Junior Independent Study: Graph Centrality Algorithms

Scott Stoudt
CSCI 200 Algorithm Analysis

April 29, 2018

Contents

List of Figures

List of Equations

1 Overview

Within the areas of graph and network theory, identifying measures of centrality can highlight important connections between various points on a graph. Centrality is used to highlight important individuals within a social environment. This can be extended to explore the social dynamics in many situations, such as plays, scripts, public forums, case trials, and much more. The focus of this project is to research three types of centrality: degree centrality, closeness centrality, and betweenness centrality. Each offers a different lens that can be applied to the text analysis process in order to answer humanistic questions about the text. Each type of centrality offers unique algorithms for calculating the various measures of centrality. The intention of studying and understanding these algorithms is so that we can see how they apply to graph centrality as a whole and specifically to network analysis.

After introducing and explaining the previously mentioned types of centrality, their applications, and the process of network analysis, Section 4 discusses the process of creating a Python program that, given enough information to create a network, creates visualizations of the network and arranges the nodes appropriately based on the user's desire to see the network in terms of degree, closeness, or betweenness.

2 Graph Centrality

Within the areas of graph theory and network analysis, we use centrality to highlight the most influential subjects within a particular graph. The elements within a particular data-set that we focus on are referred to as the vertices. When it comes to figuring out the importance of a vertex, we should probably define what "importance" means in our current context. Depending on the type of centrality, the importance of a vertex can change. In some cases it can be defined as the level of involvement within the network as a whole. Another example of importance is how well a vertex contributes to a predetermined flow throughout a network. These flows can come in many forms such as, money exchange, a particular method of communication, or following a specific idea as it is distributed across the network.

Centrality can be measured in many different ways, but for the purpose of this paper, we focus on centrality in relation to degree, betweenness, and closeness. Each method of calculating centrality has its own benefits and specific ways of displaying the data, which we examine in greater depth.

2.1 Degree Centrality

Degree centrality is the most basic of the centrality measurements and is the simplest to understand. The degree of a specific vertex within a graph is the number of edges it has, which includes self-connecting loops [6]. The degree of vertex x is denoted $\deg(x)$. Measurements for degree centrality can be further broken down into measurements of indegree and outdegree. Indegree is a value for the total number of edges, or links, that come in to a specific node. Outdegree is the number of links a particular node "sends" to other nodes. These measurements are used when dealing with directed networks, where the direction of the links between nodes are important. Directed networks are useful in figuring out which individuals in a network are contributing and distributing data the most and which are just receivers.

2.1.1 The Algorithm

What you need to know beforehand:

▶ A graph G with $|V|$ vertices and $|E|$ edges is written as $G := (V, E)$. The degree centrality for a vertex v is written $C_D(v) = \deg(v)$, which is calculated in $\Theta(V^2)$ time.

▶ $v*$ is the node with the highest degree centrality in graph G

Suppose we have some graph G, its degree centrality is:

$$C_D(G) = \frac{\sum_{i=1}^{|V|}[C_D(v*) - C_D(v_i)]}{|V|^2 - 3|V| + 2} \tag{1}$$

2.1.2 An Easier Explanation

For an undirected graph, degree centrality is found by analyzing the connections between all of nodes and figuring out how many connections each node has. Nodes with more connections have a higher degree and are more central to the graph. These aspects are made more apparent in the following discussion.

Example:
Suppose we have a graph G with the following defined edges and $V = \{0, ..., 6\}$,

$$\begin{aligned} G = \{&(0 \longleftrightarrow 1), \\ &(1 \longleftrightarrow 2), \\ &(0 \longleftrightarrow 3), \\ &(0 \longleftrightarrow 4), \\ &(0 \longleftrightarrow 5), \\ &(0 \longleftrightarrow 6)\} \end{aligned}$$

Visualizing the nodes of G and their edges results in the following figure:

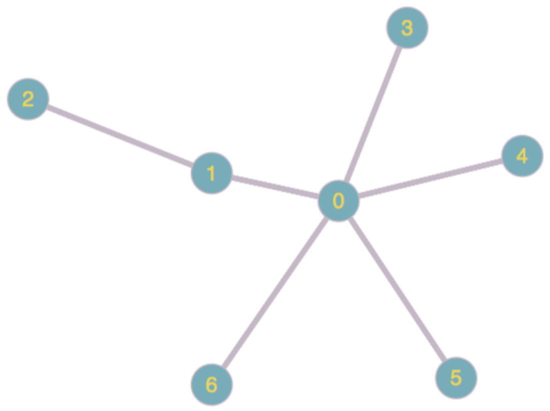

Figure 1: Graph G shows the visual representation of degree centrality for its nodes.

