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 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. We are using the “100k dataset”; available for download from: http:grouplens.orgdatasetsmovielens100k

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
```

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:

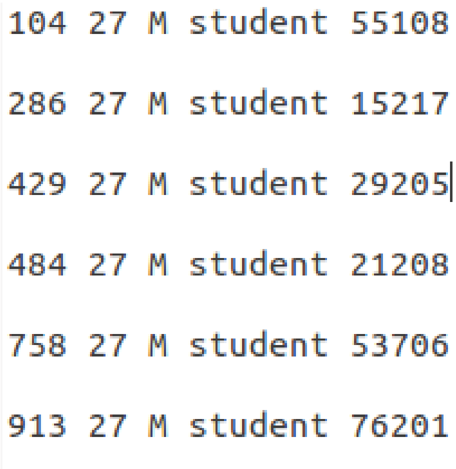
- what are their top 3 favorite films? - bottom 3 least favorite films?

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
```

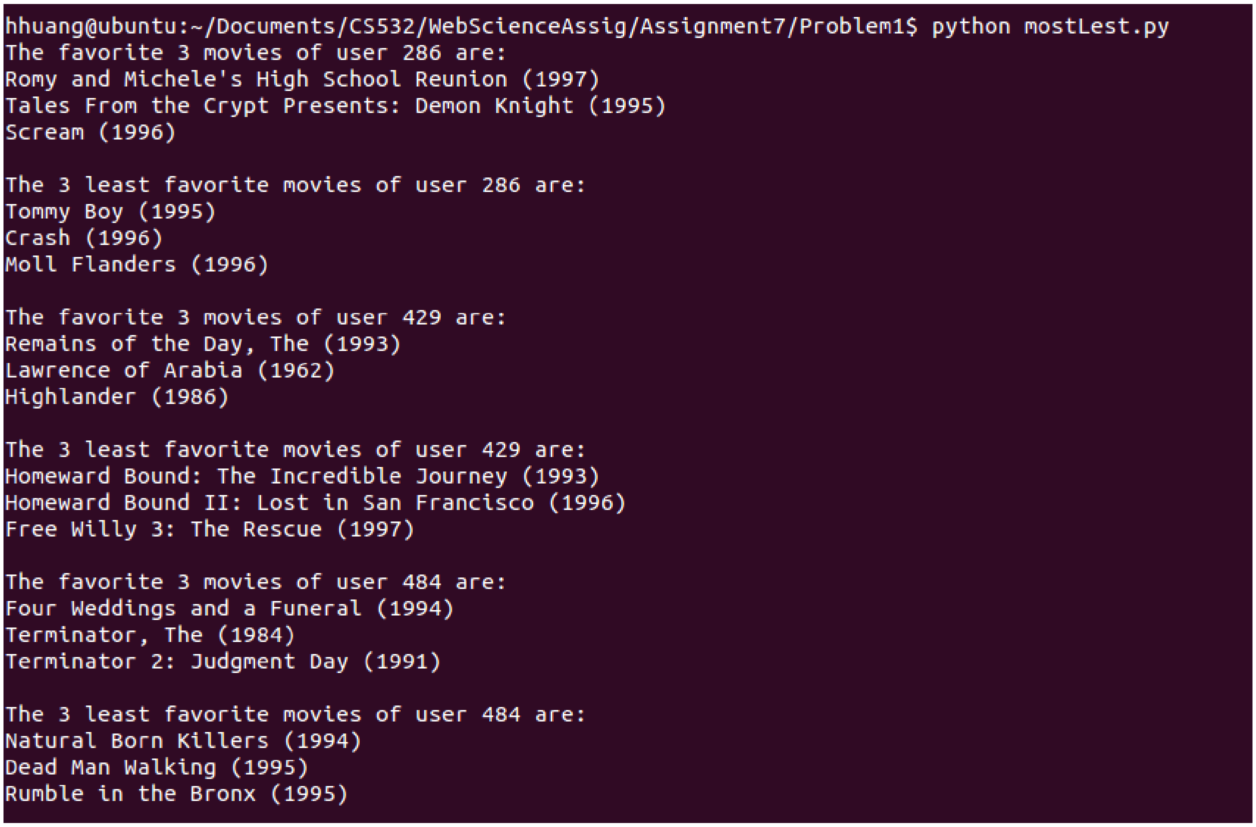
Figure 1: The initial candidates to be the substitute me

```python

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

for line in readfile:
  (userid, age, gender, occupation, zipcode) = line.split('|')
  if age == '27' and gender == 'M' and occupation == 'student':
    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.

```
286 27 M student 15217

429 27 M student 29205

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.

