CS 532: Assignment 7

Dinesh Kumar Paladhi

Spring 2016

Contents

1 Problem 1

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.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|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
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.

Questions (10 points).

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 all'').

This user is the ``substitute you''.
```

1.1 Solution

1. My first task is to find top 3 favorite and bottom 3 favorite films for three users who are closest to me in terms of age,gender and occupation.
2. In order to do this I first collected data from u.data, u.user and u.item. Sample files for all these 3 are shown in fig1, fig2 and fig3.
3. The sample combined data can be seen in fig4.
4. I took this data and then found users who are similar to me in terms of age,gender and occupation and this list can be found in fig5.
5. I got 6 users who are similar to me and I choosed 3 users randomly and their user-id's are 135, 391 and 706.
6. Now for each of this user I need to find out their top 3 favorite films and bottom 3 favorite films using the ratings given by them.
7. I have written code for this and this can be seen in listing1.
8. I got 3 top and bottom favorite films for each user and this can be seen in fig6.
9. I chose user with user-id = "706" as my substitute because my interest matched with him. Even I like star wars and Edge and but I am not interested in Phenomenon and I do not like Game, Fargo and Crash.
10. So user with user-id =1"706" is my substitute.

1.2 Code Listing

```
import json
import re
from math import sqrt

def getData():
        # ratingDataFile = open("ratingData.json","w")
        # ratingData ={}
        # for line in open('u.data'):
                # (user, movieid, rating, ts) = line.split('\t')
                # ratingData['user'] = user.strip()
                # ratingData['movieid'] = movieid.strip()
                # ratingData['rating'] = rating.strip()
                # # ratingData['ts'] = ts.strip()
                # ratingDataFile.write(json.dumps(ratingData)+",\n")

        # moviesDataFile = open("movieData.json","w")
        # movies = {}
        # for line in open('u.item'):
                # movie_id = line.split('|')[0:1][0]
                # movie_name = line.split('|')[1:2][0]
                # movies['movie_id'] = re.sub(r'[^\x00-\x7F]',' ', movie_id)
                # movies['movie_name'] = re.sub(r'[^\x00-\x7F]',' ', movie_name)
                # moviesDataFile.write(json.dumps(movies) + ',\n')


        userDataFile = open("u_data.json","w")
        UserData = {}
        for line in open('u.user'):
                UserData['user_id'] = line.split('|')[0]
                userDetails = {}
                userDetails['age'] = line.split('|')[1]
                userDetails['occupation'] = line.split('|')[3]
                userDetails['gender'] = line.split('|')[2]
                UserData['user_details']= userDetails
                ratingDataFile = open("ratingData.json","r")
                ratingData = json.load(ratingDataFile)
                movieDetails_list = []
                for user1 in ratingData:
                        r_id = user1['user']
                        if UserData['user_id'] == r_id:
                                movieDetails= {}
                                movieDetails['movie_id'] = user1['movieid']
                                movieDetails['movie_rating'] = user1['rating']
                                moviesDataFile = open("movieData.json","r")
                                movieData = json.load(moviesDataFile)
                                for user2 in movieData:
                                        movie_id = user2['movie_id']
                                        if user1['movieid'] == movie_id:
                                                movieDetails['movie_name'] = user2['
                                                    movie_name']
                                                break
                                moviesDataFile.close()
                                movieDetails_list.append(movieDetails)
                ratingDataFile.close()
                UserData['movie_details']= movieDetails_list
                userDataFile.write(json.dumps(UserData) + ',\n')

def getsimilarusers() :
        output_1= open("similarusers.json",'w')
        input_file=open("u_data.json",'r')
        input_data=json.load(input_file)
        for line in input_data :
                if(line['user_details']['gender']=='M' and line['user_details']['age']=='23'
                     and line['user_details']['occupation']=='student') :
                        output_1.write(json.dumps(line) + ',\n')
                        #print line['user_id']
def gettopbot3() :
        input_file=open("similarusers.json",'r')
        input_data=json.load(input_file)
        tcount=1
```

```
        bcount=1

        for line in input_data :
                print "User :"+line['user_id']
                print '\n'
                print "Top 3 favourite films are "
                for film in line['movie_details'] :

                        if(tcount <= 3):
                                if (film['movie_rating'] =='5'):
                                        tcount +=1

                                        print film['movie_name']+','+film['movie_rating']
                for film in line['movie_details'] :

                        if(tcount <= 3):
                                if (film['movie_rating'] =='4'):
                                        tcount +=1

                                        print film['movie_name']+','+film['movie_rating']
                print '\n'
                print "Bottom 3 least favourite films are "
                for film in line['movie_details'] :

                        if(bcount <= 3):
                                if (film['movie_rating'] =='1'):
                                        bcount +=1

                                        print film['movie_name']+','+film['movie_rating']
                for film in line['movie_details'] :

                        if(bcount <= 3):
                                if (film['movie_rating'] =='2'):
                                        bcount +=1

                                        print film['movie_name']+','+film['movie_rating']
                print '\n'
                tcount=1
                bcount=1
                #135,391,706
#706 is my substitute user because I like all the movies highly rated by him and I don't
    like his least rated movies except Fargo

