Old Dominion Univeristy

CS 495: Introduction to Web Science
Instructor: Michael L. Nelson, Ph.D
Fall 2014 4:20pm - 7:10pm R, ECSB 2120

Assignment # 8

George C. Micros UIN: 00757376

Honor Pledge

I pledge to support the Honor System of Old Dominion University. I will refrain from any form of academic dishonesty or deception, such as cheating or plagIiarism. I am aware that as a member of the academic community it is my responsibility to turn in all suspected violations of the Honor Code. I will report to a hearing if summoned.

Signed ________________________________

November 17, 2014

George C. Micros

Written Assignment 8

Fall 2014
CS 495: Introduction to Web Science
Dr. Michael Nelson

November 17, 2014

Contents

Chapter 1
Written Assignment 8

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`

There are three files which we will use:

- u.data: 100,000 ratings by 943 users on 1,682 movies. Each user has rated at least 20 movies. Users and items are numbered consecutively from 1. The data is randomly ordered. This is a tab separated list of
  user id — item id — rating — timestamp
  The time stamps are unix seconds since 1/1/1970 UTC.
  Example:
  196 242 3 881250949
  186 302 3 891717742
  22 377 1 878887116
  244 51 2 880606923
  166 346 1 886397596
  298 474 4 884182806
  115 265 2 881171488
- u.item: Information about the 1,682 movies. This is a tab separated list of
  movie id — movie title — release date — video release date — IMDb URL — unknown — Action — Adventure — Animation —Children's — Comedy — Crime — Documentary — Drama — Fantasy — Film-Noir — Horror — Musical — Mystery — Romance — Sci-Fi — Thriller — War — Western —
  The last 19 fields are the genres, a 1 indicates the movie is of that genre, a 0 indicates it is not; movies can be in several genres at once. The movie ids are the ones used in the u.data data set.
  Example:
  161—Top Gun (1986)—01-Jan-1986——http://us.imdb.com/M/title-exact?Top162—On Golden Pond (1981)—01-Jan-1981——http://us.imdb.com/M/title-exact?On163—Return of the Pink Panther, The (1974)—01-Jan-1974——http://us.imdb.com/M/title-exact?Return
- u.user: Demographic information about the users. This is a tab separated list of:
  user id — age — gender — occupation — zip code
  The user ids are the ones used in the u.data data set.
  Example:
  1—24—M—technician—85711 2—53—F—other—94043 3—23—M—writer—32067 4—24—M—technician—43537 5—33—F—other—15213

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 1 point.

The data was organized using classes for the movies, users and reviews. This helped organized the data in a way that was similar to the way it was represented in the files.

```
class movie:
  def __init__(self, data):
    data = data.strip('\n').split('|')
    self.id    = int(data[0])
    self.title  = data[1].replace(',', '')
    self.data   = data[2]
    self.vid   = data[3]
    self.url   = data[4]
    self.genre  = data[5:len(data)]
    self.scores = []
    self.cnt   = 0
    self.avg   = 0
    self.rvwrs  = []

  def avgr(self):
    if self.cnt > 0:
      self.avg = sum(self.scores)/self.cnt
    else:
      self.avg = 0

class user:
  def __init__(self,data):
    data = data.strip('\n').split('|')
    self.id    = int(data[0])
    self.age   = int(data[1])
    self.sex   = data[2]
    self.job   = data[3]
    self.zip   = data[4]
    self.film   = {}
    self.cnt   = 0

class review:
  def __init__(self, data):
    data = data.strip('\n').split('\t')
    self.user   = int(data[0])
    self.item = int(data[1])
    self.score  = int(data[2])
    self.time = int(data[3])
```

Listing 1.1: Class definitions for movies, users and reviews

Data was loaded from its respective files, parced and initialized into a new object that was appened to and list of similar objects.

```
f = open("./ml-100k/u.item")
for i in f:
  movies.append(movie(i))

f = open("./ml-100k/u.user")
for i in f:
  users.append(user(i))

f = open("./ml-100k/u.data")
for i in f:
  reviews.append(review(i))
```

