

Web Science cs532-s16

ASSIGNMENT 7 REPORT

DR. MICHAEL L. NELSON

BY: HUAN HUANG

03/31/2016

Assignment Description

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 MovieLens data sets.

The MovieLens 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. We are using the “100k dataset”; available for download from: <http://grouplens.org/datasets/movielens/100k>

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.

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
```

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.

Example:

```
161|Top Gun (1986)|01-Jan-1986||http://us.imdb.com/M/title-exact?Top
%20Gun%20(1986)|0|1|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0|0|0
```

```
162|On Golden Pond (1981)|01-Jan-1981||http://us.imdb.com/M/title-
exact?On%20Golden%20Pond%20(1981)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0
```

```
163|Return of the Pink Panther, The (1974)|01-Jan-1974||http://
us.imdb.com/M/title-exact?Return%20of%20the%20Pink%20Panther,%20The
%20(1974)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0| 0|0|0|0|0
```

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.

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. Feel free to modify the PCI code to answer the following questions.

Problem 1

Find 3 users who are closest to you in terms of age, gender, and occupation. For each of those 3 users:

- what are their top 3 favorite films? - bottom 3 least favorite films?

Based on the movie values in those 6 tables (3 users X (favorite + least)), choose a user that you feel is most like you. Feel free to note any outliers (e.g., “I mostly identify with user 123, except I did not like “Ghost” at ll”).

This user is the “substitute you”.

Answer

To solve this problem, I first downloaded the 3 files from the link provided in the assignment description above. Then I proceed to write a code to identify the users who are closest resemblance of me, based on the age, gender, and occupation in the u.user file. Below are all of the candidates who fit the 3 criteria.

```
104 27 M student 55108
286 27 M student 15217
429 27 M student 29205
484 27 M student 21208
758 27 M student 53706
913 27 M student 76201
```

Figure 1: The initial candidates to be the substitute me

```
1
2 readfile = open('u.user', 'r')
3
4 for line in readfile:
5     (userid, age, gender, occupation, zipcode) = line.split('|')
6     if age == '27' and gender == 'M' and occupation == 'student':
7         print userid, age, gender, occupation, zipcode
```

From this list, I hand picked 3 individuals for the next stage. The 3 users I picked are 286, 429, and 484.

```
1 286 27 M student 15217
2
3 429 27 M student 29205
4
5 484 27 M student 21208
```

Now, for each of the 3 user, I access all of his movie ratings, if the rating is 5, the movie id is appended to an array; if the rating is 1, the movie id is appended to another array. Then I open the u.item file to get the movie titles. I only print out 3 items in each of the array, this gives me the 3 favorite and 3 least favorite movies of the 3 users.

```

hhuang@ubuntu:~/Documents/CS532/WebScienceAssig/Assignment7/Problem1$ python mostLest.py
The favorite 3 movies of user 286 are:
Romy and Michele's High School Reunion (1997)
Tales From the Crypt Presents: Demon Knight (1995)
Scream (1996)

The 3 least favorite movies of user 286 are:
Tommy Boy (1995)
Crash (1996)
Moll Flanders (1996)

The favorite 3 movies of user 429 are:
Remains of the Day, The (1993)
Lawrence of Arabia (1962)
Highlander (1986)

The 3 least favorite movies of user 429 are:
Homeward Bound: The Incredible Journey (1993)
Homeward Bound II: Lost in San Francisco (1996)
Free Willy 3: The Rescue (1997)

The favorite 3 movies of user 484 are:
Four Weddings and a Funeral (1994)
Terminator, The (1984)
Terminator 2: Judgment Day (1991)

The 3 least favorite movies of user 484 are:
Natural Born Killers (1994)
Dead Man Walking (1995)
Rumble in the Bronx (1995)

```

Figure 2: The 3 most and least favorite movies of the 3 users

Based on the favorite and least favorite movies of the 3 users, I can clearly tell that user 484 is much closer to me in term of movie preference. Because I love The Terminator I and II. Although, I have not watched any other movies in his favorite or least favorite movie list, he is still the closest match out of the 3 candidates.

