outfile)
291         print "done with 5"
292
```

```
293         question55 = "5 cont'd. Which movie received ratings that were least like Top Gun (
                negative correlation)"
294         dissim_top_gun = get_sim_ratings("Top Gun (1986)", similar=False, top_key=lambda x,
                i: x[i][0])
295         tabulate(dissim_top_gun, 'Question 5.5: Least like Top Gun', 'lltg', ('Pearson\'s r'
                , 'Title'), outfile)
296         print "done with 5.5"
297
298         question6 = "6. Which 5 raters rated the most films? Show the raters' IDs and the
                number of films each rated."
299         counts = {user: len(user_ratings) for user, user_ratings in prefs.iteritems()}
300         sorted_counts = sorted(counts.items(), key=itemgetter(1), reverse=True)
301         top_rater_counts = get_top(sorted_counts)
302         tabulate(top_rater_counts, 'Question 6: Most Opinionated Viewers', 'opinion', ('
                Rater ID', 'Ratings Count'), outfile)
303         print "done with 6"
304
305         question7 = "7. Which 5 raters most agreed with each other? Show the raters' IDs and
                 Pearson's r, sorted by r."
306         raters_sim = calcSimilarUsers(prefs, n=1, similarity=sim_pearson)
307         sorted_sim = sorted(raters_sim.items(), key=lambda x: x[1][0], reverse=True)
308         top_sim_raters = get_top(sorted_sim, key=lambda x,i: x[i][1][0])
309         top_sim_raters = [flatten(rater) for rater in top_sim_raters]
310         tabulate(top_sim_raters, 'Question 7: Bandwagoners', 'band', ('User 1 ID', 'User 2
                ID', 'Pearson\'s r'), outfile)
311         print "done with 7"
312
313         question8 = "8. Which 5 raters most disagreed with each other (negative correlation)
                ? Show the raters' IDs and Pearson's r, sorted by r"
314         sorted_dissim = sorted(raters_sim.items(), key=lambda x: x[1][0])
315         top_dissim_raters = get_top(sorted_dissim, key=lambda x,i: x[i][1][0])
316         top_dissim_raters = [flatten(rater) for rater in top_dissim_raters]
317         tabulate(top_dissim_raters, 'Question 8: Nemeses', 'cont', ('User 1 ID', 'User 2 ID'
                , 'Pearson\'s r'), outfile)
318         print "done with 8"
319
320         question9 = "9.  What movie was rated highest on average by men over 40?"
321         averages_mo = {movie: get_avg(prefs, mid, user_filter=lambda x: x['gender']=='M' and
                 int(x['age'])>40) for mid, movie in movies.iteritems()}
322         sorted_avg_mo = sorted(averages_mo.items(), key=itemgetter(1), reverse=True)
323         top_avg_mo = get_top(sorted_avg_mo)
324         tabulate(top_avg_mo, 'Question 9: Highest Ratings by Men aged > 40', 'hrbom', ('
                Title', 'Rating'), outfile)
325         print "done with 9"
326
327         question95 = "9 cont'd. By men under 40?"
328         averages_mu = {movie: get_avg(prefs, mid, user_filter=lambda x: x['gender']=='M' and
                 int(x['age'])<40) for mid, movie in movies.iteritems()}
329         sorted_avg_mu = sorted(averages_mu.items(), key=itemgetter(1), reverse=True)
330         top_avg_mu = get_top(sorted_avg_mu)
331         tabulate(top_avg_mu, 'Question 9.5: Highest Ratings by Men aged < 40', 'hrbaom', ('
                Title', 'Rating'), outfile)
332         print "done with 9.5"
333
334         question10 = "10. What movie was rated highest on average by women over 40?"
335         averages_wo = {movie: get_avg(prefs, mid, user_filter=lambda x: x['gender']=='F' and
                 int(x['age'])>40) for mid, movie in movies.iteritems()}
336         sorted_avg_wo = sorted(averages_wo.items(), key=itemgetter(1), reverse=True)
337         top_avg_wo = get_top(sorted_avg_wo)
338         tabulate(top_avg_wo, 'Question 10: Highest Ratings by Women aged > 40', 'hrbow', ('
                Title', 'Rating'), outfile)
339         print "done with 10"
340
341         question105 = "10. cont'd. By women under 40?"
342         averages_wu = {movie: get_avg(prefs, mid, user_filter=lambda x: x['gender']=='F' and
                 int(x['age'])<40) for mid, movie in movies.iteritems()}
343         sorted_avg_wu = sorted(averages_wu.items(), key=itemgetter(1), reverse=True)
344         top_avg_wu = get_top(sorted_avg_wu)
345         tabulate(top_avg_wu, 'Question 10.5: Highest Ratings by Women aged < 40', 'hrbaow',
                ('Title', 'Rating'), outfile)
346         print "done with 10.5"
```

Listing 17: all code used

3 References

[1] Numpy Developers. Python numpy Module. http://www.numpy.org/, 2013.