#getsimilarusers()
gettopbot3()
#getData()
```

Listing 1: Python code for getting top 3 and bottom 3 favorite films for 3 users who are closest to me in terms of age,gender and occupation

1.3 Inputs

Sample u.user file

```
196|49|M|writer|55105
197|55|M|technician|75094
198|21|F|student|55414
199|30|M|writer|17604
200|40|M|programmer|93402
201|27|M|writer|E2A4H
202|41|F|educator|60201
203|25|F|student|32301
204|52|F|librarian|10960
205|47|M|lawyer|06371
206|14|F|student|53115
207|39|M|marketing|92037
208|43|M|engineer|01720
209|33|F|educator|85710
210|39|M|engineer|03060
211|66|M|salesman|32605
212|49|F|educator|61401
213|33|M|executive|55345
214|26|F|librarian|11231
215|35|M|programmer|63033
216|22|M|engineer|02215
217|22|M|other|11727
218|37|M|administrator|06513
219|32|M|programmer|43212
220|30|M|librarian|78205
221|19|M|student|20685
222|29|M|programmer|27502
223|19|F|student|47906
224|31|F|educator|43512
225|51|F|administrator|58202
226|28|M|student|92103
227|46|M|executive|60659
228|21|F|student|22003
229|29|F|librarian|22903
230|28|F|student|14476
231|48|M|librarian|01080
232|45|M|scientist|99709
233|38|M|engineer|98682
234|60|M|retired|94702
235|37|M|educator|22973
```

Figure 1: Sample list of user data

Sample u.item file

Figure 2: Sample list of movie data

Sample u.data file


```
[{"user_details": {"gender": "M", "age": "24", "occupation": "technician"}, "user_id": "1", "movie_details": [{"movie_rating
{"user_details": {"gender": "F", "age": "53", "occupation": "other"}, "user_id": "2", "movie_details": [{"movie_rating": "4"
{"user_details": {"gender": "M", "age": "23", "occupation": "writer"}, "user_id": "3", "movie_details": [{"movie_rating": "1
{"user_details": {"gender": "M", "age": "24", "occupation": "technician"}, "user_id": "4", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "33", "occupation": "other"}, "user_id": "5", "movie_details": [{"movie_rating": "3"
{"user_details": {"gender": "M", "age": "42", "occupation": "executive"}, "user_id": "6", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "57", "occupation": "administrator"}, "user_id": "7", "movie_details": [{"movie_rati
{"user_details": {"gender": "M", "age": "36", "occupation": "administrator"}, "user_id": "8", "movie_details": [{"movie_rati
{"user_details": {"gender": "M", "age": "29", "occupation": "student"}, "user_id": "9", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "53", "occupation": "lawyer"}, "user_id": "10", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "39", "occupation": "other"}, "user_id": "11", "movie_details": [{"movie_rating": "4
{"user_details": {"gender": "F", "age": "28", "occupation": "other"}, "user_id": "12", "movie_details": [{"movie_rating": "4
{"user_details": {"gender": "M", "age": "47", "occupation": "educator"}, "user_id": "13", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "45", "occupation": "scientist"}, "user_id": "14", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "49", "occupation": "educator"}, "user_id": "15", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "21", "occupation": "entertainment"}, "user_id": "16", "movie_details": [{"movie_rat
{"user_details": {"gender": "M", "age": "30", "occupation": "programmer"}, "user_id": "17", "movie_details": [{"movie_rating
{"user_details": {"gender": "F", "age": "35", "occupation": "other"}, "user_id": "18", "movie_details": [{"movie_rating": "3
{"user_details": {"gender": "M", "age": "40", "occupation": "librarian"}, "user_id": "19", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "42", "occupation": "homemaker"}, "user_id": "20", "movie_details": [{"movie_rating"
{"user_details": {"gender": "M", "age": "26", "occupation": "writer"}, "user_id": "21", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "25", "occupation": "writer"}, "user_id": "22", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "30", "occupation": "artist"}, "user_id": "23", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "21", "occupation": "artist"}, "user_id": "24", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "39", "occupation": "engineer"}, "user_id": "25", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "49", "occupation": "engineer"}, "user_id": "26", "movie_details": [{"movie_rating":
{"user_details": {"gender": "F", "age": "40", "occupation": "librarian"}, "user_id": "27", "movie_details": [{"movie_rating"
{"user_details": {"gender": "M", "age": "32", "occupation": "writer"}, "user_id": "28", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "41", "occupation": "programmer"}, "user_id": "29", "movie_details": [{"movie_rating
{"user_details": {"gender": "M", "age": "7", "occupation": "student"}, "user_id": "30", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "24", "occupation": "artist"}, "user_id": "31", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "28", "occupation": "student"}, "user_id": "32", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "33", "movie_details": [{"movie_rating":
{"user_details": {"gender": "F", "age": "38", "occupation": "administrator"}, "user_id": "34", "movie_details": [{"movie_rat
{"user_details": {"gender": "F", "age": "20", "occupation": "homemaker"}, "user_id": "35", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "19", "occupation": "student"}, "user_id": "36", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "37", "movie_details": [{"movie_rating":
```


Figure 3: Sample list of users and their rating for each movie

1.4 Outputs

Sample combined data

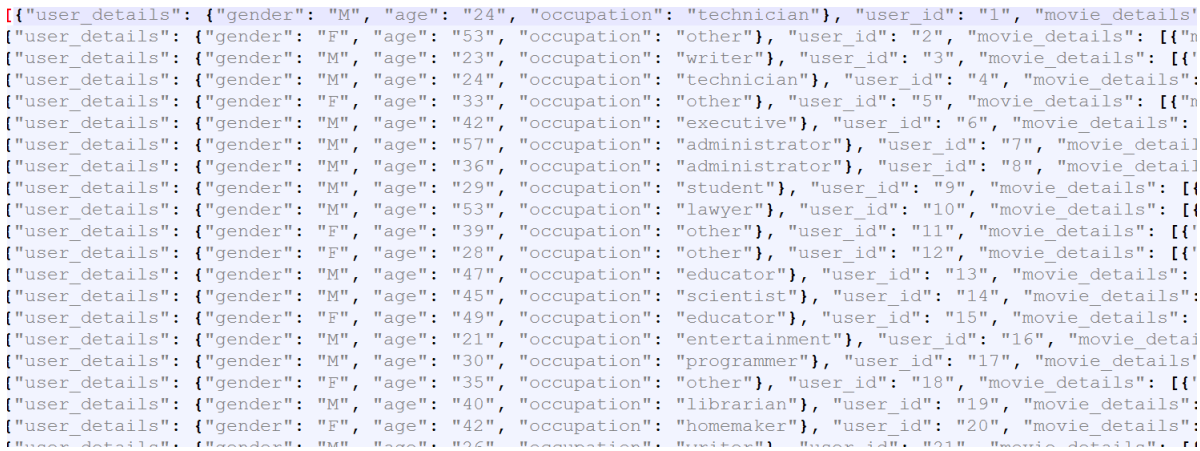

Figure 4: Sample data combining all the three data files shown above

Similar users data