```

1
2 readfile = open('u.data', 'r')
3 readfile2 = open('u.item', 'r')
4 #savefile = open('mostLest.txt', 'a')
5 data = readfile.readlines()
6 data2 = readfile2.readlines()
7
8 substitutes = ['286', '429', '484']
9 most = []
10 lest = []
11 favtitles1 = []
12 favtitles2 = []
13 favtitles3 = []
14 hatetitles1 = []
15 hatetitles2 = []
16 hatetitles3 = []
17 def favorites(item):
18     most = []
19     for line in data:
20         (userid, itemid, rating, timestamp) = line.split(' ')

```

```

21     if userid == item and rating == '5':
22         most.append((userid, itemid, rating))
23     return most
24
25 def lestfav(item):
26     lest = []
27     for line in data:
28         (userid, itemid, rating, timestamp) = line.split(' ')
29         if userid == item and rating == '1':
30             lest.append((userid, itemid, rating))
31     return lest
32
33 def movies(mvid):
34     for line in data2:
35         (movieid, movietitle, releasedate, videodate, IMDBURL, unknown, action,
36          adventure, animation, children, comedy, crime, docu, drama, fantasy,
37          filenoir, horror, musical, mystery, romance, scifi, thriller, war, western
38          ) = line.split('|')
39         if movieid == mvid:
40             return movietitle
41
42 for item in substitutes:
43     most.append(favorites(item))
44 for item in substitutes:
45     lest.append(lestfav(item))
46 subfav1 = most[0][0][1], most[0][1][1], most[0][2][1]
47 subfav2 = most[1][0][1], most[1][1][1], most[1][2][1]
48 subfav3 = most[2][0][1], most[2][1][1], most[2][2][1]
49 sublest1 = lest[0][0][1], lest[0][1][1], lest[0][2][1]
50 sublest2 = lest[1][0][1], lest[1][1][1], lest[1][2][1]
51 sublest3 = lest[2][0][1], lest[2][1][1], lest[2][2][1]
52 for mvid in subfav1:
53     favtitles1.append(movies(mvid))
54 for mvid in subfav2:
55     favtitles2.append(movies(mvid))
56 for mvid in subfav3:
57     favtitles3.append(movies(mvid))
58 for mvid in sublest1:
59     hatetitles1.append(movies(mvid))
60 for mvid in sublest2:
61     hatetitles2.append(movies(mvid))
62 for mvid in sublest3:
63     hatetitles3.append(movies(mvid))
64
65 print 'The favorite 3 movies of user ' + substitutes[0] + ' are:'
66 print favtitles1[0] + '\n' + favtitles1[1] + '\n' + favtitles1[2] + '\n'
67 print 'The 3 least favorite movies of user ' + substitutes[0] + ' are:'
68 print hatetitles1[0] + '\n' + hatetitles1[1] + '\n' + hatetitles1[2] + '\n'
69 print 'The favorite 3 movies of user ' + substitutes[1] + ' are:'
70 print favtitles2[0] + '\n' + favtitles2[1] + '\n' + favtitles2[2] + '\n'
71 print 'The 3 least favorite movies of user ' + substitutes[1] + ' are:'
72 print hatetitles2[0] + '\n' + hatetitles2[1] + '\n' + hatetitles2[2] + '\n'
73 print 'The favorite 3 movies of user ' + substitutes[2] + ' are:'
74 print favtitles3[0] + '\n' + favtitles3[1] + '\n' + favtitles3[2] + '\n'

```

```

72 print 'The 3 lest favorite movies of user ' + substitutes[2] + ' are: '
73 print hatetitles3[0] + '\n' + hatetitles3[1] + '\n' + hatetitles3[2] + '\n'

```

Problem 2

Which 5 users are most correlated to the substitute you? Which 5 users are least correlated (i.e., negative correlation)?

Answer

Let me be clear at this point, for this problem, and the rest of the problems in this assignment, I copied and modified code from recommendations.py provided by the book Programming Collective Intelligence. They have a github account where all of their code are available there, here is the link for recommendations.py: [://github.com/cataska/programming-collective-intelligence-code/blob/master/chapter2/recommendations.py](https://github.com/cataska/programming-collective-intelligence-code/blob/master/chapter2/recommendations.py).

I applied 3 functions from the recommendations.py file, the loadMovieLens() which loop through every line in u.item file and store the movie id and title into the a dictionary, then loops through u.data file to attach each movie and its ratings to its rater, the result is stored in dictionary prefs{}. Then the prefs will be passed to function topMatches() along with the user id “484” to find the most and least correlated users of user 484. The function topMatches() will call another function sim_pearson() to use Pearson method of calculating correlation coefficient between user 484 and other users based on their ratings.

