CS 595: Assignment #8

Due on Thursday, November 13, 2014

Dr. Nelson 4:20pm

Holly Harkins

Contents

Problem 1

```
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:
1.  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.

2.  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.

3.  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.

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.

Answers:
To do this assignment, I modified recommendations.py.  I added the
functionality for each question to the loadMovieLens() function.
Assignment8.py uses the data from u.data, u.item, and u.user to output
the results.  I had some trouble with question 7.  I do not believe 4
user1 agreed most with the same user 98.  I also used the tabular view
for the first time.  My code highlights how I answered the questions.
It is listed at the end of the document.
```

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

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

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

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

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

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

Problem 1 continued on next page. . .

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

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

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

Most Like Top Gun : 1.0, 'Wild America (1997)'
Least Like Top Gun: -1.0, 'Babysitter, The (1995)'

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

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

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

User1 ID	User2 ID	Pearson's r
675	99	1.0
156	98	1.0
772	98	1.0
226	98	1.0
227	98	1.0

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

User1 ID	User2 ID	r Value
655	384	0.683
13	46	0.688
130	511	0.725
327	816	0.772
796	205	0.791

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

Title	Rating
Delta of Venus (1994)	5.0
Santa with Muscles (1996)	5.0
Crossfire (1947)	5.0
Leading Man, The (1996)	5.0
Love Serenade (1996)	5.0
Prefontaine (1997)	5.0
Aiqing wansui (1994)	5.0
Love in the Afternoon (1957)	5.0
Star Kid (1997)	5.0
Angel Baby (1995)	5.0
Maya Lin: A Strong Clear Vision (1994)	5.0
Entertaining Angels: The Dorothy Day Story (1996)	5.0
Magic Hour, The (1998)	5.0
Quiet Room, The (1996)	5.0
Saint of Fort Washington, The (1993)	5.0
Perfect Candidate, A (1996)	5.0
Letter From Death Row, A (1998)	5.0
Little Princess, The (1939)	4.5
Grosse Fatigue (1994)	4.5
Sum of Us, The (1994)	4.5
Boy's Life 2 (1997)	4.5
Fille seule, La (A Single Girl) (1995)	4.5
Winter Guest, The (1997)	4.5
Man of No Importance, A (1994)	4.5
Anna (1996)	4.5
Two or Three Things I Know About Her (1966)	4.5
Innocents, The (1961)	4.5
Wallace and Gromit Aardman Animation (1996)	4.5
Casablanca (1942)	4.5
Paths of Glory (1957)	4.5

Problem 1 continued on next page...

```
9.  b. By men under 40?
```

Title	Rating
Hearts and Minds (1996)	5.0
Faithful (1996)	5.0
Marlene Dietrich: Shadow and Light (1996)	5.0
Strawberry and Chocolate (Fresa y chocolate) (1993)	5.0
Late Bloomers (1996)	5.0
Solo (1996)	5.0
Grateful Dead (1995)	5.0
Prefontaine (1997)	5.0
Rendezvous in Paris (Rendez-vous de Paris, Les) (1995)	5.0
World of Apu, The (Apur Sansar) (1959)	5.0
Aparajito (1956)	5.0
Ace Ventura: When Nature Calls (1995)	5.0
Star Kid (1997)	5.0
Two or Three Things I Know About Her (1966)	5.0
Poison Ivy II (1995)	5.0
Double Happiness (1994)	5.0
Little City (1998)	5.0
Boxing Helena (1993)	5.0
Spice World (1997)	5.0
They Made Me a Criminal (1939)	5.0
Great Day in Harlem, A (1994)	5.0
Little Princess, The (1939)	5.0
Unstrung Heroes (1995)	5.0
Leading Man, The (1996)	5.0
Indian Summer (1996)	5.0
Pather Panchali (1955)	4.8
A Chef in Love (1996)	4.7
Whole Wide World, The (1996)	4.7
Close Shave, A (1995)	4.7
Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)	4.6

Problem 1 continued on next page...

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

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

10. b. By women under 40?