Listing 1.2: Loading data from files to list of objects

1.1 Question 1

1.1.1 The Question

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

1.1.2 The Answer

THe first step to this question is assigning reviews and their scores to each movie. The reviews were read directly from the file, line by line, and the score for each review appended to a list within each movie object. Then the reviews were averaged

```
def q1(movies):
  clearScore(movies)
  for i in reviews:
    movies[i.item -1].scores.append(float(i.score))
    movies[i.item -1].cnt +=1

  for i in movies:
    i.avgr();
  hRate = sorted(movies, key=lambda x:(x.avg, x.title), reverse=True)
  f = open("q1.txt", "w")

  print "\n\tQ1: Highest Average Rating"
  for i in range(0,20):
    print '%.4f'%hRate[i].avg, hRate[i].title
    f.write("%.4f , \"%s\"\n" % (hRate[i].avg, hRate[i].title))

  f.close()
```

Listing 1.3: Python code for question 1

5.0000	They Made Me a Criminal (1939)
5.0000	Star Kid (1997)
5.0000	Someone Else's America (1995)
5.0000	Santa with Muscles (1996)
5.0000	Saint of Fort Washington The (1993)
5.0000	Prefontaine (1997)
5.0000	Marlene Dietrich: Shadow and Light (1996)
5.0000	Great Day in Harlem A (1994)
5.0000	Entertaining Angels: The Dorothy Day Story (1996)
5.0000	Aiqing wansui (1994)
4.6250	Pather Panchali (1955)
4.5000	Some Mother's Son (1996)
4.5000	Maya Lin: A Strong Clear Vision (1994)
4.5000	Everest (1998)
4.5000	Anna (1996)
4.4911	Close Shave A (1995)
4.4664	Schindler's List (1993)
4.4661	Wrong Trousers The (1993)
4.4568	Casablanca (1942)

Table 1.1: Highest Average Rating

1.2 Question 2

1.2.1 The Question

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

1.2.2 The Answer

There review infromation is read in and appeneded to each movie. There is a counter that counter the number of reviews for each movie. This could also be determined by checking the length of the review list for each movie.

```

def q2(movies):
  clearScore(movies)
  for i in reviews:
    movies[i.item -1].scores.append(float(i.score))
    movies[i.item -1].cnt +=1

  for i in movies:
    i.avgr();

  # MOST RATING
  mRate = sorted(movies, key=lambda x:x.cnt, reverse=True)
  f = open("q2.txt", "w")
  print "\n\tQ2: Most Ratings"
  for i in range(0,5):
    print mRate[i].cnt, mRate[i].title
    f.write("% d, \"%s\"\n" % (mRate[i].cnt, mRate[i].title))
  f.close()
```

Listing 1.4: Python code for question 2

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

Table 1.2: Movies with Most Ratings

1.3 Question 3

1.3.1 The Question

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

1.3.2 The Answer

The reviews were read line by line and before hey were appeneded to the respective movie there reviewer was checked to determine if she met the criterion of being a female.

```
def q3(movies, reviews, users):
  clearScore(movies)

  for i in reviews:
    if users[i.user-1].sex == "F":
      movies[i.item-1].scores.append(float(i.score))
      movies[i.item-1].cnt +=1

  for i in movies:
    i.avgr();
  wmn = sorted(movies,key=lambda x:(x.avg, x.title), reverse=True)

  f = open("q3.txt", "w")
  print "\n\tQ3: Highest Average by women"
  for i in range(0,20):
    print '%.4f'%wmn[i].avg, wmn[i].title
    f.write("%.4f, \"%s\"\n" % (wmn[i].avg, wmn[i].title))
  f.close()
```