```

hhuang@ubuntu:~/Documents/CS532/WebScienceAssig/Assignment7/Problem2$ python correlated.py
The top 5 users who are most correlated to the substitute me are:
(1.0, '598')
(1.0, '574')
(1.0, '570')
(1.0, '531')
(1.0, '485')

The top 5 users who are least correlated to the substitute me are:
(-1.0, '740')
(-1.0, '681')
(-1.0, '662')
(-1.0, '273')
(-1.0000000000000004, '111')

hhuang@ubuntu:~/Documents/CS532/WebScienceAssig/Assignment7/Problem2$ python correlated.py > correlated.txt

```

Figure 3: The 5 most and least correlated users to the substitute me

```

1 #!/usr/bin/python
2 from math import sqrt
3

```

```

4 def sim_pearson(prefs, p1, p2):
5     '''
6     Returns the Pearson correlation coefficient for p1 and p2.
7     '''
8
9     # Get the list of mutually rated items
10    si = {}
11    for item in prefs[p1]:
12        if item in prefs[p2]:
13            si[item] = 1
14    # If they are no ratings in common, return 0
15    if len(si) == 0:
16        return 0
17    # Sum calculations
18    n = len(si)
19    # Sums of all the preferences
20    sum1 = sum([prefs[p1][it] for it in si])
21    sum2 = sum([prefs[p2][it] for it in si])
22    # Sums of the squares
23    sum1Sq = sum([pow(prefs[p1][it], 2) for it in si])
24    sum2Sq = sum([pow(prefs[p2][it], 2) for it in si])
25    # Sum of the products
26    pSum = sum([prefs[p1][it] * prefs[p2][it] for it in si])
27    # Calculate r (Pearson score)
28    num = pSum - sum1 * sum2 / n
29    den = sqrt((sum1Sq - pow(sum1, 2) / n) * (sum2Sq - pow(sum2, 2) / n))
30    if den == 0:
31        return 0
32    r = num / den
33    return r
34
35 def topMatches(prefs, person, t=5, b=-5, similarity=sim_pearson):
36     scores=[(similarity(prefs, person, other), other)
37             for other in prefs if other!=person]
38     scores.sort()
39     scores.reverse()
40     return scores[0:t], scores[b:]
41
42
43 movies={}
44 for line in open('u.item'):
45     (id, title)=line.split('|')[0:2]
46     movies[id]=title
47
48 # Load data
49 prefs={}
50 for line in open('u.data'):
51     (user, movieid, rating)=line.split('\t')[0:3]
52     prefs.setdefault(user, {})
53     prefs[user][movies[movieid]]=float(rating)
54
55 (mostMatching, leastMatching) = topMatches(prefs, '484')
56 print 'The top 5 users who are most correlated to the substitute me are:'
57 print str(mostMatching[0]) + '\n' + str(mostMatching[1]) + '\n' + str(

```



```

mostMatching[2]) + '\n' + str(mostMatching[3]) + '\n' + str(mostMatching
[4]) + '\n'
58
59 print 'The top 5 users who are least correlated to the substitute me are:'
60 print str(leastMatching[0]) + '\n' + str(leastMatching[1]) + '\n' + str(
    leastMatching[2]) + '\n' + str(leastMatching[3]) + '\n' + str(
    leastMatching[4]) + '\n'