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

Listing 1: Modified recommendations.py

```
from __future__ import division
# 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
import numpy as np
import operator
from collections import Counter


# 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

  return round(r,3)

# Returns the best matches for person from the prefs dictionary.
# Number of results and similarity function are optional params.
def topMatches(prefs,person,n=1682,similarity=sim_pearson):
  scores=[(similarity(prefs,person,other),other)
                  for other in prefs if other!=person]
  scores.sort()
  scores.reverse()
  return scores[0:n]

# Gets recommendations for a person by using a weighted average
# of every other user's rankings
def getRecommendations(prefs,person,similarity=sim_pearson):
  totals={}
  simSums={}
  for other in prefs:
    # don't compare me to myself
    if other==person: continue
    sim=similarity(prefs,person,other)

    # ignore scores of zero or lower
    if sim<=0: continue
    for item in prefs[other]:

      # only score movies I haven't seen yet
      if item not in prefs[person] or prefs[person][item]==0:
        # Similarity * Score
        totals.setdefault(item,0)
        totals[item]+=prefs[other][item]*sim
        # Sum of similarities
```

```
        simSums.setdefault(item,0)
        simSums[item]+=sim

  # Create the normalized list
  rankings=[(total/simSums[item],item) for item,total in totals.items()]

  # Return the sorted list
  rankings.sort()
  rankings.reverse()
  return rankings

def transformPrefs(prefs):
  result={}
  for person in prefs:
    for item in prefs[person]:
      result.setdefault(item,{})

      # Flip item and person
      result[item][person]=prefs[person][item]
  return result


def calculateSimilarItems(prefs,n=10):
  # Create a dictionary of items showing which other items they
  # are most similar to.
  result={}
  # Invert the preference matrix to be item-centric
  itemPrefs=transformPrefs(prefs)
  c=0
  for item in itemPrefs:
    # Status updates for large datasets
    c+=1
    if c%100==0: print "%d / %d" % (c,len(itemPrefs))
    # Find the most similar items to this one
    scores=topMatches(itemPrefs,item,n=n,similarity=sim_distance)
    result[item]=scores
  return result

def getRecommendedItems(prefs,itemMatch,user):
  userRatings=prefs[user]
  scores={}
  totalSim={}
  # Loop over items rated by this user
  for (item,rating) in userRatings.items( ):

    # Loop over items similar to this one
    for (similarity,item2) in itemMatch[item]:

      # Ignore if this user has already rated this item
      if item2 in userRatings: continue
      # Weighted sum of rating times similarity
      scores.setdefault(item2,0)
      scores[item2]+=similarity*rating
```

```
      # Sum of all the similarities
      totalSim.setdefault(item2,0)
      totalSim[item2]+=similarity

  # Divide each total score by total weighting to get an average
  rankings=[(score/totalSim[item],item) for item,score in scores.items( )]

  # Return the rankings from highest to lowest
  rankings.sort( )
  rankings.reverse( )
  return rankings

def calculateSimilarUser(prefs,n=5):
  # Create a dictionary of users showing which other users they
  # are most similar to.
  result={}
  c=0
  for user in prefs:
    # Status updates for large datasets
    c+=1
    if c%100==0: print "%d / %d" % (c,len(prefs))
    # Find the most similar items to this one
    scores=topMatches(prefs,user,n=n,similarity=sim_pearson)
    result[user]=scores
  return result

     # Find the mean
def mean(a):
    return sum(a) / len(a)


def loadMovieLens():
  # Get movie titles
  movies={}
  for line in open('u.item'):
    (id,title)=line.split('|')[0:2]
    movies[id]=title
  # Load data
  prefs={}
  for line in open('u.data'):
    (user,movieid,rating)=line.split('\t')[0:3]
    prefs.setdefault(user,{})
    prefs[user][movies[movieid]]=float(rating)

  # Load data gender
  gender={}
  for line in open('u.user'):
    (id,age,g) = line.split('|')[0:3]
    gender.setdefault(id,[])
    age_g= [age,g]
    gender[id]=age_g

  ##Q1. What 5 movies have the highest average ratings?
```

```
rating_hi_avg = {}
for user in prefs.keys():
   for key,value in prefs[user].iteritems():
        rating_hi_avg.setdefault(key,[])
        rating_hi_avg[key].append(value)
average = {}
for movie in rating_hi_avg.keys():
  average[movie] = mean(rating_hi_avg[movie])

sorted_x = sorted(average.iteritems(), key=operator.itemgetter(1))
sorted_x.reverse()
print ("Q1 Answer")
for (key,value) in sorted_x[0:10]:
 print key,'  &  ',value,' \\\\'

##Q2. What 5 movies received the most ratings?
lengthList = {}
for movie in rating_hi_avg.keys():
  lengthList[movie] = len(rating_hi_avg[movie])

sorted_x = sorted(lengthList.iteritems(), key=operator.itemgetter(1))
sorted_x.reverse()
print ("Q2 Answer")
for (key,value) in sorted_x[0:5]:
 print key,'  &  ',value,' \\\\'

##Q3. What 5 movies were rated the highest on average by women?
rating_women = {}
for user in prefs.keys():
  if gender[user][1] == 'M':
    continue
  for key,value in prefs[user].iteritems():
        rating_women.setdefault(key,[])
        rating_women[key].append(value)
average = {}
for movie in rating_women.keys():
  average[movie] = mean(rating_women[movie])

sorted_x = sorted(average.iteritems(), key=operator.itemgetter(1))
sorted_x.reverse()
print ("Q3 Answer")
for (key,value) in sorted_x[0:11]:
 print key,'  &  ',value,' \\\\'

##Q4. What 5 movies were rated the highest on average by men?
rating_men = {}
for user in prefs.keys():
  if gender[user][1] == 'F':
    continue
  for key,value in prefs[user].iteritems():
        rating_men.setdefault(key,[])
        rating_men[key].append(value)
average = {}
```