Listing 1.5: Python code for question 3

5.0000	Year of the Horse (1997)
5.0000	Visitors The (Visiteurs Les) (1993)
5.0000	Telling Lies in America (1997)
5.0000	Stripes (1981)
5.0000	Someone Else's America (1995)
5.0000	Prefontaine (1997)
5.0000	Mina Tannenbaum (1994)
5.0000	Maya Lin: A Strong Clear Vision (1994)
5.0000	Foreign Correspondent (1940)
5.0000	Faster Pussycat! Kill! Kill! (1965)
5.0000	Everest (1998)
4.6329	Schindler's List (1993)
4.6316	Close Shave A (1995)
4.5625	Shawshank Redemption The (1994)
4.5333	Wallace & Gromit: The Best of Aardman Animation (1996)

Table 1.3: Highest Average Rating by Women

1.4 Question 4

1.4.1 The Question

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

1.4.2 The Answer

The reviews were read line by line and before hey were appeneded to the respective movie there reviewer was checked to determine if he met the criterion of being a male.

```
def q4(movies, reviews, users):
  clearScore(movies)
  for i in reviews:
    if users[i.user-1].sex == "M":
      movies[i.item-1].scores.append(float(i.score))
      movies[i.item-1].cnt +=1
  for i in movies:
    i.avgr();
  men = sorted(movies, key=lambda x:x.avg, reverse=True)
  f = open("q4.txt", "w")
  print "\n\tQ4: Highest Average by men"
  for i in range(0,30):
    print '%.4f'%men[i].avg, men[i].title
    f.write("%.4f, \"%s\"\n" % (men[i].avg, men[i].title))
  f.close()
```

Listing 1.6: Python code for question 4

5.0000	Great Day in Harlem A (1994)
5.0000	They Made Me a Criminal (1939)
5.0000	Quiet Room The (1996)
5.0000	Hugo Pool (1997)
5.0000	Prefontaine (1997)
5.0000	Letter From Death Row A (1998)
5.0000	Marlene Dietrich: Shadow and Light (1996)
5.0000	Star Kid (1997)
5.0000	Delta of Venus (1994)
5.0000	Saint of Fort Washington The (1993)
5.0000	Santa with Muscles (1996)
5.0000	Aiqing wansui (1994)
5.0000	Love Serenade (1996)
5.0000	Leading Man The (1996)
5.0000	Entertaining Angels: The Dorothy Day Story (1996)
5.0000	Little City (1998)
4.6667	Two or Three Things I Know About Her (1966)
4.6667	Little Princess The (1939)
4.6250	Pather Panchali (1955)
4.5000	A Chef in Love (1996)
4.5000	Anna (1996)
4.5000	Sliding Doors (1998)
4.5000	Grosse Fatigue (1994)
4.5000	Bitter Sugar (Azucar Amargo) (1996)
4.4734	Casablanca (1942)

Table 1.4: Highest Average Rating by Men

1.5 Question 6

1.5.1 The Question

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

1.5.2 The Answer

The reviews were read line by line and appened to each user. The users were then sorted based on number of reviews. The top five were taken from the sorted list.

```
def q6(users):
  for i in reviews:
    users[i.user-1].cnt += 1

  usr = sorted(users,key=lambda x:x.cnt, reverse=True)

  #
  f = open("q6.txt", "w")
  print "\n\tQ6: Rated most pics"
  print  "cnt, id"
  for i in range(0,5):
    print  usr[i].cnt, usr[i].id
    f.write("%d,  %d\n" % (usr[i].cnt, usr[i].id))
  f.close()
```

Listing 1.7: Python code for question 6

737	405
685	655
636	13
540	450
518	276

Table 1.5: Raters with the Most Reviews

1.6 Question 7

1.6.1 The Question

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

1.6.2 The Answer

In order to determine the cluster of 5 most agreed reviewers all permutations of reviewers would have to be considered and some metric of correlation amongst them be computed. For the set of all permutations we wuld use k choose n, where k = 5 and n = 943

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$
$$\binom{943}{5} = \frac{943!}{5!\,(9435)!} = 6.14843263080310^{12}$$

which is really big and very impractical to have to go through that many iterations.
The alternative is to use a greedy algorithm that is assumed to converge near the local minimum of this problem. By computing the nearest revierwer for each reviewer we can reduce the number of operations and produce a relationship with each reviewer.

