CSE 231 Spring 2017

Programming Project 7

(edit 3/1: fixed a typo in *num_in_common_between_lists(user1_friend_lst, user2_friend_lst* as described in c) `calc_similarity_scores(network)`)

(edit 3/7: fixed a typo in project specification 2-f) *user_id* that user enters should be in the range [0,n-1] (i.e., from 0 to *n-1*, inclusive))

This assignment is worth 50 Points (5% of the course grade) and must be completed and turned in before 11:59 on Monday, March 20.

Assignment Overview

This assignment will give you more experience on the use of:

1. Lists
2. Functions

Have you ever wondered how social media sites suggest friends to you? Well, most of the sites do indeed use highly sophisticated algorithms, but for this project you will implement a naive method to recommend the most likely new friend to users of a social network.

Assignment Deliverable

The deliverable for this assignment is the following file:

proj07.py – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via the **handin system** before the project deadline.

Background

A social network such as Facebook consists of a set of users and connections between the users (i.e., there exists a connection between you and everyone who you have befriended). An interesting problem is to write a computer program that can automatically suggest possible new connections (i.e., friends) for each user. For this project you will implement a system that will, for each user, suggest the most probable user to befriend based upon the intersection of your common friends. In other words, the user that you will suggest to Person A is the person who has the most friends in common with Person A, but who currently is not friends with Person A.

The algorithm begins with: *for* each user go through all the other users (another "for") and calculate the number of friends they have in common. Then for a given user the friend you will suggest to them is the user in the social network who they are currently not friends with, but have the most friends in common. Intuitively, it makes sense why they might want to be connected as friends. (Note that when you look for your most common friend in this scheme it will be you, i.e. you will have to remember to remove yourself from consideration.)

Project Description / Specification

1) Two files have been provided for you to run your program with. One is a small test file containing a made-up set of users and connections between them (that file is `small_network_data.txt`). The other is a subset of a real Facebook dataset, which was obtained from:
https://snap.stanford.edu/data/egonets-Facebook.html

The format of both files is the same:
The first line of the file is an integer representing the number of users in the given network.
The following lines are of the form: user_u user_v
where user_u and user_v are the IDs of two users who are friends.
For example, here is a very small file that has 5 users in the social network:

```
5
0 1
1 2
1 4
2 3
```

The above is a representation of a social network that contains 5 users.
User ID=0 is friends with User IDs = 1
User ID=1 is friends with User IDs = 0, 2, 4
User ID=2 is friends with User IDs = 1, 3
User ID=3 is friends with User IDs = 2
User ID=4 is friends with User IDs = 1

2) We provide a file with function stubs and one complete function `init_matrix(n).` You must implement the following functions:

a) `open_file()` prompts the user to enter a file name. The program will try to open the data file. Appropriate error message should be shown if the data file cannot be opened. This function will loop until it receives proper input and successfully opens the file. It returns a file pointer.

b) `read_file(fp)` will have the file pointer `fp` passed into it. The first line of the file contains the value for the variable *n*, the number of users in the social network. We have provided the code to read *n* and initialize a list *network* containing *n* nested empty lists (e.g., *n* = 3 would correlate to creating the list [[], [], []]). This list of lists will be used to hold the list of friends for each of the *n* users. You can then use a `for` loop to iterate over the remaining lines of the file. For each line of the file you will want to obtain the two user id values on that line—we denote those user ids as *u* and *v*. You need to place *v* into *u*'s list and also place *u* into *v*'s list, since they are mutual friends (hint: use the list `append()` method to place each user id into a list). Finally you will return *network*, which is the list of lists that was created.

c) `calc_similarity_scores(network)` takes the social network as the parameter `network` and creates a "similarity matrix" for all the users in the social network. Here the definition of "similarity" is the number of friends that any pair of users have in common. The first thing you will want to do is initialize a list of lists that is *n* x *n* in size to all zeros that will hold the similarity (i.e., number of common friends) for each pairing of users. In other words, you will create a list of lists named *`similarity_matrix`* that will have *n* nested lists, each

containing *n* zeros. (e.g., *n*=3 gives the following list [[0,0,0], [0,0,0], [0,0,0]]). We provide the function `init_matrix(n)` for you that will create and initialize the matrix.
You will then use two nested `for` loops to iterate over all pairs of users in the *network*. For each pair of users you can obtain their list of friends from `network`, and then call the function *num_in_common_between_lists(user1_friend_lst, user2_friend_lst)*, which will return an integer, *num_in_common*, representing the number of friends those two users have in common. The int *num_in_common* will need to be stored in locations [user1][user2] and [user2][user1] of the list of lists *similarity_matrix*. Return *similarity_matrix*
Suggestion, to iterate over all pairs of users use the following:

```
for i in range(n):
        for j in range(n):
             # call num_in_common_between_lists using
                user i's list and user j's list as arguments
```