```
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "33", "movie_details": [{"movie_rating": "3", "m
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "37", "movie_details": [{"movie_rating": "4", "m
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "135", "movie_details": [{"movie_rating": "4", "
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "66", "movie_details": [{"movie_rating": "4", "m
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "391", "movie_details": [{"movie_rating": "3", "
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "408", "movie_details": [{"movie_rating": "3", "
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "706", "movie_details": [{"movie_rating": "3", "
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "838", "movie_details": [{"movie_rating": "3", "
```


Figure 5: List of users similar to me in terms of age,gender and occupation

Output file

```
atria:~/Webscience/cs532-s16/Assignment 7/1> python mov_1.py
User :135


Top 3 favourite films are
Silence of the Lambs, The (1991),5
Taxi Driver (1976),4
Liar Liar (1997),4


Bottom 3 least favourite films are
Tales from the Hood (1995),1
Highlander (1986),2
Star Trek III: The Search for Spock (1984),2


User :391


Top 3 favourite films are
Pulp Fiction (1994),5
Brothers McMullen, The (1995),5
Apocalypse Now (1979),5


Bottom 3 least favourite films are
Mimic (1997),1
White Squall (1996),2
Terminator, The (1984),2


User :706


Top 3 favourite films are
Star Wars (1977),5
Edge, The (1997),5
Phenomenon (1996),5


Bottom 3 least favourite films are
Game, The (1997),1
Fargo (1996),1
Crash (1996),1


atria:~/Webscience/cs532-s16/Assignment 7/1>
```

Figure 6: File shows 5 top favorite and least favorite movies for each of the 3 users who are closest to me

2 Problem 2

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

2.1 Solution

1. In this question I need to find out 5 users who are most and least correlated to my substitute.
2. For doing this I used some function from recommendations.py as reference taken from a book called "Programming Collective Intelligence."
3. I used u.data file and sent it as input to my program to get the solution. The program can be found in listing2.
4. For doing this I need to find the sim pearson's coefficient which for each user.
5. If the coefficient for each user is 1 or nearer to 1, then that user is most correlated to my substitute and if the coefficient is negative then that user is least coefficient to my substitute.
6. I found the top 5 most correlated users and bottom 5 least correlated users which can be seen in fig8.
7. First column represent the value of coefficient followed by user-id.

2.2 Code Listing