As we can see, the 0 node has the most links with other nodes, so it is placed at the center of the visualization. Further more, node 1 has two links (one of the links being with the 0 node) and it is the node with the second highest degree; therefore it too is near the middle of the visualization.

2.2 Betweenness Centrality

Betweenness centrality is a measurement based on the shortest paths within a graph from node to node [3]. Each node within a graph can be connected to every other node via at least one shortest path. The betweenness centrality of a particular node in the graph is calculated based on the number of these shortest paths that pass through the vertex in question. When calculating betweenness centrality, you must first consider which type of graph you are using: a weighted or unweighted graph. For unweighted graphs, the shortest path means the minimum number of hops between the two nodes. However, for weighted graphs, the shortest path means the paths whose total weight of the edges is the smallest. Measuring the betweenness of nodes within a graph tells us which nodes are the most influential (with nodes of the highest betweenness being the most influential). Areas that benefit from betweenness centrality calculations include biology and the study of ecosystems, social networking, and computer networking for determining which servers, cell towers, etc. hold the most control over a particular geographical area.

2.2.1 The Algorithms

Unweighted:

$$g(v) = \Sigma_{s,t \in N} \frac{\sigma_{st}(v)}{\sigma_{st}}, \text{ where } s \neq v \neq t \tag{2}$$

The algorithm calculates the betweenness centrality of some node v (written $g(v)$). Here, σ_{st} represents the total number of shortest paths that exist between the given nodes s and t. $\sigma_{st}(v)$ is the number of previously mentioned paths that run through node v.

Weighted:

The algorithm for finding betweenness in a weighted network works a bit differently, since we are switching our focus from fewest amount of hops between nodes to the path with smallest total edge weight. The algorithm is as follows:

$$s_i = \Sigma_{j=1}^{N} a_{ij} w_{ij} \tag{3}$$

Where a_{ij} and w_{ij} are adjacency and weight matrices between the nodes i and j. The adjacency matrix A, which is NxN (N again being the number of nodes in the graph), is created by recording if an edge exists between two nodes. For all entries, if an edge exists between nodes i and j, then a_{ij} and a_{ji} are 1. The matrix entry is 0 if an edge does not exist between the two nodes. The weight matrix W is created by recording the weights of the edges between two adjacent nodes. For example, if we had a pair of adjacent nodes 1 and 2 with an edge value of 4, the matrix entry for $a_{1,2}$ and $a_{2,1}$ would be 4. For the entry in the adjacency matrix of two nodes that are not adjacent to each other, it would be '∞'.

For our purposes here, we look at unweighted graphs. The software created for calculating and visualizing centrality examines connections between various nodes (individuals) and how they interact with one another. Since it would not make a ton of sense to place arbitrary weights between the various interactions, the program will use the above algorithm for unweighted networks.

2.2.2 An Example

Consider the following network:

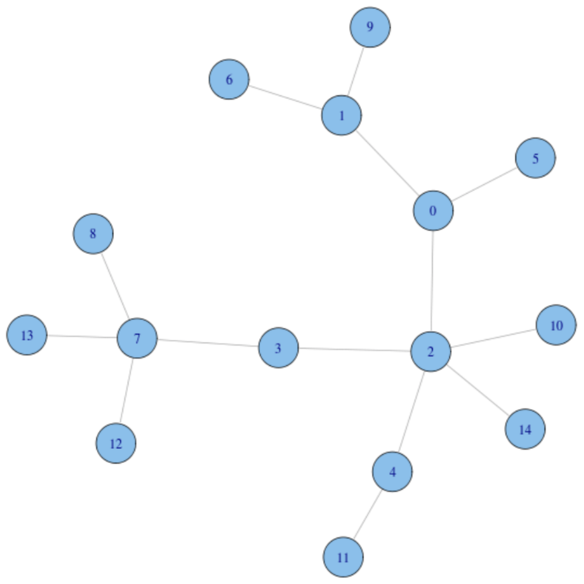

Figure 2: Image from [1]

After calculating the betweenness centrality of each node, we can update the network, replacing the node ID with each node's respective betweenness:

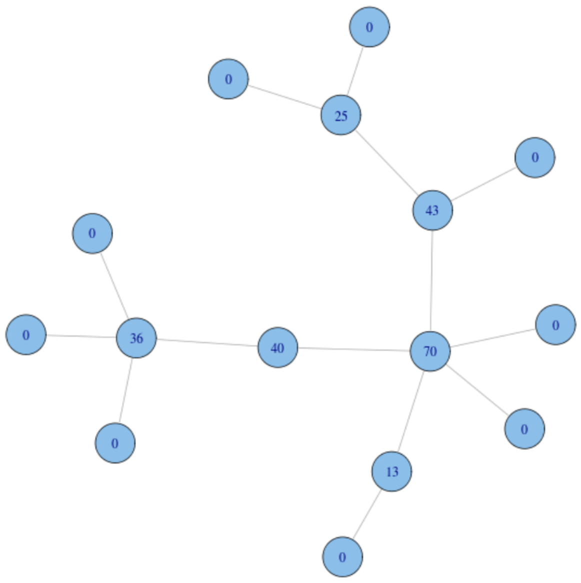

Figure 3: Image from [1]

The edge nodes, or the nodes that only have one connection, have betweenness centrality equal to zero, which is obvious because they cannot be passed through when navigating from two different points. In this relatively simple example, it is no surprise that node 2 has the greatest betweenness centrality with 70 as all ends of the network have to pass through it to get to others. If, for example, a connection were to be added between nodes 6 and 8, the centrality of the nodes (particularly node 2) would change noticeably as the group of nodes in the top and left of the network have a new, shorter way of reaching each other that does not involve passing through node 2.

2.3 Closeness Centrality

The closeness of a node is measured by adding up the lengths of the shortest paths from the node in question to all the other nodes in the network and dividing that number by the number of nodes in the network [5]. The node with the largest closeness centrality is closest to all the other nodes. Closeness centrality calculations are useful in determining which individuals in a network are positioned best to quickly communicate with the network as a whole. Some fields that benefit from using closeness centrality measurements are computer networking, economics, and political science. Since the world is just one big network, any other field that deals with the movement of people, things, and ideas across the globe would benefit from studying closeness centrality.

2.3.1 The Algorithm

There are many different versions of the algorithm for measuring closeness centrality, but for the purpose of this paper, we use the modified version of Bavelas' definition [4]:

$$C(x) = \frac{N}{\Sigma_y d(y,x)} \tag{4}$$

This formula shows that the closeness centrality of some x in a graph is calculated by taking N, the number of nodes in the graph, and dividing it by the sum of weights of the shortest path between x and each of the other nodes in the graph. Here $d(y,x)$ is the shortest distance between the node in question, x, and another node in the graph, y.

2.3.2 An Example

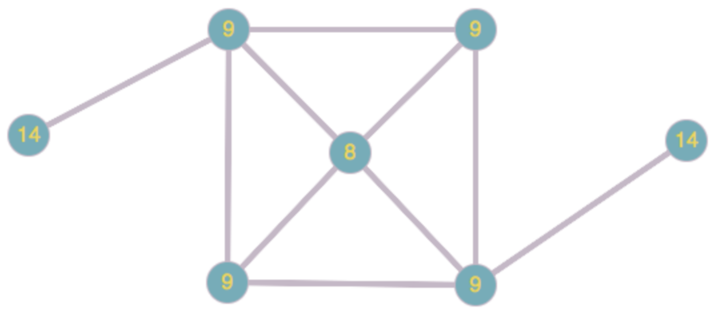

Figure 4: An example network, where the value inside the nodes indicate their sum of shortest path weights.

Above is a simple network consisting of 7 nodes. Each node is labeled with their respective sum of shortest path weights, or the bottom portion of the fraction in the formula presented in Equation 4. For example, the center node is 8 because if you add up the number of hops it takes for that node to reach each of the other nodes, you would get 8; one for each of the surrounding nodes and two for each of the end nodes labeled "14". To find the exact closeness calculation for node 8, we compute $\frac{N}{8}$. Since this graph has 7 nodes, our closeness centrality would be $\frac{7}{8} = 0.875$. If we were to compute the exact closeness centrality for all the other nodes, we find that node 8, whose value is .875, is the greatest. Thus, the node labeled with an 8 has the greatest closeness, meaning it can interact with all the other nodes in the graph in the shortest number of hops.