p3,p9
p9,p23
p23,p25
p25,p33
p33,p77

Once these relationship have been computed a chain of five uniques reviewers is produced. Many of these chains are degenerate, pointing to the same reviewers and repeating.

p3,p9
p9,p3
p3,p9
p9,p3
p3,p9

These can be eliminated easily. From the remaining chains a metric must be devised based on the weighed connections to determine the most agreed 5. The metric chosen was the sum of the correlations of every member to the remaining memebers. The higher the sum the most closely correlated the memebrs are to each other. The group with the highest sum was chosen and the correlation between users shown to be 1.

```python
def q7():
    prefs =  loadMovieLens(path='./ml-100k')
    c = greedy(prefs, True)
    x = maths(c, prefs)

    x.sort(key=lambda x:x[5], reverse=True)
    y = []
    for i in x:
        if len(i) == len(set(i)):
            y.append(i)

    print "Most Agreed Reviewers"
    for i in range(0,5):
        print y[i]
```

Listing 1.8: Python code for question 7

```
def greedy(prefs, best):
    c = [[]]*(len(prefs))
    for i in prefs:
        e = []
        for j in prefs:
            d = []
            num = sim_pearson(prefs,i,j);
            d.append(num)
            d.append(i)
            d.append(j)
            if (i != j):
                e.append(d)
        e.sort(key=lambda x:x[:][0], reverse=best)
        c[int(e[0][1])-1] = e[0]
    return c
```

Listing 1.9: Greddy Algorithm for determining most correlated reviewer

```
def maths(c, prefs):
    d = []
    for i in c:
        e = []
        e.append(i[1])
        e.append(i[2])
        for j in range(2,5):
            e.append(c[int(e[j-1])-1][2])
        tSum = 0
        for j in range (0,5):
            for k in range (0,5):
                tSum += sim_pearson(prefs, e[j], e[k])
        e.append(tSum)
        d.append(e)
    return d
```

Listing 1.10: Determining chain of 5 reviewers from correlated

```
def q7():
    prefs = loadMovieLens(path='./ml-100k')
    c = greedy(prefs, True)
    x = maths(c, prefs)

    x.sort(key=lambda x:x[5], reverse=True)
    y = []
    for i in x:
        if len(i) == len(set(i)):
            y.append(i)

    print "Most Agreed Reviewers"
    for i in range(0,5):
        print y[i]
```

Listing 1.11: Python code for question 7

```
def sim_pearson(prefs, p1, p2):
    '''
    Returns the Pearson correlation coefficient for p1 and 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
    return r
```

Listing 1.12: Correlation between two reviewers

Most Agreed Reviewers
656−> 909 r = 1.00
909−> 869 r = 1.00
869−> 857 r = 1.00
857−> 748 r = 1.00

Highly Agreed Groups
[’656’, ’909’, ’869’, ’857’, ’748’, 21.743243005407813]
[’520’, ’694’, ’306’, ’188’, ’732’, 20.35143510753761]
[’806’, ’909’, ’869’, ’857’, ’748’, 20.28283545995625]
[’249’, ’909’, ’869’, ’857’, ’748’, 19.98780466540942]
[’251’, ’909’, ’869’, ’857’, ’748’, 19.714669425731906]

1.7 Question 8

1.7.1 The Question

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

1.7.2 The Answer

The solution for this question is similar to that of question 7. However the main different being that the goal is to find the users that disagree the most with each other, being least correlated. The main difference in the coe implementing this solution is the order of sorting. The sorting was reverese from question 7, so that the least correlated naighbors are chosen from the greedy algorithm. The least correlated group was chosen using the same metric as previously.

```
def q8():
    prefs =  loadMovieLens(path='./ml-100k')
    c = greedy(prefs, False)
    x = maths(c, prefs)

    x.sort(key=lambda x:x[5], reverse=False)
    y = []
    for i in x:
        if len(i) == len(set(i)):
            y.append(i)
    print "Least Agreed Reviewers"
    for i in range(0,5):
        print y[i]
```