```
1
2  from math import sqrt
3
4  def sim_pearson(prefs,p1,p2):
5
6    # Get the list of mutually rated items
7    si={}
8    for item in prefs[p1]:
9      if item in prefs[p2]: si[item]=1
10
11   # if they are no ratings in common, return 0
12   if len(si)==0: return 0
13
14   # Sum calculations
15   n=len(si)
16
17   # Sums of all the preferences
18   sum1=sum([int(prefs[p1][it]) for it in si])
19   sum2=sum([int(prefs[p2][it]) for it in si])
20
21   # Sums of the squares
22   sum1Sq=sum([pow(int(prefs[p1][it]),2) for it in si])
23   sum2Sq=sum([pow(int(prefs[p2][it]),2) for it in si])
24
25   # Sum of the products
26   pSum=sum([int(prefs[p1][it])*int(prefs[p2][it]) for it in si])
27
28   # Calculate r (Pearson score)
29   num=pSum-(sum1*sum2/n)
30   den=sqrt((sum1Sq-pow(sum1,2)/n)*(sum2Sq-pow(sum2,2)/n))
31   if den==0: return 0
32
33   r=num/den
34
35   return r
36
37
38
39 input = open('u.data','r')
40
41 pref = {}
42 user_count =0
43 for line in input:
44         (user_id,item_id ,rating,ts) = line.split()
45         if user_id in pref:
46                 pref[user_id][item_id] = rating
47         else:
48                 user_count = user_count + 1
49                 pref[user_id] = {}
50
51 result = []
52 for i in range(1,user_count+1):
53         cofficient =  sim_pearson(pref,'706',str(i))
54         result.append((cofficient, i))
55 result.sort()
56 result.reverse()
57 a=result[:5]
58 print "Most Correlated"
59 for i in a:
60         print i
61 print '\n'
62 result.reverse()
63 print "Least Correlated"
64 b=result[:5]
65 for i in b:
66         print i
```

Listing 2: Python Code for getting finding gender of each follower

2.3 Inputs

Sample u.data file


```
[{"user_details": {"gender": "M", "age": "24", "occupation": "technician"}, "user_id": "1", "movie_details": [{"movie_rating
{"user_details": {"gender": "F", "age": "53", "occupation": "other"}, "user_id": "2", "movie_details": [{"movie_rating": "4"
{"user_details": {"gender": "M", "age": "23", "occupation": "writer"}, "user_id": "3", "movie_details": [{"movie_rating": "1
{"user_details": {"gender": "M", "age": "24", "occupation": "technician"}, "user_id": "4", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "33", "occupation": "other"}, "user_id": "5", "movie_details": [{"movie_rating": "3"
{"user_details": {"gender": "M", "age": "42", "occupation": "executive"}, "user_id": "6", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "57", "occupation": "administrator"}, "user_id": "7", "movie_details": [{"movie_rati
{"user_details": {"gender": "M", "age": "36", "occupation": "administrator"}, "user_id": "8", "movie_details": [{"movie_rati
{"user_details": {"gender": "M", "age": "29", "occupation": "student"}, "user_id": "9", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "53", "occupation": "lawyer"}, "user_id": "10", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "39", "occupation": "other"}, "user_id": "11", "movie_details": [{"movie_rating": "4
{"user_details": {"gender": "F", "age": "28", "occupation": "other"}, "user_id": "12", "movie_details": [{"movie_rating": "4
{"user_details": {"gender": "M", "age": "47", "occupation": "educator"}, "user_id": "13", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "45", "occupation": "scientist"}, "user_id": "14", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "49", "occupation": "educator"}, "user_id": "15", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "21", "occupation": "entertainment"}, "user_id": "16", "movie_details": [{"movie_rat
{"user_details": {"gender": "M", "age": "30", "occupation": "programmer"}, "user_id": "17", "movie_details": [{"movie_rating
{"user_details": {"gender": "F", "age": "35", "occupation": "other"}, "user_id": "18", "movie_details": [{"movie_rating": "3
{"user_details": {"gender": "M", "age": "40", "occupation": "librarian"}, "user_id": "19", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "42", "occupation": "homemaker"}, "user_id": "20", "movie_details": [{"movie_rating"
{"user_details": {"gender": "M", "age": "26", "occupation": "writer"}, "user_id": "21", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "25", "occupation": "writer"}, "user_id": "22", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "30", "occupation": "artist"}, "user_id": "23", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "21", "occupation": "artist"}, "user_id": "24", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "39", "occupation": "engineer"}, "user_id": "25", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "49", "occupation": "engineer"}, "user_id": "26", "movie_details": [{"movie_rating":
{"user_details": {"gender": "F", "age": "40", "occupation": "librarian"}, "user_id": "27", "movie_details": [{"movie_rating"
{"user_details": {"gender": "M", "age": "32", "occupation": "writer"}, "user_id": "28", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "41", "occupation": "programmer"}, "user_id": "29", "movie_details": [{"movie_rating
{"user_details": {"gender": "M", "age": "7", "occupation": "student"}, "user_id": "30", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "24", "occupation": "artist"}, "user_id": "31", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "28", "occupation": "student"}, "user_id": "32", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "33", "movie_details": [{"movie_rating":
{"user_details": {"gender": "F", "age": "38", "occupation": "administrator"}, "user_id": "34", "movie_details": [{"movie_rat
{"user_details": {"gender": "F", "age": "20", "occupation": "homemaker"}, "user_id": "35", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "19", "occupation": "student"}, "user_id": "36", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "37", "movie_details": [{"movie_rating":
```


Figure 7: Sample list of users and their rating for each movie

2.4 Output

output file

