CS 432/532: Web Science

Spring 2017

Assignment 7

Michael Micros

Instructor: Michael L. Nelson

Old Dominon University
Norfolk, Virginia

April 6, 2017

Problem 1: Finding the "substitute you"

It is important to note that for all the questions of this assignment sample code from the "Collective Intelligence Programming" book was used. Therefore all functions for calculating correlations, providing recommendations and loading in data from MovieLens dataset.

In order to find a user that will represent my taste in movies, it was first necessary to filter the users and isolate the users that were most simillar to me. After reading through all the users in the "u.users" file, I created a list of all users that were 24 years old, had 'student' as occupation, and were male.

```
1  #! /usr/bin/python
2
3  import recommendations as re
4
5  data = re.loadMovieLens()
6  f = open("Alter Egos",'w')
7
8  alterEgos = []
9  for line in open('./data/u.user'):
10     u = ( id, age, gender, occupation, zip)=line.split('|')[0:5]
11
12     if (u[2]=='M' and u[1]=='24' and u[3]=='student') :
13         alterEgos.append(u)
14
15 for i in alterEgos[0:3]:
16     f.write("\n\n-------------------------------\n")
```

Figure 1: Code for filtering users by age,gender and occupation

For 3 users from the list I displayed their highest rated and lowest rated films and selected the user with whom I identified the most. The results for the 3 users are writen in a text file named "Alter Egos". The user I selected as the "substitute me" in the list was User 73. Though his top picks were not my favorite films, User 73 seemed to have the best taste out of all the other users.

```
3  -------------------------------
4  User : ['73', '24', 'M', 'student']
5
6  Top : ('Shawshank Redemption, The (1994)', 5.0)
7  ('Delicatessen (1991)', 5.0)
8  ('Godfather: Part II, The (1974)', 5.0)
9  ('Clockwork Orange, A (1971)', 5.0)
10 ('Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)', 5.0)
11
12 Bottom :('Home Alone (1990)', 1.0)
13 ('Home Alone 3 (1997)', 1.0)
14 ('Jurassic Park (1993)', 2.0)
15 ('Saint, The (1997)', 2.0)
16 ('Toy Story (1995)', 2.0)
17
```

Figure 2: User 73's highest and lowest rated films.

Problem 2: Most and Least correlated Users to "me"

In order to find the 5 most and least correlated users to the "substitute me" I used the topMathes() function provided in the "Collective Intelligence" book. This functions computes the Pearson correlation between a desired User and all the users in the database. The only problem with this function is that it computes the correlation based on common rated movies without taking into account how many films both users have seen and rated. This means that if 2 users have only one common rated film and gave it the same score they will have a perfect correlation. That obviously means are results will be quite unreliable. To avoid this problem I modified the code in the sim_pearson() function to compute the correlation for users that had more than 20 rated movies in common. The results were stored in the file "Recommended Users" file.

```
# Returns the Pearson correlation coefficient for p1
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

  # Find the number of elements
  n=len(si)

  # if they are no ratings in common, return 0
  if n<20: return 0
  #if n==0: return 0

  # Add up all the preferences
```

Figure 3: Modification in sim_pearson() function. Compute correlation between user that have more than 20 films in common.

```
3  import recommendations as re
4  data = re.loadMovieLens()
5
6
7  matches = re.topMatches(data,'73')
8  top = matches[0:5]
9
10 f = open("Recommended Users",'w')
11 f.write("-----------------------------\n")
12 f.write("TOP 5 (Positive Correlation)\n")
13 f.write("-----------------------------\n")
14
15
16 f.write("\nUSER\tCorr\tCommon Rated Movies")
17
18
19 for i in top:
```

Figure 4: Code from part2.py

```
1  ----------------------------
2  TOP 5 (Positive Correlation)
3  ----------------------------
4
5  USER    Corr    Common Rated Movies
6  397             0.81    24
7  360             0.75    20
8  177             0.69    30
9  601             0.69    24
10 249             0.69    27
11
12 ------------------------------
13 BOTTOM 5 (Negative Correlation)
14 ------------------------------
15
16 USER    Corr    Common Rated Movies
17 109             -0.43   28
18 468             -0.31   21
19 454             -0.28   32
20 399             -0.28   30
21 125             -0.28   21
```

Figure 5: Top 5 and Bottom 5 correlated Users

Problem 3: Top 5 and Bottom 5 recommendations

In order to generate ratings for the films I have not yet seen, the getRecommendations() function was used for User 73 (my alter ego). This function goes through all the users that I am positively correlated to and finds which films they have rated that I have not seen. Greater weight is given to users that are more correlated to me. Finally, an average of all the scored is computed, which is the predicted rating for the unseen film.

I believe this method could be modified to include also the users that I was negatively correlated to. If someone I have completely opposite taste with rates a movie terribly, I might enjoy it. The way the getRecommendation() function currently works does not account for such cases.