Listing 1.13: Python code for question 8

```
def greedy(prefs, best):
    c = [[]]*(len(prefs))
    for i in prefs:
        e = []
        for j in prefs:
            d = []
            num = sim_pearson(prefs,i,j);
            d.append(num)
            d.append(i)
            d.append(j)
            if (i != j):
                e.append(d)
        e.sort(key=lambda x:x[:][0], reverse=best)
        c[int(e[0][1])-1] = e[0]
    return c
```

Listing 1.14: Greddy Algorithm for determining most correlated reviewer

```
def maths(c, prefs):
    d = []
    for i in c:
        e = []
        e.append(i[1])
        e.append(i[2])
        for j in range(2,5):
            e.append(c[int(e[j-1])-1][2])
        tSum = 0
```

```
        for j in range (0,5):
            for k in range (0,5):
                tSum +=  sim_pearson(prefs, e[j], e[k])
        e.append(tSum)
        d.append(e)
    return d
```

Listing 1.15: Determining chain of 5 reviewers from correlated

```
def sim_pearson(prefs, p1, p2):
    '''
    Returns the Pearson correlation coefficient for p1 and 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
    return r
```

Listing 1.16: Correlation between two reviewers

Least Agreed Reviewers
853− > 336 r = -1.00
336− > 414 r = -1.00
414− > 641 r = -1.00
641− > 134 r = -1.00

Highly Disagreed Groups
['853', '336', '414', '641', '134', -8.178267558914557]
['359', '760', '928', '432', '180', -5.94077103218698]
['895', '760', '928', '432', '180', -5.847371703822146]
['425', '477', '811', '441', '475', -5.381056081661583]
['67', '698', '898', '599', '19', -5.246363309059925]

1.8 Question 9

1.8.1 The Question

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

1.8.2 The Answer

This quesion is similar to question 4 where the highst rated movies by men were requested. This question adds the additional condition for age of the reviewer. The reviews were appened to each movie based on the conditions and then avergaed.

```
def q9a(movies, reviews, users):
  clearScore(movies)
  for i in reviews:
    if (users[i.user-1].sex == "M") and (users[i.user-1].age > 40):
      movies[i.item-1].scores.append(float(i.score))
      movies[i.item-1].cnt +=1
  for i in movies:
    i.avgr();
  m40 = sorted(movies,key=lambda x:(x.avg, x.title), reverse=True)
  f = open("q9a.txt", "w")
  print "\n\tQ9a: Highest Average by men over 40"
  for i in range(0,30):
    print '%.4f'%m40[i].avg, m40[i].title
    f.write("%.4f, \"%s\"\n" % (m40[i].avg, m40[i].title))
  f.close()
```

Listing 1.17: Python code to for men over 40

```
def q9b(movies, reviews, users):
  clearScore(movies)
  for i in reviews:
    if (users[i.user-1].sex == "M") and (users[i.user-1].age < 40):
      movies[i.item-1].scores.append(float(i.score))
      movies[i.item-1].cnt +=1
  for i in movies:
    i.avgr();
  m30 = sorted(movies,key=lambda x:(x.avg, x.title), reverse=True)
  f = open("q9b.txt", "w")
  print "\n\tQ9b: Highest Average by men under 40"
  for i in range(0,40):
    print '%.4f'%m30[i].avg, m30[i].title
    f.write("%.4f, \"%s\"\n" % (m30[i].avg, m30[i].title ))
  f.close()
```

