Assignment 7 Report

Professor Nelson

By Tyler Medina

4/6/17

Part 1

1.1 Getting Substitute Candidates

To get candidates to substitute me for the assignment, I searched the data set for people that had a similar age (23), gender (male), and occupation (student). I decided to use the first three hits that were returned. This ended up being users 33, 37, and 66.

```
readfile = open('u.user', 'r')

for line in readfile:
    (id, age, gender, occupation, zipcode) = line.split('|')
    if age == '23' and gender == 'M' and occupation == 'student':
        print (id, age, gender, occupation, zipcode)
```

1.2 Finding the Right Substitute

I used the loadMovieLens function provided in the textbook to get the all the movies my substitute candidates reviewed. With the full list, I could manually see the top 3 movies and bottom 3 movies each candidate had rated.

```
def loadMovieLens(path='./data'):
  # Get movie titles
  movies={}
  for line in open(path+'/u.item'):
    (id,title)=line.split('|')[0:2]
    movies[id]=title

  # Load data
  prefs={}
  for line in open(path+'/u.data'):
    (user,movieid,rating,ts)=line.split('\t')
    prefs.setdefault(user,{})
    prefs[user][movies[movieid]]=float(rating)
  return prefs

if __name__ == '__main__':

    prefs=loadMovieLens()
    print("Movies reviewed by person 66: " )
    print (prefs['66'])
```

1.3 Analysis

By taking the data I gathered, I could create this table below to organize the data. This makes it simple to pick out the person that would best represent my taste in movies in this assignment.

Person 33's top 3:

Movie Title	Rating
Titanic (1997)	5.0
Air Force One	4.0
Alien: Resurrection (1997)	4.0

Person 33's bottom 3:

Movie Title	Rating
Liar Liar (1997)	3.0
Soul Food (1997)	3.0
Devil's Own, The (1997)	3.0

Person 37's top 3:

Movie Title	Rating
Pulp Fiction (1994)	5.0
Raiders of the Lost Ark (1981)	5.0
The Terminator (1984)	5.0

Person 37's bottom 3:

Movie Title	Rating
Jurassic Park (1993)	1.0
Twister (1996)	2.0
The Arrival (1996)	2.0

Person 66's top 3:

Movie Title	Rating
Return of the Jedi (1983)	5.0
Air Force One (1997)	5.0
Ransom (1996)	5.0

Person 66's bottom 3:

Movie Title	Rating
English Patient, The (1996)	1.0
Muppet treasure Island (1996)	1.0
Excessive Baggage (1997)	1.0

The person that would most accurately represent me is person 37. I was close to choosing person 33, but the gave Alien: Resurrection a 4.0, and that's unacceptable. Person 66 was appealing, but when you put Air Force One in the same tier as Return of the Jedi, then you can't be taken seriously. Person 37 on the other hand, gave Terminator and Pulp Fiction a 5.0, so he knows what he's doing.

Part 2

2.1 Finding Correlated users

By using the sim_pearson and topMatches functions that are provided in the Collective Intelligence textbook, I could find the users that most and least correlate with the substitute me.

```
def sim_pearson(prefs, p1, p2):
    si = {}
    for item in prefs[p1]:
        if item in prefs[p2]:
            si[item] = 1
    if len(si) == 0:
        return 0
    n = len(si)
    sum1 = sum([prefs[p1][it] for it in si])
    sum2 = sum([prefs[p2][it] for it in si])
    sum1Sq = sum([pow(prefs[p1][it], 2) for it in si])
    sum2Sq = sum([pow(prefs[p2][it], 2) for it in si])
    pSum = sum([prefs[p1][it] * prefs[p2][it] for it in si])
    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
```

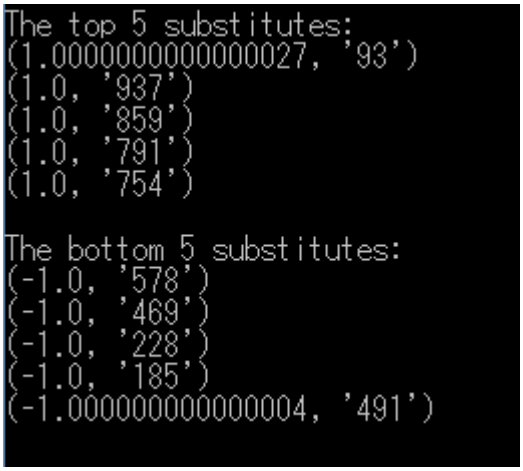
```
def topMatches(prefs,person,t=5,b=-5,similarity=sim_pearson):
  scores=[(similarity(prefs,person,other),other)
                for other in prefs if other!=person]
  scores.sort()
  scores.reverse()
  return scores[0:t], scores[b:]


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)

(mostMatching, leastMatching) = topMatches(prefs, '37')
print 'The top 5 substitutes:'
print str(mostMatching[0]) + '\n' + str(mostMatching[1]) + '\n' + str(mostMatching[2]) + '\n' + str(mostMatching[3]) + '\n' + str(mostMatching[4]) + '\n'

print 'The bottom 5 substitutes:'
print str(leastMatching[0]) + '\n' + str(leastMatching[1]) + '\n' + str(leastMatching[2]) + '\n' + str(leastMatching[3]) + '\n' + str(leastMatching[4]) + '\n'
```