```
Most Correlated
(1.1547005383792517, 351)
(1.01418510567422, 926)
(1.0, 879)
(1.0, 785)
(1.0, 706)


Least Correlated
(-1.0, 420)
(-1.0, 662)
(-0.9486832980505138, 35)
(-0.9486832980505138, 139)
(-0.9486832980505138, 559)
atria:~/Webscience/cs532-s16/Assignment 7/2>
```

Figure 8: File shows 5 users with their user id's on right side who are most correlated and least correlated to my substitute

3 Problem 3

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

3.1 Solution

1. Here I need to compute the ratings for the films my substitute has not seen. Now I need find out top 5 and bottom 5 recommendations for films that my substitute should see.
2. The bottom 5 recommendations are the films my substitute hate to see.
3. I have found these recommendations using a function called "getRecommendations" from recommendations.py.
4. The program for this question can be found in listing3.
5. This function gives me a list of movies and their rating. These are recommendations for my substitute.
6. But all these are not considered as output, only the top 5 that is movies with rating as 5 are chosen and bottom 5 that is movies with rating as 1 are chosen.
7. So these top 5 movies are recommended to my substitute expecting that he/she likes them.
8. The output file showing movie names and their ratings can be seen in fig11.

3.2 Code Listing

```
1
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([int(prefs[p1][it]) for it in si])
18   sum2=sum([int(prefs[p2][it]) for it in si])
19
20   # Sums of the squares
21   sum1Sq=sum([pow(int(prefs[p1][it]),2) for it in si])
22   sum2Sq=sum([pow(int(prefs[p2][it]),2) for it in si])
23
24   # Sum of the products
25   pSum=sum([int(prefs[p1][it])*int(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 def getRecommendations(prefs,person,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]+=int(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
63
64
65 def getmovienames(list) :
66         input = open('u.item','r')
67         movie_name=[]
68         movie_ids=[]
69         movie_ratings=[]
70         for lane in list:
```

```
                movie_ids.append(lane[1])
                movie_ratings.append(lane[0])
        for next in input:
                next=next.split("|")
                id=next[0]
                name=next[1]
                #print list
                if id in movie_ids :
                        movie_name.append(name)
        return movie_name,movie_ratings
input = open('u.data','r')

pref = {}
user_count =0
for line in input:
        (user_id,item_id ,rating,ts) = line.split()
        if user_id in pref:
                pref[user_id][item_id] = rating
        else:
                user_count = user_count + 1
                pref[user_id] = {}

result =  getRecommendations(pref,'706',similarity=sim_pearson)
bot_result = result[1 : 5]
top_result = result[-5 :]
print "Most recommended Movies and their ratings"
most,rating=getmovienames(bot_result)
for i in range(0,len(most)) :
        print  most[i],rating[i]
print '\n'
Least,rating=getmovienames(top_result)
print "Least recommended Movies and their ratings"
for k in range(0,len(Least)):
        print Least[k],rating[k]
```

Listing 3: Python Code for generating top 5 and least 5 movie recommendations that my substitute should see

3.3 Inputs

Sample u.data file


```
[{"user_details": {"gender": "M", "age": "24", "occupation": "technician"}, "user_id": "1", "movie_details": [{"movie_rating
{"user_details": {"gender": "F", "age": "53", "occupation": "other"}, "user_id": "2", "movie_details": [{"movie_rating": "4"
{"user_details": {"gender": "M", "age": "23", "occupation": "writer"}, "user_id": "3", "movie_details": [{"movie_rating": "1
{"user_details": {"gender": "M", "age": "24", "occupation": "technician"}, "user_id": "4", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "33", "occupation": "other"}, "user_id": "5", "movie_details": [{"movie_rating": "3"
{"user_details": {"gender": "M", "age": "42", "occupation": "executive"}, "user_id": "6", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "57", "occupation": "administrator"}, "user_id": "7", "movie_details": [{"movie_rati
{"user_details": {"gender": "M", "age": "36", "occupation": "administrator"}, "user_id": "8", "movie_details": [{"movie_rati
{"user_details": {"gender": "M", "age": "29", "occupation": "student"}, "user_id": "9", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "53", "occupation": "lawyer"}, "user_id": "10", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "39", "occupation": "other"}, "user_id": "11", "movie_details": [{"movie_rating": "4
{"user_details": {"gender": "F", "age": "28", "occupation": "other"}, "user_id": "12", "movie_details": [{"movie_rating": "4
{"user_details": {"gender": "M", "age": "47", "occupation": "educator"}, "user_id": "13", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "45", "occupation": "scientist"}, "user_id": "14", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "49", "occupation": "educator"}, "user_id": "15", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "21", "occupation": "entertainment"}, "user_id": "16", "movie_details": [{"movie_rat
{"user_details": {"gender": "M", "age": "30", "occupation": "programmer"}, "user_id": "17", "movie_details": [{"movie_rating
{"user_details": {"gender": "F", "age": "35", "occupation": "other"}, "user_id": "18", "movie_details": [{"movie_rating": "3
{"user_details": {"gender": "M", "age": "40", "occupation": "librarian"}, "user_id": "19", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "42", "occupation": "homemaker"}, "user_id": "20", "movie_details": [{"movie_rating"
{"user_details": {"gender": "M", "age": "26", "occupation": "writer"}, "user_id": "21", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "25", "occupation": "writer"}, "user_id": "22", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "30", "occupation": "artist"}, "user_id": "23", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "21", "occupation": "artist"}, "user_id": "24", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "39", "occupation": "engineer"}, "user_id": "25", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "49", "occupation": "engineer"}, "user_id": "26", "movie_details": [{"movie_rating":
{"user_details": {"gender": "F", "age": "40", "occupation": "librarian"}, "user_id": "27", "movie_details": [{"movie_rating"
{"user_details": {"gender": "M", "age": "32", "occupation": "writer"}, "user_id": "28", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "41", "occupation": "programmer"}, "user_id": "29", "movie_details": [{"movie_rating
{"user_details": {"gender": "M", "age": "7", "occupation": "student"}, "user_id": "30", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "24", "occupation": "artist"}, "user_id": "31", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "28", "occupation": "student"}, "user_id": "32", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "33", "movie_details": [{"movie_rating":
{"user_details": {"gender": "F", "age": "38", "occupation": "administrator"}, "user_id": "34", "movie_details": [{"movie_rat
{"user_details": {"gender": "F", "age": "20", "occupation": "homemaker"}, "user_id": "35", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "19", "occupation": "student"}, "user_id": "36", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "37", "movie_details": [{"movie_rating":
```


Figure 9: Sample list of users and their rating for each movie

Sample u.item file

Figure 10: Sample list of movie data

3.4 Outputs

Output file