Listing 1.18: Python code to for men under 40

5.0000	World of Apu The (Apur Sansar) (1959)
5.0000	Unstrung Heroes (1995)
5.0000	Two or Three Things I Know About Her (1966)
5.0000	They Made Me a Criminal (1939)
5.0000	Strawberry and Chocolate (Fresa y chocolate) (1993)
5.0000	Star Kid (1997)
5.0000	Spice World (1997)
5.0000	Solo (1996)
5.0000	Rendezvous in Paris (Rendez-vous de Paris Les) (1995)
5.0000	Prefontaine (1997)
5.0000	Poison Ivy II (1995)
5.0000	Marlene Dietrich: Shadow and Light (1996)
5.0000	Little Princess The (1939)
5.0000	Little City (1998)
5.0000	Leading Man The (1996)
5.0000	Late Bloomers (1996)
5.0000	Indian Summer (1996)
5.0000	Hearts and Minds (1996)
5.0000	Great Day in Harlem A (1994)
5.0000	Grateful Dead (1995)
5.0000	Faithful (1996)
5.0000	Double Happiness (1994)
5.0000	Boxing Helena (1993)
5.0000	Aparajito (1956)
5.0000	Ace Ventura: When Nature Calls (1995)
4.8000	Pather Panchali (1955)
4.6667	Whole Wide World The (1996)
4.6667	A Chef in Love (1996)
4.6471	Close Shave A (1995)
4.6000	Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)

Table 1.6: Highest Average Rating by Men over 40

5.0000	Star Kid (1997)
5.0000	Santa with Muscles (1996)
5.0000	Saint of Fort Washington The (1993)
5.0000	Quiet Room The (1996)
5.0000	Prefontaine (1997)
5.0000	Perfect Candidate A (1996)
5.0000	Maya Lin: A Strong Clear Vision (1994)
5.0000	Magic Hour The (1998)
5.0000	Love in the Afternoon (1957)
5.0000	Love Serenade (1996)
5.0000	Letter From Death Row A (1998)
5.0000	Leading Man The (1996)
5.0000	Hugo Pool (1997)
5.0000	Entertaining Angels: The Dorothy Day Story (1996)
5.0000	Delta of Venus (1994)
5.0000	Crossfire (1947)
5.0000	Angel Baby (1995)
5.0000	Aiqing wansui (1994)
4.5000	Winter Guest The (1997)
4.5000	Two or Three Things I Know About Her (1966)
4.5000	Sum of Us The (1994)
4.5000	Sliding Doors (1998)
4.5000	Man of No Importance A (1994)
4.5000	Little Princess The (1939)
4.5000	Innocents The (1961)
4.5000	Grosse Fatigue (1994)
4.5000	Fille seule La (A Single Girl) (1995)
4.5000	Boy's Life 2 (1997)
4.5000	Anna (1996)
4.4762	Wallace & Gromit: The Best of Aardman Animation (1996)
4.4754	Casablanca (1942)
4.4706	Paths of Glory (1957)

Table 1.7: Highest Average Rating by Men under 40

1.9 Question 10

1.9.1 The Question

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

1.9.2 The Answer

This quesion is similar to question 3 where the highst rated movies by women were requested. This question adds the additional condition for age of the reviewer. The reviews were appened to each movie based on the conditions and then avergaed.

```
def q10a(movies, reviews, users):
  clearScore(movies)
  for i in reviews:
    if (users[i.user-1].sex == "F") and (users[i.user-1].age > 40):
      movies[i.item-1].scores.append(float(i.score))
      movies[i.item-1].cnt +=1
  for i in movies:
    i.avgr();
  w40 = sorted(movies,key=lambda x:(x.avg, x.title), reverse=True)
  f = open("q10a.txt", "w")
  print "\n\tQ10a: Highest Average by women over 40"
  for i in range(0,40):
    print  '%.4f'%w40[i].avg, w40[i].title
    f.write("%.4f,  \"%s\"\n" % (w40[i].avg, w40[i].title))
  f.close()
```

Listing 1.19: Python code to for women over 40

```
def q10b(rmovies, reviews, users):
  clearScore(movies)
  for i in reviews:
    if (users[i.user-1].sex == "F") and (users[i.user-1].age < 40):
      movies[i.item-1].scores.append(float(i.score))
      movies[i.item-1].cnt +=1
  for i in movies:
    i.avgr();
  w30 = sorted(movies,key=lambda x:(x.avg, x.title), reverse=True)
  f = open("q10b.txt", "w")
  print "\n\tQ10b: Highest Average by women under 40"
  for i in range(0,40):
    print  '%.4f'%w30[i].avg, w30[i].title
    f.write("%.4f,  \"%s\"\n" % (w30[i].avg, w30[i].title))
  f.close()
```