The results for the top and bottom 5 recommendations are saved in the "Best & Worst Recommendations" file.

```
1  ----------------------------
2  Top 5 Recommendations
3  ----------------------------
4  Predicted Rating  Movie
5  5.0               Tough and Deadly (1995)
6  5.0               They Made Me a Criminal (1939)
7  5.0               Star Kid (1997)
8  5.0               Someone Else's America (1995)
9  5.0               Santa with Muscles (1996)
10
11 ----------------------------
12 Bottom 5 Recommendations
13 ----------------------------
14 Predicted Rating  Movie
15 1.0               3 Ninjas: High Noon At Mega Mountain (1998)
16 1.0               Amityville 1992: It's About Time (1992)
17 1.0               Amityville 3-D (1983)
18 1.0               Amityville Curse, The (1990)
19 1.0               Amityville: A New Generation (1993)
```

Figure 6: Top 5 and Bottom 5 correlated Users

```python
#! /usr/bin/python

import recommendations as re
data = re.loadMovieLens()

itemsim=re.getRecommendations(data,'73')

f = open("Best & Worst Recommendations",'w')

f.write("----------------------------\n")
f.write("Top 5 Recommendations\n")
f.write("----------------------------\n")

f.write("Predicted Rating\tMovie")
for i in itemsim[0:5]:
    f.write("\n" + str(round(i[0],2)) + "\t\t\t\t\t" +i[1] )

itemsim.reverse()

f.write("\n\n----------------------------\n")
f.write("Bottom 5 Recommendations\n")
f.write("----------------------------\n")

f.write("Predicted Rating\tMovie")
for i in itemsim[0:5]:
    f.write("\n" + str(round(i[0],2)) + "\t\t\t\t\t" +i[1] )

```

Figure 7: The code for part3 (part3.py)

Problem 4: Correlations to my favorite and least favorite films

In order to find correlations to my favorite and least favorite films it was necessary to slightly modify the loadMovieLens() function. In all the previous parts the u.data was stored in a dictionary with the 'user id ' as the default key. The change that needed to be made was to make the 'movie id' the default key. This is equivalent to transforming the dictionary as is mentioned in the tectbook. Then the topMatches() function is used on the 2 films in order to generate the most correlated films. Before, when calling this function we were comparing 2 users and looking at which movies they have rated. Now we are comparing 2 movies and looking at which users have rated them.

The film chosen as my favorite film was 'Reservoir Dogs' with a movie id of '156'. My least favorite was 'Phat Beach' with a movie id of '1162'.

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

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

Figure 8: Modified loadMovieLens() function

```
3  import recommendations as re
4  data,mov = re.loadMovieLens()
5
6  f = open("MY Correlated movies",'w')
7
8  sim = re.topMatches(data,'156')
9  top = sim[0:5]
10 bottom = sim[len(sim)-5:len(sim)]
11
12 f.write("My favorite movie : Reservoir Dogs (1992)\n")
13 f.write("-----------------------------\n")
14 f.write("Top 5 Correlations\n")
15 f.write("-----------------------------\n")
16
17 f.write("Corr\tMovie")
```

Figure 9: Code for part4 (part4.py)

The results for the 5 most correlated films to Reservoir Dogs were not films that I really knew, but the results were not accurate. I have not seen any of them and after researching them I was only really convinced that 3 were worth watchng. 'Murder, My Sweet (1944)' and 'Thieves (Voleurs, Les) (1996)' seemed like they would be interesting films. But I really want to see 'No Escape (1994)' because the thought of Ray Liotta as an action hero is just hilarious to me. As for the 5 worst correlations I have no objections since all of them are 90's rom coms. It is interesting to note that almost all of the suggestions are in the same decade as 'Reservoir Dogs' meaning that the users(most influencial) were probably all of a certain generation that had watched all these films throughout the 90s.

```
My favorite movie : Reservoir Dogs (1992)
-----------------------------
Top 5 Correlations
-----------------------------
Corr  Movie
1.0   For Love or Money (1993)
1.0   Picture Bride (1995)
1.0   No Escape (1994)
1.0   Murder, My Sweet (1944)
1.0   Thieves (Voleurs, Les) (1996)

-----------------------------
Bottom 5 Correlations
-----------------------------
Corr  Movie
-1.0  Feast of July (1995)
-1.0  Pyromaniac's Love Story, A (1995)
-1.0  That Darn Cat! (1997)
-1.0  Two Much (1996)
-1.0  Whole Wide World, The (1996)
```

Figure 10: Top 5 and Bottom 5 correlations for 'Reservoir Dogs'

The results for the 5 most correlated films to 'Phat Beach' seemed correct in the sense that they were similar to the target film, with the exception of 'A time to kill' and '7 years in Tibet'. The 5 worst corellated films also seemed very accurate with the 'Usual Suspects' standing out as a pretty good movie , and was expecting as a positive correlation for 'Reservoir Dogs'("Gimme the keys you c*******er!! – I love that scene). These correlations follow a similar pattern to the one noticed previously, in that the majority of them are 90s films.

```
My least favorite movie : Phat Beach (1996)
----------------------------
Top 5 Correlations
----------------------------
Corr  Movie
1.0   Time to Kill, A (1996)
1.0   Seven Years in Tibet (1997)
1.0   Air Force One (1997)
1.0   Don't Be a Menace to South Central While Drinking Your Juice in the Hood (1996)
1.0   Stupids, The (1996)

----------------------------
Bottom 5 Correlations
----------------------------
Corr  Movie
-1.0  Love Bug, The (1969)
-1.0  Wizard of Oz, The (1939)
-1.0  Ghost in the Shell (Kokaku kidotai) (1995)
-1.0  Usual Suspects, The (1995)
-1.0  When We Were Kings (1996)
```

Figure 11: Top 5 and Bottom 5 correlations for 'Phat Beach'