```
sirius:~/Webscience/cs532-s16/Assignment 7/3> python mov_3.py
Most recommended Movies and their ratings
Thirty-Two Short Films About Glenn Gould (1993) 5.0
Great Day in Harlem, A (1994) 5.0
Two or Three Things I Know About Her (1966) 5.0
Someone Else's America (1995) 5.0


Least recommended Movies and their ratings
Theodore Rex (1995) 1.0
Bloodsport 2 (1995) 1.0
Calendar Girl (1993) 1.0
Zeus and Roxanne (1997) 1.0
Endless Summer 2, The (1994) 1.0
sirius:~/Webscience/cs532-s16/Assignment 7/3>
```

Figure 11: File contains top 5 and least 5 movie recommendations that my substitute should see

4 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?
```

4.1 Solution

1. I need to choose my top favorite and least favorite movies from u.item file. For each of this film I need to find out top 5 most correlated and bottom 5 least correlated films and for that I need to give my comments.
2. I have chosen "Die Hard (1988)" as my top favorite film and "Kazaam (1996)" as my least favorite film.
3. Now I have used functions from recommendations.py to compute the top 5 correlated and least 5 correlated films.
4. I have used loadMovieLens, transformPrefs and topMatches functions from recommendations.py.
5. This in turn uses sim pearson's coefficient for getting correlation.
6. If the coefficient for each film is 1 or nearer to 1, then that film is most correlated to my film and if the coefficient is negative then that user is least correlated to my film.
7. The program for this can be found in listing5.
8. Recommendations.py program can be seen in listing4.
9. The output file showing sim pearson's coefficient and movie names can be seen in the fig14.

4.2 Code Listing

Code Listing 1

```
# 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

# 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