Listing 1.20: Python code to for women under 40

5.0000	Wrong Trousers The (1993)
5.0000	Visitors The (Visiteurs Les) (1993)
5.0000	Top Hat (1935)
5.0000	Tombstone (1993)
5.0000	Swept from the Sea (1997)
5.0000	Shallow Grave (1994)
5.0000	Shall We Dance? (1937)
5.0000	Safe (1995)
5.0000	Quest The (1996)
5.0000	Pocahontas (1995)
5.0000	Nightmare Before Christmas The (1993)
5.0000	Mina Tannenbaum (1994)
5.0000	Mary Shelley's Frankenstein (1994)
5.0000	Ma vie en rose (My Life in Pink) (1997)
5.0000	Letter From Death Row A (1998)
5.0000	In the Bleak Midwinter (1995)
5.0000	Great Dictator The (1940)
5.0000	Grand Day Out A (1992)
5.0000	Gold Diggers: The Secret of Bear Mountain (1995)
5.0000	Funny Face (1957)
5.0000	Foreign Correspondent (1940)
5.0000	Bride of Frankenstein (1935)
5.0000	Best Men (1997)
5.0000	Band Wagon The (1953)
5.0000	Balto (1995)
5.0000	Angel Baby (1995)
4.8000	Once Were Warriors (1994)
4.7000	Fantasia (1940)
4.6667	Last of the Mohicans The (1992)
4.5714	Christmas Carol A (1938)

Table 1.8: Highest Average Rating by Women over 40

5.0000	Year of the Horse (1997)
5.0000	Wedding Gift The (1994)
5.0000	Umbrellas of Cherbourg The (Parapluies de Cherbourg Les) (1964)
5.0000	Telling Lies in America (1997)
5.0000	Stripes (1981)
5.0000	Someone Else's America (1995)
5.0000	Prefontaine (1997)
5.0000	Nico Icon (1995)
5.0000	Mina Tannenbaum (1994)
5.0000	Maya Lin: A Strong Clear Vision (1994)
5.0000	Horseman on the Roof The (Hussard sur le toit Le) (1995)
5.0000	Heaven's Prisoners (1996)
5.0000	Grace of My Heart (1996)
5.0000	Faster Pussycat! Kill! Kill! (1965)
5.0000	Everest (1998)
5.0000	Don't Be a Menace to South Central While Drinking Your Juice in the Hood (1996)
5.0000	Backbeat (1993)
4.8182	Wallace & Gromit: The Best of Aardman Animation (1996)
4.8000	Paradise Lost: The Child Murders at Robin Hood Hills (1996)
4.8000	Anne Frank Remembered (1995)
4.7021	Shawshank Redemption The (1994)
4.7000	Shall We Dance? (1996)

Table 1.9: Highest Average Rating by Women under 40

References

1. Toby Segaran. *Programming collective intelligence: building smart web 2.0 applications.* " O'Reilly Media, Inc.", 2007.
2. Maja J Mataric. Designing emergent behaviors: From local interactions to collective intelligence. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 432–441, 1993.
3. J Riedl and J Konstan. Movielens dataset, 1998.
4. grouplens. MovieLens Data. `http://grouplens.org/datasets/movielens/`.
5. K. Arthur Endsley. Programming Collective Intelligence Code. `https://github.com/cataska/programming-collective-intelligence-code/tree/master/chapter2`.
6. Bradley N Miller, Istvan Albert, Shyong K Lam, Joseph A Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266. ACM, 2003.
7. RA Whitaker. A fast algorithm for the greedy interchange for large-scale clustering and median location-problems. *Infor*, 21(2):95–108, 1983.