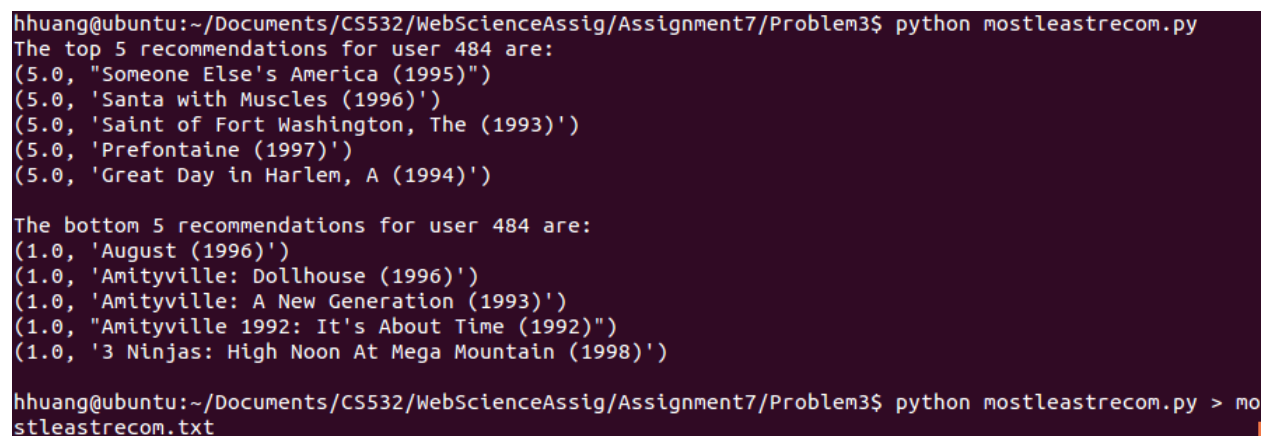
```

Problem 3

Compute ratings for all the films that the substitute you hasn't seen. Provide a list of the top 5 recommendations for films that the substitute you should see. Provide a list of the bottom 5 recommendations (i.e., films the substitute you is almost certain to hate).

Answer

For this problem, I used 3 functions from recommendations.py: use loadMovieLens() to get the necessary data, sim_pearson() to calculate the correlation coefficient of user 484 and other users, and getRecommendations() to calculate the recommendations for user 484 based on the result from sim_pearson() function.



```

hhuang@ubuntu:~/Documents/CS532/WebScienceAssig/Assignment7/Problem3$ python mostleastrecom.py
The top 5 recommendations for user 484 are:
(5.0, 'Someone Else's America (1995)')
(5.0, 'Santa with Muscles (1996)')
(5.0, 'Saint of Fort Washington, The (1993)')
(5.0, 'Prefontaine (1997)')
(5.0, 'Great Day in Harlem, A (1994)')

The bottom 5 recommendations for user 484 are:
(1.0, 'August (1996)')
(1.0, 'Amityville: Dollhouse (1996)')
(1.0, 'Amityville: A New Generation (1993)')
(1.0, 'Amityville 1992: It's About Time (1992)')
(1.0, '3 Ninjas: High Noon At Mega Mountain (1998)')

hhuang@ubuntu:~/Documents/CS532/WebScienceAssig/Assignment7/Problem3$ python mostleastrecom.py > mo
stleastrecom.txt

```

Figure 4: The 5 most and least recommended films for the substitute me

```

1 #!/usr/bin/python
2 from math import sqrt
3
4 def sim_pearson(prefs, p1, p2):
5     '''
6     Returns the Pearson correlation coefficient for p1 and p2.
7     '''
8
9     # Get the list of mutually rated items

```

```

10  si = {}
11  for item in prefs[p1]:
12      if item in prefs[p2]:
13          si[item] = 1
14  # If they are no ratings in common, return 0
15  if len(si) == 0:
16      return 0
17  # Sum calculations
18  n = len(si)
19  # Sums of all the preferences
20  sum1 = sum([prefs[p1][it] for it in si])
21  sum2 = sum([prefs[p2][it] for it in si])
22  # Sums of the squares
23  sum1Sq = sum([pow(prefs[p1][it], 2) for it in si])
24  sum2Sq = sum([pow(prefs[p2][it], 2) for it in si])
25  # Sum of the products
26  pSum = sum([prefs[p1][it] * prefs[p2][it] for it in si])
27  # Calculate r (Pearson score)
28  num = pSum - sum1 * sum2 / n
29  den = sqrt((sum1Sq - pow(sum1, 2) / n) * (sum2Sq - pow(sum2, 2) / n))
30  if den == 0:
31      return 0
32  r = num / den
33  return r
34
35 def getRecommendations(prefs, person, t=5, b=-5, similarity=sim_pearson):
36     totals={}
37     simSums={}
38     for other in prefs:
39         # don't compare me to myself
40         if other==person: continue
41         sim=similarity(prefs, person, other)
42
43         # ignore scores of zero or lower
44         if sim<=0: continue
45         for item in prefs[other]:
46
47             # only score movies I haven't seen yet
48             if item not in prefs[person] or prefs[person][item]==0:
49                 # Similarity * Score
50                 totals.setdefault(item,0)
51                 totals[item]+=prefs[other][item]*sim
52                 # Sum of similarities
53                 simSums.setdefault(item,0)
54                 simSums[item]+=sim
55
56     # Create the normalized list
57     rankings=[(total/simSums[item],item) for item,total in totals.items()]
58
59     # Return the sorted list
60     rankings.sort()
61     rankings.reverse()
62     return rankings[:t], rankings[b:]
63

```