```

```
69    r=num/den
70
71    return r
72
73  # Returns the best matches for person from the prefs dictionary.
74  # Number of results and similarity function are optional params.
75  def topMatches(prefs,person,n=5,similarity=sim_pearson):
76    scores=[(similarity(prefs,person,other),other)
77                    for other in prefs if other!=person]
78    scores.sort()
79    scores.reverse()
80    return scores[0:n]
81
82  # Gets recommendations for a person by using a weighted average
83  # of every other user's rankings
84  def getRecommendations(prefs,person,similarity=sim_pearson):
85    totals={}
86    simSums={}
87    for other in prefs:
88      # don't compare me to myself
89      if other==person: continue
90      sim=similarity(prefs,person,other)
91
92      # ignore scores of zero or lower
93      if sim<=0: continue
94      for item in prefs[other]:
95
96        # only score movies I haven't seen yet
97        if item not in prefs[person] or prefs[person][item]==0:
98          # Similarity * Score
99          totals.setdefault(item,0)
100         totals[item]+=prefs[other][item]*sim
101         # Sum of similarities
102         simSums.setdefault(item,0)
103         simSums[item]+=sim
104
105   # Create the normalized list
106   rankings=[(total/simSums[item],item) for item,total in totals.items()]
107
108   # Return the sorted list
109   rankings.sort()
110   rankings.reverse()
111   return rankings
112
113 def transformPrefs(prefs):
114   result={}
115   for person in prefs:
116     for item in prefs[person]:
117       result.setdefault(item,{})
118
119       # Flip item and person
120       result[item][person]=prefs[person][item]
121   return result
122
123
124 def calculateSimilarItems(prefs,n=10):
125   # Create a dictionary of items showing which other items they
126   # are most similar to.
127   result={}
128   # Invert the preference matrix to be item-centric
129   itemPrefs=transformPrefs(prefs)
130   c=0
131   for item in itemPrefs:
132     # Status updates for large datasets
133     c+=1
134     if c%100==0: print "%d / %d" % (c,len(itemPrefs))
135     # Find the most similar items to this one
136     scores=topMatches(itemPrefs,item,n=n,similarity=sim_pearson)
137     result[item]=scores
138   return result
139
140 def getRecommendedItems(prefs,itemMatch,user):
```

```
141     userRatings=prefs[user]
142     scores={}
143     totalSim={}
144     # Loop over items rated by this user
145     for (item,rating) in userRatings.items( ):
146
147       # Loop over items similar to this one
148       for (similarity,item2) in itemMatch[item]:
149
150         # Ignore if this user has already rated this item
151         if item2 in userRatings: continue
152         # Weighted sum of rating times similarity
153         scores.setdefault(item2,0)
154         scores[item2]+=similarity*rating
155         # Sum of all the similarities
156         totalSim.setdefault(item2,0)
157         totalSim[item2]+=similarity
158
159     # Divide each total score by total weighting to get an average
160     rankings=[(score/totalSim[item],item) for item,score in scores.items( )]
161
162     # Return the rankings from highest to lowest
163     rankings.sort( )
164     rankings.reverse( )
165     return rankings
166
167 def loadMovieLens():
168     # Get movie titles
169     movies={}
170     for line in open('u.item'):
171       (id,title)=line.split('|')[0:2]
172       movies[id]=title
173
174     # Load data
175     prefs={}
176     for line in open('u.data'):
177       (user,movieid,rating,ts)=line.split('\t')
178       prefs.setdefault(user,{})
179       prefs[user][movies[movieid]]=float(rating)
180     return prefs
```

Listing 4: Python code with various functions. This is a reference code.

Code Listing 2

```
#!/usr/local/bin/python
import sys
import recommendations

if __name__ == '__main__':
    fav_moviename   = "Die Hard (1988)"
    hate_moviename = "Kazaam (1996)"
    prefs = recommendations.loadMovieLens()
    itemPrefs = recommendations.transformPrefs(prefs)
    fav_results    = recommendations.topMatches(itemPrefs,fav_moviename,2000)
    hate_results    = recommendations.topMatches(itemPrefs,hate_moviename,2000)

print "Most 5 correlated for my top favourite movie"
for i in fav_results[0:5]:
        print i[0],i[1]
print '\n'
print "Least 5 correlated for my top favourite movie"
fav_results.reverse()
for i in fav_results[0:5]:
        print i[0],i[1]
print '\n'
print "Most 5 correlated for my least favourite movie"
for i in hate_results[0:5]:
        print i[0],i[1]
print '\n'
print "Least 5 correlated for my least favourite movie"
hate_results.reverse()
for i in hate_results[0:5]:
        print i[0],i[1]
```

Listing 5: Python code for generating top 5 and least 5 correlated films for my top favorite and least favorite film

4.3 Inputs

Sample u.item file

Figure 12: Sample list of movie data

Sample u.data file


```
[{"user_details": {"gender": "M", "age": "24", "occupation": "technician"}, "user_id": "1", "movie_details": [{"movie_rating
{"user_details": {"gender": "F", "age": "53", "occupation": "other"}, "user_id": "2", "movie_details": [{"movie_rating": "4"
{"user_details": {"gender": "M", "age": "23", "occupation": "writer"}, "user_id": "3", "movie_details": [{"movie_rating": "1
{"user_details": {"gender": "M", "age": "24", "occupation": "technician"}, "user_id": "4", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "33", "occupation": "other"}, "user_id": "5", "movie_details": [{"movie_rating": "3"
{"user_details": {"gender": "M", "age": "42", "occupation": "executive"}, "user_id": "6", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "57", "occupation": "administrator"}, "user_id": "7", "movie_details": [{"movie_rati
{"user_details": {"gender": "M", "age": "36", "occupation": "administrator"}, "user_id": "8", "movie_details": [{"movie_rati
{"user_details": {"gender": "M", "age": "29", "occupation": "student"}, "user_id": "9", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "53", "occupation": "lawyer"}, "user_id": "10", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "39", "occupation": "other"}, "user_id": "11", "movie_details": [{"movie_rating": "4
{"user_details": {"gender": "F", "age": "28", "occupation": "other"}, "user_id": "12", "movie_details": [{"movie_rating": "4
{"user_details": {"gender": "M", "age": "47", "occupation": "educator"}, "user_id": "13", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "45", "occupation": "scientist"}, "user_id": "14", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "49", "occupation": "educator"}, "user_id": "15", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "21", "occupation": "entertainment"}, "user_id": "16", "movie_details": [{"movie_rat
{"user_details": {"gender": "M", "age": "30", "occupation": "programmer"}, "user_id": "17", "movie_details": [{"movie_rating
{"user_details": {"gender": "F", "age": "35", "occupation": "other"}, "user_id": "18", "movie_details": [{"movie_rating": "3
{"user_details": {"gender": "M", "age": "40", "occupation": "librarian"}, "user_id": "19", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "42", "occupation": "homemaker"}, "user_id": "20", "movie_details": [{"movie_rating"
{"user_details": {"gender": "M", "age": "26", "occupation": "writer"}, "user_id": "21", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "25", "occupation": "writer"}, "user_id": "22", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "30", "occupation": "artist"}, "user_id": "23", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "21", "occupation": "artist"}, "user_id": "24", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "39", "occupation": "engineer"}, "user_id": "25", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "49", "occupation": "engineer"}, "user_id": "26", "movie_details": [{"movie_rating":
{"user_details": {"gender": "F", "age": "40", "occupation": "librarian"}, "user_id": "27", "movie_details": [{"movie_rating"
{"user_details": {"gender": "M", "age": "32", "occupation": "writer"}, "user_id": "28", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "41", "occupation": "programmer"}, "user_id": "29", "movie_details": [{"movie_rating
{"user_details": {"gender": "M", "age": "7", "occupation": "student"}, "user_id": "30", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "M", "age": "24", "occupation": "artist"}, "user_id": "31", "movie_details": [{"movie_rating": "
{"user_details": {"gender": "F", "age": "28", "occupation": "student"}, "user_id": "32", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "33", "movie_details": [{"movie_rating":
{"user_details": {"gender": "F", "age": "38", "occupation": "administrator"}, "user_id": "34", "movie_details": [{"movie_rat
{"user_details": {"gender": "F", "age": "20", "occupation": "homemaker"}, "user_id": "35", "movie_details": [{"movie_rating"
{"user_details": {"gender": "F", "age": "19", "occupation": "student"}, "user_id": "36", "movie_details": [{"movie_rating":
{"user_details": {"gender": "M", "age": "23", "occupation": "student"}, "user_id": "37", "movie_details": [{"movie_rating":
```