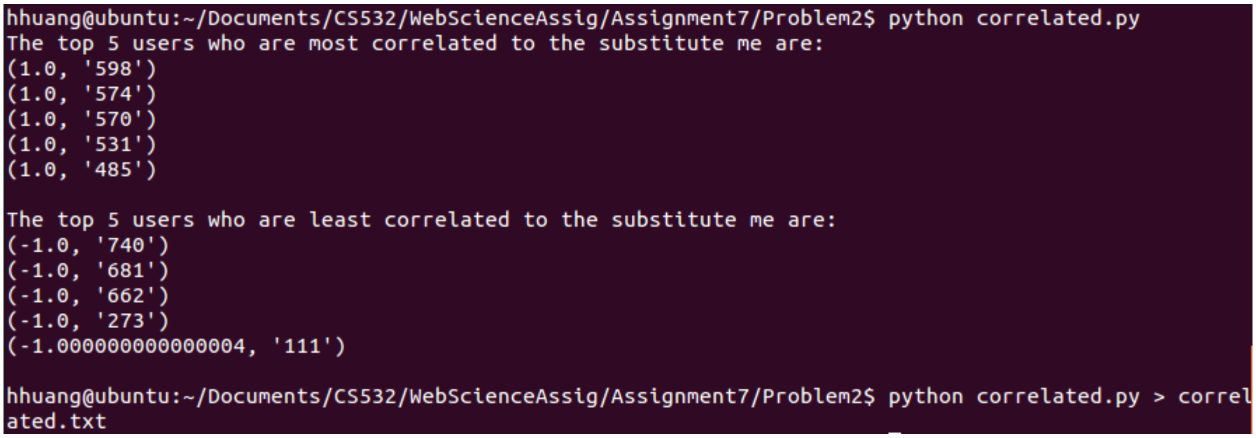
Problem 2

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`.

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')
(-1.000000000000004, '111')

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

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

```
#!/usr/bin/python
from math import sqrt

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

  # Get the list of mutually rated items
```

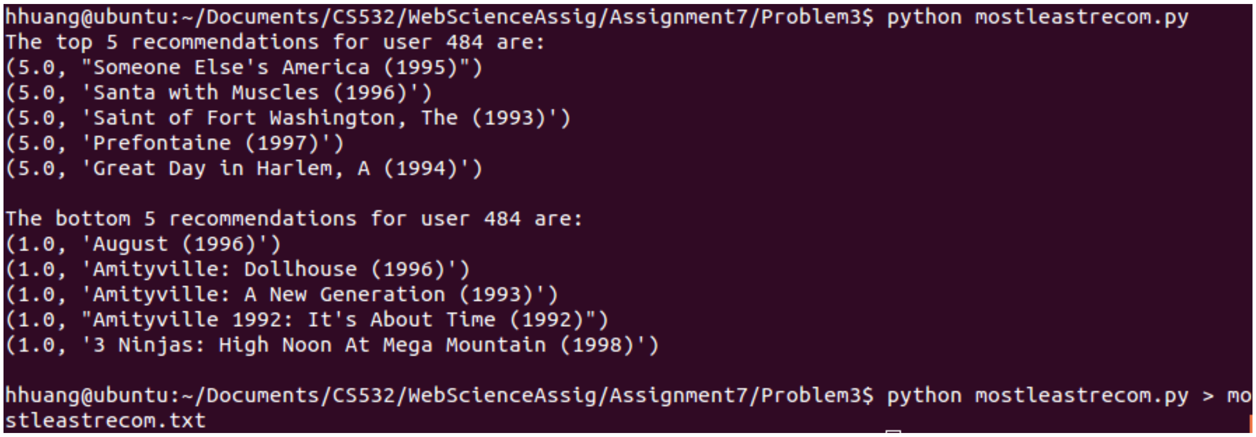
```
    si = {}
    for item in prefs[p1]:
        if item in prefs[p2]:
            si[item] = 1
    # If they are no ratings in common, return 0
    if len(si) == 0:
        return 0
    # Sum calculations
    n = len(si)
    # Sums of all the preferences
    sum1 = sum([prefs[p1][it] for it in si])
    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

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, '484')
print 'The top 5 users who are most correlated to the substitute me are:'
print str(mostMatching[0]) + '\n' + str(mostMatching[1]) + '\n' + str(
    mostMatching[2]) + '\n' + str(mostMatching[3]) + '\n' + str(mostMatching
    [4]) + '\n'

print 'The top 5 users who are least correlated to the substitute me are:'
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

```python
#!/usr/bin/python
from math import sqrt

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

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:]

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,ts)=line.split('\t')
  prefs.setdefault(user,{})
  prefs[user][movies[movieid]]=float(rating)

