OLD DOMINION UNIVERSITY

CS 495: Introduction to Web Science
Instructor: Micheal L. Nelson, Ph.D
Fall 2014 Thursdays 4:20pm – 7:10pm ECSB 2120

Assignment # 5
Joseph Elder UIN: 00844802

Honor Pledge
I pledge to support the Honor System of Old Dominion University. I will refrain from any form of academic dishonesty or deception, such as cheating or plagiarism. I am aware that as a member of the academic community it is my responsibility to turn in all suspected violations of the Honor Code. I will report to a hearing if summoned

October 9, 2014

Contents

1 Assignment 5

1.1 Question 1

1.1.1 The Problem

The "friendship paradox" (http://en.wikipedia.org/wiki/Friendship_paradox) says that your friends have more friends than you do.

1. Explore the friendship paradox for your Twitter account. Since Twitter has directional links (i.e., "followers" and "following"), we'll be investigating if the people you follow (Twitter calls these people "friends") follow more people than you. If you are following < 50 people, use my Twitter account "phonedude_mln" instead of your own.

Create a graph of the number of friends (y-axis) and the friends sorted by number of friends (x-axis). (The friends don't need to be labeled on the x-axis as "Bob", "Mary", etc. -- just 1, 2, 3 ...) In other words, if you have 100 friends your x-axis will be 1..101 (100 + you), and the y-axis value will be number of friends that each of those friends has. The friend with the lowest number of friends will be first and the friend with the highest number of friends will be last.

Do include yourself in the graph and label yourself accordingly. Compute the mean, standard deviation, and median of the number of friends that your friends have.

The appropriate part of the Twitter API to use is:

https://dev.twitter.com/rest/reference/get/friends/list

1.1.2 The Solution

For assignment five I chose to use my Twitter account @up5free to explore the friendship paradox concept. Python version 2.7.6 and the Python module Tweepy were used to generate usable data from Twitter for graphing. Tweepy's homepage defines the module as “An easy-to-use Python library for accessing the Twitter API.” The API object from the Tweepy module allows quick access in Python to the all Twitter RESTful API methods.

The file “A5.py” ultimately will print the friendship paradox results in regards to my Twitter friends to the text file “paradox.txt”. The program first uses Tweepy to get a complete list of all my friends from Twitter. Since I am new to Twitter I chose exactly 50 Twitter users to follow somewhat randomly based off my interests. Since Twitter users can potentially have thousands of friends attempting to simply print all friends is by default not possible, the API limits you to a set number of users. While you can adjust that set number, the simplest way to return all users no matter how many you are following is to use a Tweepy cursor.

Because of the potential to have a huge number of friends or followers on Twitter, when querying for all friends results are returned in pages. The Tweepy cursor is used to navigate through all of the pages, allowing for processing of each page to return every friend, or user who I am currently following. After a friend is processed from the page the next procedure is to retrieve the friend count for that user. The method to get friend count is similar to the method to get all of my friends, instead of my name being used the usernames of my Twitter friends are used to gather all friends. Those friendships are then counted using the python “len()” method to return friend count. The results are printed to paradox.txt. Which are used to generate the graph using libre office calc, software similar to microsoft excel.

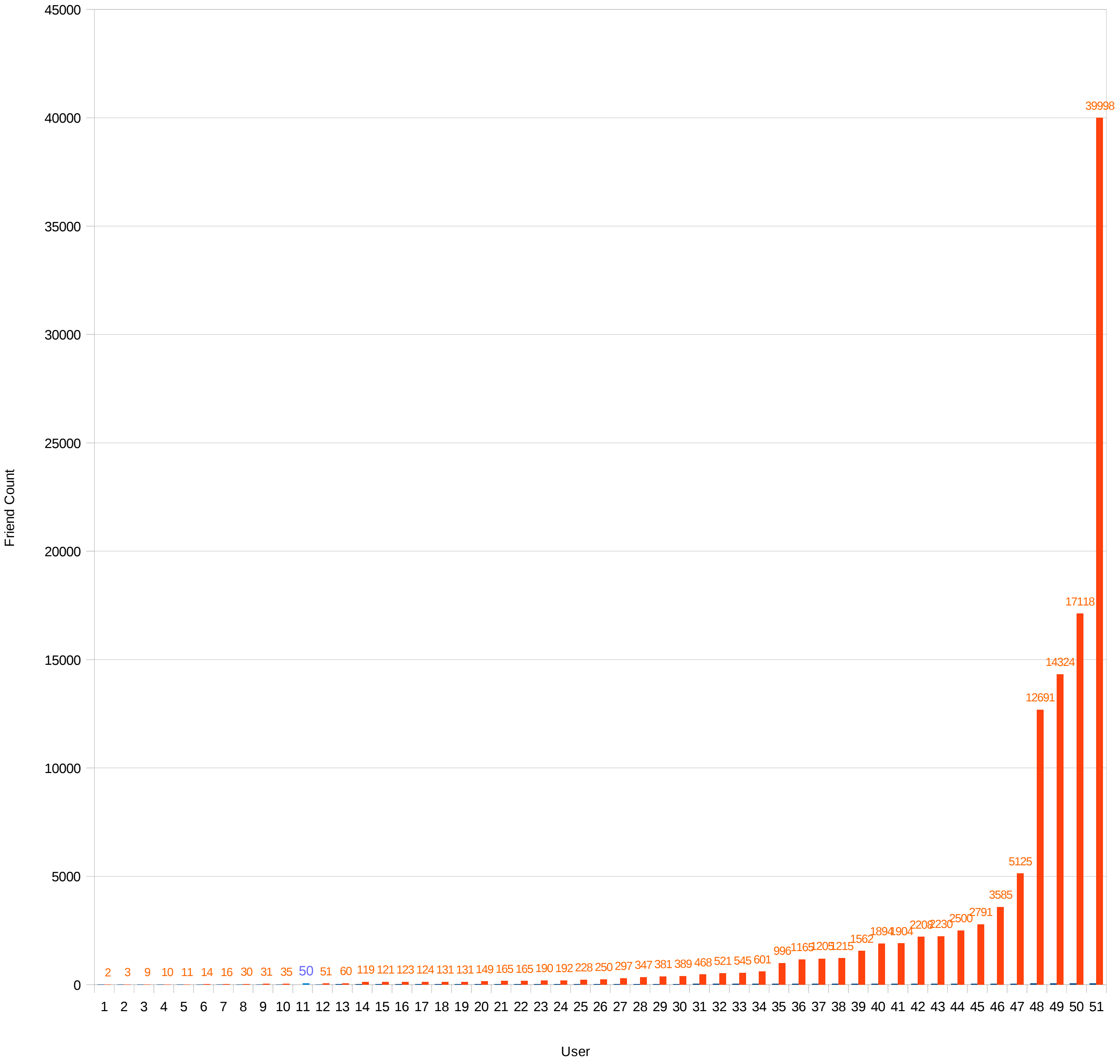
The Twitter API is rate limited, meaning you can only execute a specified number of GET requests for every 15 minutes. To deal with the rate limit the time module must be imported in Python to allow the program to sleep for a given period of time. This sleep is necessary to ensure that the rate limit is not exceeded throwing an exception.