Figure 13: Sample list of users and their rating for each movie

4.4 Output

Output file

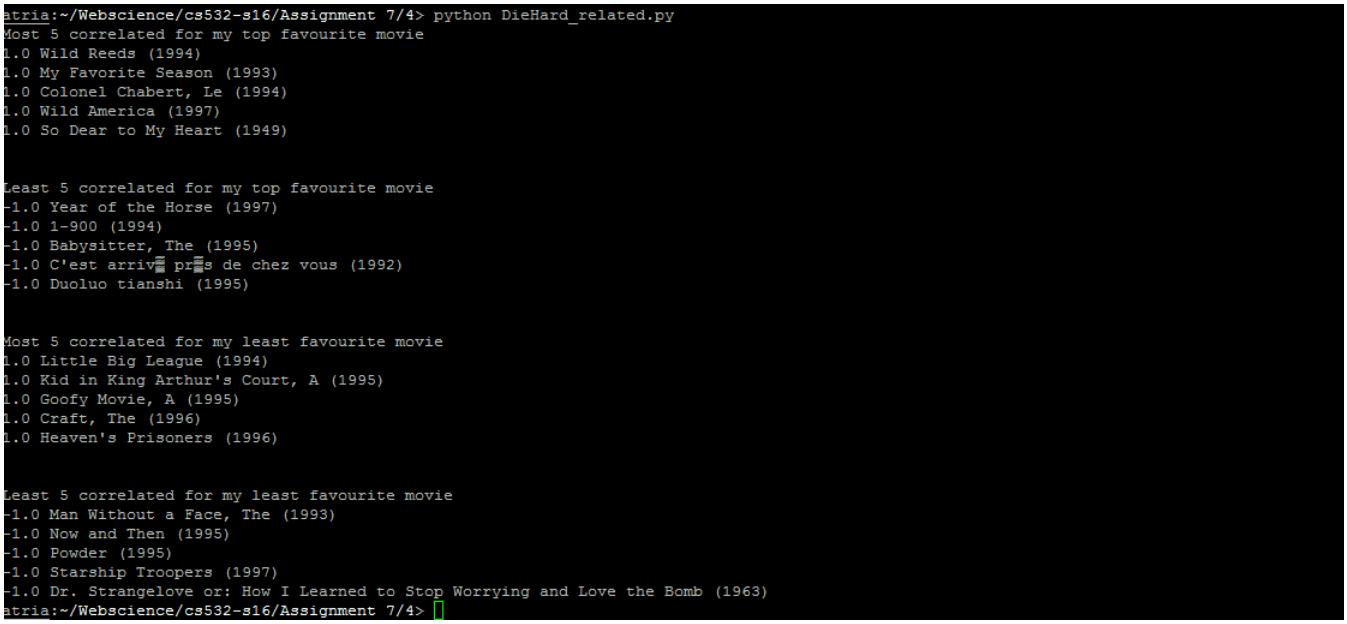

Figure 14: File contains top 5 and least 5 correlated films for my top favorite and least favorite film

Bibliography

[1] K.Arthur Endsley, Recommendation.py, https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter2/recommendations.py, 2014

[2] GroupLens,Movielens 100k dataset,http://grouplens.org/datasets/movielens/100k/, 2016