(toprecommendation, botrecommendations) = getRecommendations(prefs, '484')
print 'The top 5 recommendations for user 484 are:'
print str(toprecommendation[0]) + '\n' + str(toprecommendation[1]) + '\n' + str(toprecommendation[2]) + '\n' + str(toprecommendation[3]) + '\n' + str(toprecommendation[4]) + '\n'
print 'The bottom 5 recommendations for user 484 are:'
print str(botrecommendations[0]) + '\n' + str(botrecommendations[1]) + '\n' + str(botrecommendations[2]) + '\n' + str(botrecommendations[3]) + '\n' + str(botrecommendations[4]) + '\n'
```

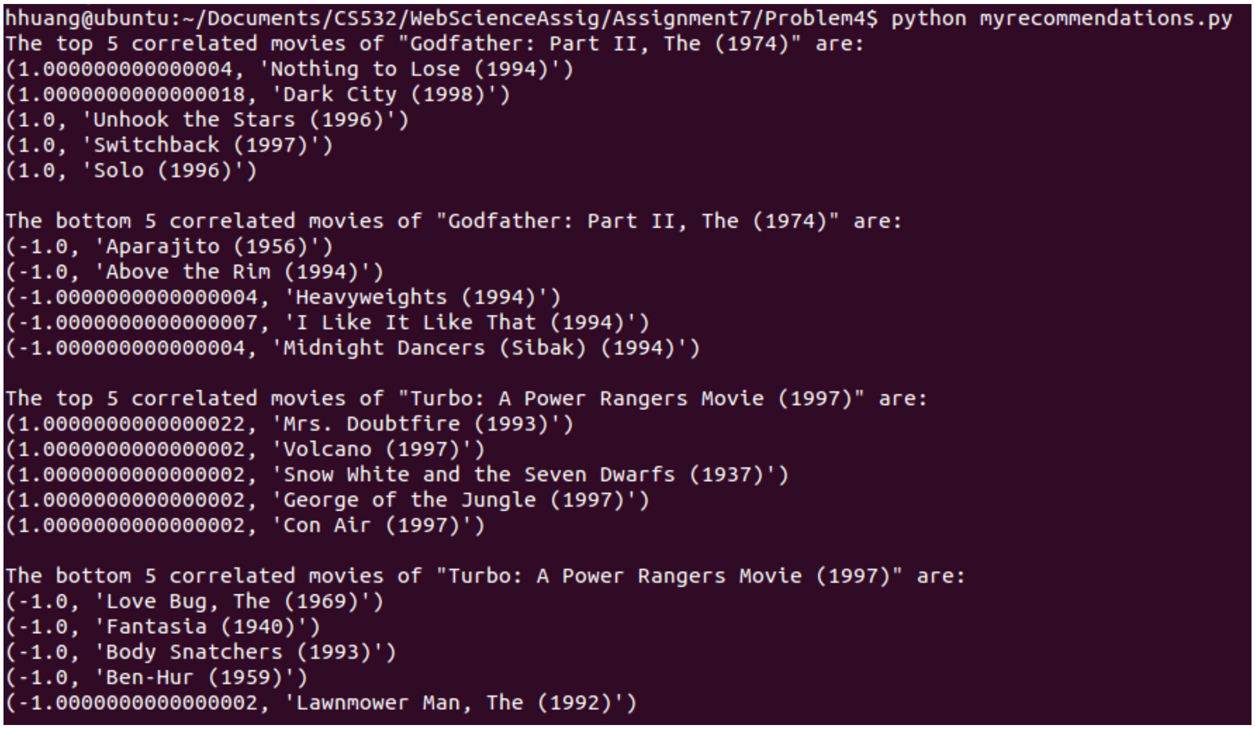
Problem 4

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.

```
#!/usr/bin/python
from math import sqrt

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)
```



```
hhuang@ubuntu:~/Documents/CS532/WebScienceAssig/Assignment7/Problem4$ python myrecommendations.py
The top 5 correlated movies of "Godfather: Part II, The (1974)" are:
(1.000000000000004, 'Nothing to Lose (1994)')
(1.0000000000000018, '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)')
(-1.000000000000004, 'Midnight Dancers (Sibak) (1994)')

The top 5 correlated movies of "Turbo: A Power Rangers Movie (1997)" are:
(1.0000000000000022, '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

```python

  # 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

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:]
```

```


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

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,ts)=line.split('\t')
  prefs.setdefault(user,{})
  prefs[user][movies[movieid]]=float(rating)

movchoice = 'Godfather: Part II, The (1974)'
lestfav = 'Turbo: A Power Rangers Movie (1997)'
remap = transformPrefs(prefs)
(top1, least1) = topMatches(remap,movchoice,similarity=sim_pearson)
(top2, least2) = topMatches(remap,lestfav ,similarity=sim_pearson)

print 'The top 5 correlated movies of "Godfather: Part II, The (1974)" are:'
print str(top1[0]) + '\n' + str(top1[1]) + '\n' + str(top1[2]) + '\n' + str(top1[3]) + '\n' + str(top1[4]) + '\n'
print 'The bottom 5 correlated movies of "Godfather: Part II, The (1974)" are:'
print str(least1[0]) + '\n' + str(least1[1]) + '\n' + str(least1[2]) + '\n' + str(least1[3]) + '\n' + str(least1[4]) + '\n'
print 'The top 5 correlated movies of "Turbo: A Power Rangers Movie (1997)" are:'
print str(top2[0]) + '\n' + str(top2[1]) + '\n' + str(top2[2]) + '\n' + str(top2[3]) + '\n' + str(top2[4]) + '\n'
print 'The bottom 5 correlated movies of "Turbo: A Power Rangers Movie (1997)" are:'
print str(least2[0]) + '\n' + str(least2[1]) + '\n' + str(least2[2]) + '\n' + str(least2[3]) + '\n' + str(least2[4]) + '\n'

#result = calculateSimilarItems(prefs)
```

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.