15 Nov 2019
GE-103

Suggesting Friends

Sagalpreet Singh
(2019CSB1113)

Problem Statement

- *The given text file is the original data that has been obtained from SNAP (Stanford Network Analysis Project). The data is anonymized (names replaced by numbers) and represents undirected social circles. It is provided in the form of lines with each line containing two numbers (numbers here actually represent people) who are friends on Facebook.*

- *Unfortunately, the data obtained from SNAP contains only friend pairs and no other information. To tackle this, a code has to be written to randomly allocate a school, a college and a workplace from amongst fifty schools, colleges and workplaces (all generalized variables) to all the individual files. Also, gender (male or female) has to be randomly allocated to the individual users.*

- *The aim of the project is to take an input in the form of user name and print at most Ten friend suggestions with whom the user is not connected but maybe interested in doing so.*

- *The program has to be written in Python.*

MOTIVATION

The very fact that any two persons are connected through common links on an average of 6 people inspired us to take up this project.

In order to grow the social circle, we created a program to suggest friends and reduce the average distance (if not make it 0) between persons who actually know each other but aren't connected.

With each addition of a new friend on the basis of given friend suggestions, the further suggestions are bettered.

Not only would this help people who know each other establish links but also bring people from common backgrounds and with common interests closer to each other.

The example for this is clearly visible in the friend suggestions provided by Facebook.

Project Components

- Project Report
- Original Data File (facebook_combined.txt)
- User files (‘ ’.txt) (created from original file for all users through file handling)
- Python file to create user files (creating_users.py)
- Python file to add random data about users (add_random_data.py)
- The main python program to suggest friends for (Score&friendUpdate.py)

The project was coordinated between the members using GitHub as the files for the project were exclusively made without intervention of each other.

CREATING USERS.PY

THIS PYTHON PROGRAM WAS WRITTEN TO ACCESS DATA FROM FACEBOOK_COMBINED.PY AND CREATE SEPARATE USER FILES FOR EACH OF THE USER PRESENT IN THE SOCIAL CIRCLE.

THERE WERE A TOTAL OF 4039 USER FILES CREATED.

ADD_RANDOM_DATA.PY

THIS PYTHON PROGRAM WAS WRITTEN TO ADD RANDOM DATA TO USER FILES THAT HAVE BEEN CREATED SEPARATELY FOR EACH USER. THIS WAS DONE IN ORDER TO INCREASE THE NUMBER OF CRITERIA FOR PROVIDING SUGGESTIONS. THE ADDED RANDOM DATA CONSISTS OF SCHOOL, COLLEGE ATTENDED, GENDER AND WORKPLACE.

AFTER BRAINSTORMING, WE DECIDED THE RELATIVE WEIGHTAGE FOR EACH OF THESE CRITERIA AS FOLLOWS:

- SCHOOL – 1
- COLLEGE – 1
- WORKPLACE - 2
- GENDER - 1
- MUTUAL FRIENDS – 3 EACH

THE GENDER CRITERION: THIS WAS DECIDED KEEPING IN MIND THE CURRENT TREND AND HUMAN PSYCHOLOGY OF BEING EAGER TO CONNECT TO THE PEOPLE OF OPPOSITE GENDER RATHER THAN THOSE OF THE SAME GENDER.

ALL OTHER CRITERIA: FOR EACH MATCH IN THE REST OF THE CRITERIA, RESPECTIVE SCORES WERE INCREMENTED.

SCORE&FRIENDUPDATE.PY

THIS IS THE MAIN PYTHON FILE WHICH CONSISTS OF THE PROGRAM THAT TAKES USER NAME AS INPUT AND OUTPUTS THE TOP 10 FRIEND SUGGESTIONS ON THE BASIS OF THE SCORE AWARDED AS PER THE ABOVE WRITTEN CRITERIA.

BELOW WRITTEN IS THE ALGORITHM FOR THE SAME:

1. The first step was to create a function that would return the score of matching of two parametrized user inputs. This was accomplished by creating a **comp(a,b)** function where a and b refer to users to be compared.
2. Since the function had to be called repeatedly to check for collisions with friends of friends who weren't linked with the user, a dictionary was created by the name **friend** to store the respective scores of comparisons.
3. The body of the program was written so as to take the input in form of user name(whose friend suggestions are to be displayed), say 'a'.
4. A nested loop was run to access the friends of friends of 'a' and filter them out with the condition that the person should not already be friends with 'a' and should not be 'a' itself.
   They were then compared with 'a' using **comp** function.
5. Yet another challenge was to sort dictionary on the basis of value for which we don't have any predefined function. As this function was to be used in the code(which was already running in $\mathbf{n^2}$ time complexity), it was important to get the best possible way out. The sort function for this was then written using algorithm similar to quick sort, thereby getting time complexity of **n(logn)** for sorting.
6. The top 10 suggestions were then printed.

Things we learnt from this project

***GitHub**: We learnt to use GitHub. Also, we learnt to do tasks in a coordinated manner, read others' codes, find errors, correct them, make appropriate use of comments and make additions to them.*

***Sorting Dictionary**: We learnt to sort dictionary on the basis of values without using any in-built function in n(logn) time complexity.*

***File Handling:** We learnt the importance of file handling and how retrieval of data can be made easier and more accessible.*

***Time Complexity:** Handling 4039 files was a tough challenge and we failed a lot of times while working with them, due to bad time complexity and inefficient memory management .We tried our best to save time in comparing, sorting and analyzing.*

***Importance of pen-paper:** We practically realized, that we should spend more time reading the code than writing it as the distinction became clearer in this project which was longer than usual codes that we write.*

“Computer Science is no more about computers than astronomy is about telescopes.”
— Edsger W. Dijkstra