Part 3

3.1 Calculating Recommendations

To get the recommendations, I used the getRecommendations function found on page 16 in the Collective Intelligence textbook.

```
def getRecommendations(prefs,person,t=5,b=-5,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[:t], rankings[b:]
```

```
The top 5 recommendations for user 37 are:
(5.0, 'They Made Me a Criminal (1939)')
(5.0, "Someone Else's America (1995)")
(5.0, 'Santa with Muscles (1996)')
(5.0, 'Prefontaine (1997)')
(5.0, 'Marlene Dietrich: Shadow and Light (1996) ')

The bottom 5 recommendations for user 37 are:
(1.0, 'Amityville Curse, The (1990)')
(1.0, 'Amityville 3-D (1983)')
(1.0, "Amityville 1992: It's About Time (1992)")
(1.0, 'American Strays (1996)')
(1.0, '3 Ninjas: High Noon At Mega Mountain (1998)')
```

3.2 Analyzing the Results

I am unfamiliar with these recommendations. After reading all the synopsis' of the top five recommendations I can honestly conclude that none of these movies interest me. The bottom five recommendations have low ratings. This is probably why they were the bottom recommendations, but I wouldn't mind watching the three Amityville movies that were recommended. I am a fan of horror movie. Even if they are terrible, I can still find valuable entertainment in them. This disparity can be a result of two things. One possibility is that I could have some far outlier of movie preferences that can't be reasonably be catered to by this recommendation function. The other possible reason is that there could be a wider difference in movie preferences between me and my substitute. Maybe the substitute would be satisfied with these recommendations, and he just isn't a very good substitute for me. This could be a problem with getting substitutes strictly off age, occupation, and gender.

Part 4

4.1 Getting Personal Correlations

On page 18 in the Collective Intelligence textbook there is a transformPrefs function. This function swaps the items and people. Instead of searching for similarities between people, it will look for similarities between movies. This allows me to search for recommendations based of my favorite and least favorite movies in the data set. The movie I chose for my favorite is Aliens (1986) and the movie I chose as my least favorite is Jaws 3-D (1983).

```
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
```

```
The top 5 movies similar to "Aliens (1986)" are:
(1.0000000000000004, 'Carpool (1996)')
(1.0000000000000004, 'Jefferson in Paris (1995)')
(1.0, 'Underneath, The (1995)')
(1.0, 'Tough and Deadly (1995)')
(1.0, 'Total Eclipse (1995)')

The bottom 5 movies similar to "Aliens (1986)" are:
(-1.0, 'Curdled (1996)')
(-1.0, 'Country Life (1994)')
(-1.0, 'Bliss (1997)')
(-1.0, 'Babysitter, The (1995)')
(-1.0, 'Assignment, The (1997)')

The top 5 movies similar to "Jaws 3-D (1983)" are:
(1.0000000000000033, 'Adventures of Robin Hood, The (1938)')
(1.0000000000000027, 'Daylight (1996)')
(1.0000000000000027, 'Arsenic and Old Lace (1944)')
(1.0000000000000007, 'Crash (1996)')
(1.0, 'Wild Things (1998)')

The bottom 5 movies similar to "Jaws 3-D (1983)" are:
(-1.0000000000000002, 'Delicatessen (1991)')
(-1.0000000000000007, 'Turbulence (1997)')
(-1.0000000000000007, 'Corrina, Corrina (1994)')
(-1.000000000000001, 'Kalifornia (1993)')
(-1.000000000000001, 'Addiction, The (1995)')
```

4.2 Analyzing the Results

I haven't seen any of these movies. After reading the synopsis of each of the correlated films, I am disappointed. A lot of the movies aren't horror, and the ones that are, are about serial killer. I was hoping that the movies that were determined to be correlated would have monsters in them like in Jaws 3-D and Aliens. My hypothesis would be that there is a small pool of films containing monsters in the data set and that it's unrealistic to expect the functions to find significant correlation between them.