```

64 movies={}
65 for line in open('u.item'):
66     (id,title)=line.split('|')[0:2]
67     movies[id]=title
68
69 # Load data
70 prefs={}
71 for line in open('u.data'):
72     (user,movieid,rating,ts)=line.split('\t')
73     prefs.setdefault(user,{})
74     prefs[user][movies[movieid]]=float(rating)
75
76 (toprecommendation, botrecommendations) = getRecommendations(prefs, '484')
77 print 'The top 5 recommendations for user 484 are:'
78 print str(toprecommendation[0]) + '\n' + str(toprecommendation[1]) + '\n' +
       str(toprecommendation[2]) + '\n' + str(toprecommendation[3]) + '\n' + str(
       toprecommendation[4]) + '\n'
79 print 'The bottom 5 recommendations for user 484 are:'
80 print str(botrecommendations[0]) + '\n' + str(botrecommendations[1]) + '\n' +
       str(botrecommendations[2]) + '\n' + str(botrecommendations[3]) + '\n' +
       str(botrecommendations[4]) + '\n'

```

Problem 4

Choose your (the real you, not the substitute you) favorite and least favorite film from the data. For each film, generate a list of the top 5 most correlated and bottom 5 least correlated films. Based on your knowledge of the resulting films, do you agree with the results? In other words, do you personally like / dislike the resulting films?

Answer

My favorite movie in this list is “The Godfather: Part II(1974)” and my least favorite movie is “Turbo: A Power Rangers Movie (1997)”. The method to solve this problem is similar to problem 2, but this time, I have to use the function transformPrefs() to remap the dictionary perfs{}. As a result, when the function topMatches() calls function sim_pearson(), it is passing a set of data that enables sim_pearson() to calculate the correlation of movies based on their ratings.

It is hard to determine the accuracy of the recommendations based on this result. Because I have not watched any of the most and least correlated movies for Godfather II, therefore, I can't really agree or disagree with the results of my favorite movie. However, I do agree with the list for my least favorite movie. The top correlated films of “Turbo: A Power Rangers Movie (1997)” are all horrible movies that I have no interest in whatsoever.

```

hhuang@ubuntu:~/Documents/CS532/WebScienceAssig/Assignment7/Problem4$ python myrecommendations.py
The top 5 correlated movies of "Godfather: Part II, The (1974)" are:
(1.0000000000000004, 'Nothing to Lose (1994)')
(1.00000000000000018, 'Dark City (1998)')
(1.0, 'Unhook the Stars (1996)')
(1.0, 'Switchback (1997)')
(1.0, 'Solo (1996)')

The bottom 5 correlated movies of "Godfather: Part II, The (1974)" are:
(-1.0, 'Aparajito (1956)')
(-1.0, 'Above the Rim (1994)')
(-1.0000000000000004, 'Heavyweights (1994)')
(-1.0000000000000007, 'I Like It Like That (1994)')
(-1.0000000000000004, 'Midnight Dancers (Sibak) (1994)')

The top 5 correlated movies of "Turbo: A Power Rangers Movie (1997)" are:
(1.00000000000000022, 'Mrs. Doubtfire (1993)')
(1.0000000000000002, 'Volcano (1997)')
(1.0000000000000002, 'Snow White and the Seven Dwarfs (1937)')
(1.0000000000000002, 'George of the Jungle (1997)')
(1.0000000000000002, 'Con Air (1997)')