Problem 1 continued on next page. . .

```
for movie in rating_men.keys():
  average[movie] = mean(rating_men[movie])

sorted_x = sorted(average.iteritems(), key=operator.itemgetter(1))
sorted_x.reverse()
print ("Q4 Answer")
for (key,value) in sorted_x[0:15]:
 print key,'  &  ',value,' \\\\'

##Q5. What movie received ratings most like Top Gun?
# reverse pref
item_prefs = transformPrefs(prefs)
print ("Q5 Answer")
print('Most Like : ',topMatches(item_prefs,'Top Gun (1986)')[0])
print('Least Like : ',topMatches(item_prefs,'Top Gun (1986)')[len(topMatches(
    item_prefs,'Top Gun (1986)')) - 1 ])

##Q6. Which 5 raters rated the most films?
userMostRating = {}
for user in prefs.keys():
   userMostRating.setdefault(user,len(prefs[user]))

sorted_x = sorted(userMostRating.iteritems(), key=operator.itemgetter(1))
sorted_x.reverse()
print ("Q6 Answer")
for (key,value) in sorted_x[0:5]:
 print key,'  &  ',value,' \\\\'

##Q7. Which 5 raters most agreed with each other?
print ("Q7 Answer")
simi_users = calculateSimilarUser(prefs,1)
i=0
cumulative={}
for user in simi_users:
  cumulative.setdefault(user,0)
  print("UserId: "), user
  num=0
  raters=[]
  print("UserValue: ")
  print simi_users[user]
  for (key,value) in simi_users[user]:
    num+=key
    raters.append(value)
  print(num)
  if num >= 4.0:
    i=1+i
  cumulative[user]=(num,raters)
  print cumulative[user]

  print (i)
sorted_x = sorted(cumulative.iteritems(), key=operator.itemgetter(1))
sorted_x.reverse()
```

```
for (key,value) in sorted_x[0:5]:
 print key,'  &  ',value,' \\\\'

##Q8. Which 5 raters most disagreed with each other (negative correlation)

sorted_x = sorted(cumulative.iteritems(), key=operator.itemgetter(1))
print ("Q8 Answer")
for (key,value) in sorted_x[0:5]:
 print key,'  &  ',value,' \\\\'

##Q9. What movie was rated highest on average by men over 40?


rating_m_up={}
rating_m_down={}
for user in prefs.keys():
  if gender[user][0] < '40' and gender[user][1] == 'M':
    for key,value in prefs[user].iteritems():
        rating_m_down.setdefault(key,[])
        rating_m_down[key].append(value)
  elif gender[user][0] > '40' and gender[user][1] == 'M':
    for key,value in prefs[user].iteritems():
       rating_m_up.setdefault(key,[])
       rating_m_up[key].append(value)
  else:
     continue
average_m_down = {}
for movie in rating_m_down.keys():
  average_m_down[movie] = mean(rating_m_down[movie])

sorted_md = sorted(average_m_down.iteritems(), key=operator.itemgetter(1))
sorted_md.reverse()
print ("Q9 A Answer")
for (key,value) in sorted_md[0:30]:
 print key,'  &  ',value,' \\\\'
average_m_up = {}
for movie in rating_m_up.keys():
  average_m_up[movie] = mean(rating_m_up[movie])

sorted_mu = sorted(average_m_up.iteritems(), key=operator.itemgetter(1))
sorted_mu.reverse()
print ("Q9 B Answer")
for (key,value) in sorted_mu[0:30]:
 print key,'  &  ',value,' \\\\'

##Q10. What movie was rated highest on average by women over 40?
rating_w_up={}
rating_w_down={}
for user in prefs.keys():
  if gender[user][0] < '40' and gender[user][1] == 'F':
    for key,value in prefs[user].iteritems():
        rating_w_down.setdefault(key,[])
        rating_w_down[key].append(value)
  elif gender[user][0] > '40' and gender[user][1] == 'F':
```

```
      for key,value in prefs[user].iteritems():
         rating_w_up.setdefault(key,[])
         rating_w_up[key].append(value)
    else:
       continue

  average_w_up = {}
  for movie in rating_w_up.keys():
    average_w_up[movie] = mean(rating_w_up[movie])

  sorted_wu = sorted(average_w_up.iteritems(), key=operator.itemgetter(1))
  sorted_wu.reverse()

  print ("Q10 A Answer")
  for (key,value) in sorted_wu[0:30]:
   print key,'  &  ',value,' \\\\'

  average_w_down = {}
  for movie in rating_w_down.keys():
    average_w_down[movie] = mean(rating_w_down[movie])

  sorted_wd = sorted(average_w_down.iteritems(), key=operator.itemgetter(1))
  sorted_wd.reverse()

  print ("Q10 B Answer")
  for (key,value) in sorted_wd[0:30]:
   print key,'  &  ',value,' \\\\'

  return prefs

loadMovieLens();
```

References

[1] GroupLens. MovieLens. http://grouplens.org/datasets/movielens/