d) `num_in_common_between_lists(list1, list2)` is a function that calculates the number of common items between two lists. To do this you can have a `for` loop that iterates over the items of *list1* and then an `if` statement that checks whether or not that item is also in *list2*. This function is to return an integer, which is the number of items that the two lists had in common (i.e., the number of items that were found in both of the lists).
(e.g., if list1 = [1,3,5,6] and list2 = [1, 2, 3], then the value 2 would be returned since they have two items in common (the values 1 and 3)).

e) `recommend(user_id,network,similarity_matrix)` This *user_id* will be used as an index into the *similarity_matrix* which will give you access to the list of similarity scores for that given user (*user_id*) for all other users. You should then determine the largest value in this list and return its index as the most similar. Of course, you don't want to recommend someone who is already a friend and it also doesn't make sense to recommend the person as his or her own friend. It is the most similar because by having the largest value in the list it means that it is the user who had the most friends in common with *user_id*. (Hint: while developing this function using the small network file I printed out `similarity_matrix[user_id]` and `network[user_id]` so I could check that the correct recommendation was calculated.)

f) `main()` In the main program you will need to first call *open_file()*, followed by calling *read_file(fp),* and then *calc_similarity_scores(network).* After obtaining the *similarity_matrix* you will need to have a `while` loop that will repeatedly ask the user for a *user_id* that should be in the range of [0,*n-1*] (i.e., from 0 to *n-1*, inclusive). Call *recommend* to get a recommendation.

g) Note that when prompting the user for the *user_id* you should validate that the input is in the correct range, which is 0 to *n*. Also, you must use a `try-except` for the cases when the user inputs a non-integer value (e.g., for the case if they try to input the string "hello" instead of an int).

h) After displaying the suggested friend for the user that had their id as *user_id* you should prompt the user if they want to see the suggestion for another user input or if they want to exit. To exit the program you should accept all forms of the string 'no' (i.e., 'NO', 'No', 'nO', and 'no').

Sample Outputs

Function Test 1
(Tests the `read_file` function; reads the file `small_network_data.txt`)

```
network:
[[1, 2, 3], [0, 4, 6, 7, 9], [0, 3, 6, 8, 9], [0, 2, 8, 9], [1, 6, 7,
8], [9], [1, 2, 4, 8], [1, 4, 8], [2, 3, 4, 6, 7], [1, 2, 3, 5]]
--------------------
network printed nicely:
0 : [1, 2, 3]
1 : [0, 4, 6, 7, 9]
2 : [0, 3, 6, 8, 9]
3 : [0, 2, 8, 9]
4 : [1, 6, 7, 8]
5 : [9]
6 : [1, 2, 4, 8]
7 : [1, 4, 8]
8 : [2, 3, 4, 6, 7]
9 : [1, 2, 3, 5]
```

Function Test 2
(Tests the `num_in_common_between_lists` function; uses `network[1]` and `network[2]` from Test 1 for first test)

```
should be 3: 3
should be 1: 1
```

Function Test 3
(Tests the `calc_similarity_scores` function; uses `network` from Test1 as the argument)

```
similarity matrix
[[3, 0, 1, 1, 1, 0, 2, 1, 2, 3], [0, 5, 3, 2, 2, 1, 1, 1, 3, 0], [1,
3, 5, 3, 2, 1, 1, 1, 2, 1], [1, 2, 3, 4, 1, 1, 2, 1, 1, 1], [1, 2, 2,
1, 4, 0, 2, 2, 2, 1], [0, 1, 1, 1, 0, 1, 0, 0, 0, 0], [2, 1, 1, 2, 2,
0, 4, 3, 2, 2], [1, 1, 1, 1, 2, 0, 3, 3, 1, 1], [2, 3, 2, 1, 2, 0, 2,
1, 5, 2], [3, 0, 1, 1, 1, 0, 2, 1, 2, 4]]
--------------------
similarity matrix printed nicely:
0 : [3, 0, 1, 1, 1, 0, 2, 1, 2, 3]
1 : [0, 5, 3, 2, 2, 1, 1, 1, 3, 0]
2 : [1, 3, 5, 3, 2, 1, 1, 1, 2, 1]
3 : [1, 2, 3, 4, 1, 1, 2, 1, 1, 1]
4 : [1, 2, 2, 1, 4, 0, 2, 2, 2, 1]
5 : [0, 1, 1, 1, 0, 1, 0, 0, 0, 0]
6 : [2, 1, 1, 2, 2, 0, 4, 3, 2, 2]
7 : [1, 1, 1, 1, 2, 0, 3, 3, 1, 1]
8 : [2, 3, 2, 1, 2, 0, 2, 1, 5, 2]
9 : [3, 0, 1, 1, 1, 0, 2, 1, 2, 4]
```