The bottom 5 correlated movies of "Turbo: A Power Rangers Movie (1997)" are:
(-1.0, 'Love Bug, The (1969)')
(-1.0, 'Fantasia (1940)')
(-1.0, 'Body Snatchers (1993)')
(-1.0, 'Ben-Hur (1959)')
(-1.0000000000000002, 'Lawnmower Man, The (1992)')

```

Figure 5: The 5 most and least correlated films for my most and least favorite movies

```

1 #!/usr/bin/python
2 from math import sqrt
3
4 def sim_pearson(prefs,p1,p2):
5     # Get the list of mutually rated items
6     si={}
7     for item in prefs[p1]:
8         if item in prefs[p2]: si[item]=1
9
10    # if they are no ratings in common, return 0
11    if len(si)==0: return 0
12
13    # Sum calculations
14    n=len( si )
15
16    # Sums of all the preferences
17    sum1=sum([prefs[p1][it] for it in si])
18    sum2=sum([prefs[p2][it] for it in si])
19
20    # Sums of the squares
21    sum1Sq=sum([pow(prefs[p1][it],2) for it in si])
22    sum2Sq=sum([pow(prefs[p2][it],2) for it in si])
23
24    # Sum of the products
25    pSum=sum([prefs[p1][it]*prefs[p2][it] for it in si])
26
27    # Calculate r (Pearson score)

```

```

28 num=pSum-(sum1*sum2/n)
29 den=sqrt((sum1Sq-pow(sum1,2)/n)*(sum2Sq-pow(sum2,2)/n))
30 if den==0: return 0
31
32 r=num/den
33
34 return r
35
36 def topMatches(prefs, person, t=5, b=-5, similarity=sim_pearson):
37     scores=[(similarity(prefs, person, other), other)
38              for other in prefs if other!=person]
39     scores.sort()
40     scores.reverse()
41     return scores[0:t], scores[b:]
42
43
44 def transformPrefs(prefs):
45     result={}
46     for person in prefs:
47         for item in prefs[person]:
48             result.setdefault(item, {})
49
50         # Flip item and person
51         result[item][person]=prefs[person][item]
52     return result
53
54 movies={}
55 for line in open('u.item'):
56     (id, title)=line.split('|')[0:2]
57     movies[id]=title
58
59 # Load data
60 prefs={}
61 for line in open('u.data'):
62     (user, movieid, rating, ts)=line.split('\t')
63     prefs.setdefault(user, {})
64     prefs[user][movies[movieid]]=float(rating)
65
66 movchoice = 'Godfather: Part II, The (1974)'
67 lestfav = 'Turbo: A Power Rangers Movie (1997)'
68 remap = transformPrefs(prefs)
69 (top1, least1) = topMatches(remap, movchoice, similarity=sim_pearson)
70 (top2, least2) = topMatches(remap, lestfav, similarity=sim_pearson)
71
72 print 'The top 5 correlated movies of "Godfather: Part II, The (1974)" are:'
73 print str(top1[0]) + '\n' + str(top1[1]) + '\n' + str(top1[2]) + '\n' + str(
74     top1[3]) + '\n' + str(top1[4]) + '\n'
75 print 'The bottom 5 correlated movies of "Godfather: Part II, The (1974)" are:'
76 print str(least1[0]) + '\n' + str(least1[1]) + '\n' + str(least1[2]) + '\n' +
77     str(least1[3]) + '\n' + str(least1[4]) + '\n'
78 print 'The top 5 correlated movies of "Turbo: A Power Rangers Movie (1997)"
79 are:'
80 print str(top2[0]) + '\n' + str(top2[1]) + '\n' + str(top2[2]) + '\n' + str(

```

```
    top2[3]) + '\n' + str(top2[4]) + '\n'
78 print 'The bottom 5 correlated movies of "Turbo: A Power Rangers Movie (1997)"
    are:'
79 print str(least2[0]) + '\n' + str(least2[1]) + '\n' + str(least2[2]) + '\n' +
    str(least2[3]) + '\n' + str(least2[4]) + '\n'
80
81 #result = calculateSimilarItems(prefs)
```