Another python program, which I had coincidentally already created for a statistics class, “stdDeviation.py” is used to generate standard deviation as well as the mean for the friend count dataset. This program accepts each friend count as input including myself, then calculates the mean and the standard deviation automatically. The mean for this dataset is 2322.94 which would likely be rounded to 2323 friends since we cannot have fractional friendships. The standard deviation for this dataset is 16232.047. The median for this set of data set including myself is 250 friends.

My friend count is rank the 11th lowest on the graph so I am represented by 11. Additionally the graph data and data-labels show all of my Twitter friends represented in orange and myself represented in blue. The Graph clearly shows an out lier value for the 51st entry which in turn should have some effect on the resulting data.

(This space intentionally left blank)

Twitter Friendship Paradox



1.2 Question 2

1.2.1 The Problem

2. Using your Facebook account, repeat question #1 (if you have >
50 friends).

Start at: https://developers.facebook.com/docs/graph-api/reference/v2.1/user/friends

or perhaps:

http://socialnetimporter.codeplex.com/

1.2.2 The Solution

The Python module “facebook-sdk” is imported by “import facebook” to allow access to Facebook's API. Compared to the Twitter API the Facebook API has become much more restrictive for accessing data about users even the users you are friends with. In version 2.0 of the Facebook API “2.0/me/friends” returns only the Facebook users who have allowed access to the web application being developed. Since the application being used for this class currently has no users besides myself the query will unfortunately return empty. An example of this being executed using Facebook's Graphs API can be found in my github repository at “A5/facebook/fbgraphs.py” however it only returns an empty set.

Facebook allows access to information by another method known as Facebook Query Langue. Facebook Query Language or FQL queries can be executed in Python via web reuqest. The python module “urllib2” is used in the python file “A5/facebook/fbfql.py” to execute Facebook queries. I found the following useful FQL query example on StackOverflow “SELECT friend_count FROM user WHERE uid = UserID” will return the friend count of a user given a valid user ID. The file “fbfql.py” utilizes this directly, supplied my Facebook uid 100000108537035, the query successfully returns my friend count of 166. Though the FQL solution would work Facebook uids are not easy to come by using either method to access Facebook's API.

I found my uid by manually scraping my homepage photo which contains a link, however this would work for any user. The uid can be found by searching the text of this link for a number separated by two dots or periods. The number after the second period will be the uid for whatever homepage you are collecting it from.

In theory from here it would not be difficult to then create python code to login to my Facebook account navigate to my friends list then loop through all of my friend's homepages in search of their uids in similar fashion to how it would be done on a browser. An automated process which would most likely utilize BeautifulSoup could be implemented to parse the HTML from each homepage. After parsing each homepage each uid would be stored in a list for easy access. The FQL query in “fbfql.py” would the be executed for each uid, looping through all users returning their friend count. The results from each query would then be processed returning only the friend count for each user which could easily be written to an output file. This output would be correct however from what information I could gather on the Internet Facebook does not allow for scraping of their pages due to privacy violations of users.

References

1. https://docs.python.org/2/
2. https://dev.twitter.com/rest/reference/get/friends/list
3. https://developers.facebook.com/docs/graph-api/reference/v2.1/user/friends
4. http://www.tweepy.org/
5. http://tweepy.readthedocs.org/en/v2.3.0/
6. http://stackoverflow.com/questions/3527518/how-can-i-count-the-total-friends-of-a-facebook-user-by-uid
7. http://stackoverflow.com/questions/23417356/facebook-graph-api-v2-0-me-friends-returns-empty-or-only-friends-who-also-use-m