Function Test 4
(Tests the `recommend` function; uses `similarity_matrix` from Test3)

```
person,friend: 3 1
person,friend: 0 9
```

Test 1
(note that an alternative correct answer for 3 is 6 – there was a tie)

```
Facebook friend recommendation.

Enter a filename: small_network_data.txt

Enter an integer in the range 0 to 9 : 0
The suggested friend for 0 is 9

Do you want to continue (yes/no)? yes
Enter an integer in the range 0 to 9 : 3
The suggested friend for 3 is 1

Do you want to continue (yes/no)? yes
Enter an integer in the range 0 to 9 : 8
The suggested friend for 8 is 1

Do you want to continue (yes/no)? No
```

Test 2

```
Facebook friend recommendation.

Enter a filename: xxxxxx
Error in filename.
Enter a filename: small_network_data.txt

Enter an integer in the range 0 to 9 : abc
Error: input must be an int between 0 and 9
Enter an integer in the range 0 to 9 : -1
Error: input must be an int between 0 and 9
Enter an integer in the range 0 to 9 : 15
Error: input must be an int between 0 and 9
Enter an integer in the range 0 to 9 : 10
Error: input must be an int between 0 and 9
Enter an integer in the range 0 to 9 : 0
The suggested friend for 0 is 9

Do you want to continue (yes/no)? nO
```

Test 3
Note that this test takes time because of the size of the social network.

```
Facebook friend recommendation.

Enter a filename: facebook_1000_data.txt

Enter an integer in the range 0 to 999 : 88
The suggested friend for 88 is 213

Do you want to continue (yes/no)? yes
Enter an integer in the range 0 to 999 : 20
The suggested friend for 20 is 116

Do you want to continue (yes/no)? yes
Enter an integer in the range 0 to 999 : 5
The suggested friend for 5 is 67

Do you want to continue (yes/no)? yes
Enter an integer in the range 0 to 999 : 500
The suggested friend for 500 is 559

Do you want to continue (yes/no)? no
```

Scoring Rubric

```
Computer Project #07                           Scoring Summary

General Requirements

______    6 pts    Coding Standard 1-9
        (descriptive comments, function header, etc...)


Implementation:

__0__    (8 pts)  Pass test1:

__0__    (8 pts)  Pass test2

__0__    (8 pts)  Pass test3

__0__    (16 pts)  All functions work as specified.
                   (4 pt) read_file
                   (4 pt) num_in_common_between_lists
                   (4 pt) calc_similarity_scores
                   (4 pt) recommend
__0__    (4 pts)  Main, correct loop control, error checking


TA Comments:
```

Optional Testing
This test suite has a test program for each of the three functions and two for the entire program. They are handled slightly differently.

1. Test individual functions using `function_testX.py` and the entire program using `run_file.py.`
   You need to have the files `test1.txt`, `test2.txt,` and `test3.txt` from the project directory.
   Make sure that you have the following lines at the top of your program (only for testing):

```
import sys
def input( prompt=None ):
    if prompt != None:
        print( prompt, end="" )
    aaa_str = sys.stdin.readline()
    aaa_str = aaa_str.rstrip( "\n" )
    print( aaa_str )
    return aaa_str
```

Educational Research

When you have completed the project insert the 5-line comment specified below.

For each of the following statements, please respond with how much they apply to your experience completing the programming project, on the following scale:

1 = Strongly disagree / Not true of me at all
2
3
4 = Neither agree nor disagree / Somewhat true of me
5
6
7 = Strongly agree / Extremely true of me

****Please note that your responses to these questions will not affect your project grade, so please answer as honestly as possible.****

Q1: Upon completing the project, I felt proud/accomplished

Q2: While working on the project, I often felt frustrated/annoyed

Q3: While working on the project, I felt inadequate/stupid

Q4: Considering the difficulty of this course, the teacher, and my skills, I think I will do well in this course.

Q5: I ran the optional test cases (choose 7=Yes, 1=No)

Please insert your answers into the bottom of your project program as a comment, formatted exactly as follows (so we can write a program to extract them).

```
# Questions
# Q1: 5
# Q2: 3
# Q3: 4
# Q4: 6
# Q5: 7
```