3 Application of Graph Centrality

3.1 Social Network Analysis

Social network analysis is the study of social structures using networks and graph theory. Isolating the patterns in social interactions, we can collect information to help answer some important social questions like: "Who knows what", "Who knows who", and "What ideas are circulating in a network and are they popular" [11]. Network analysis is helpful in answering the hard questions that are not apparent on the surface. For example, everyone knows that the boss of a company holds a substantial amount of information and has contact with everyone in the company, either directly or indirectly through the chain of command. However, there are many influential people in a workplace environment who are not in traditional positions of power. Through network analysis, we might see individuals who have influence among their co-workers be it from, social dominance, seniority, or other sociopolitical reasons.

To do this, a network is created using the individuals in the target environment as nodes and human interactions as the links between the nodes. In the network creation process, there is room for modification. Links can be limited or separated into different forms of communications such as written, verbal, and electronic. Links can also be broken up to highlight the content of the conversation to show when conversations are positive, negative, meaningful, small talk, etc. These judgments are made by the individual who is curating the data set based on the information they hope to extract from the visualizations. Properties of the links can be modified as well, specifying the direction the information is passed or adding weights to links that are assigned to give certain links more or less importance in the graph.

In literature and film, social network analysis is used to extract "similarities between narrative and social structure"[12]. Furthermore, comparing network analyses of many different works from the same author or director can highlight uniformity in how the writer creates social structure and dialogue within their pieces. A further study could be done in analyzing the differences in artificially created networks and naturally occurring social networks around the globe.

3.1.1 Brief Introduction to Computational Sociology

Computational sociology is a sub-field of sociology, which studies the culture, human interaction, and interactions between humans. Blending computer science with natural science, computational sociology uses computer simulations, text mining, and social network analysis tools, like the one developed for this project, to model, analyze, and form/test hypotheses on complex social processes [7].

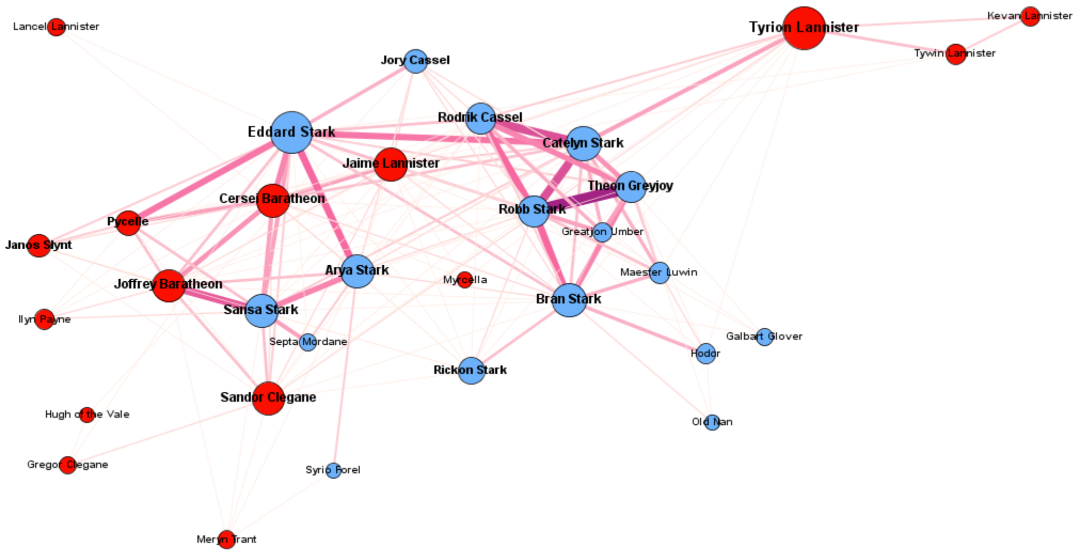

Figure 5: A degree centrality graph from the gameofnodes blog post [2]

The above network is comprised of individuals from two conflicting houses from the popular HBO series, *Game of Thrones*. The show is set in a fake medieval world that is full of its own very real and fleshed out culture, economy, and social interactions. This particular example is perfect for showing how network analysis can create visual networks to show how individuals interact with each other. From this graph, we can tell who the important individuals are within the respective houses based on who interacts with the most people in their group. Furthermore, we can see the key individuals who create the important ties between the two factions. These individuals more than likely are the ones who hold the most information about the overall situation of the realm at a given time. By visualizing all the interactions into one visualization, these minor, but obviously key characters are highlighted; showing off one of the benefits of computational sociology and network visualization.

4 Network Analysis Application

4.1 Introduction

This network analysis application uses Python, as it is a great language for data visualization. Python has a large amount of libraries one can use and thus for this application, the NetworkX data visualization package [9] is employed. NetworkX is great for network structuring, analysis, and visualization.

Rather than having the user compile the program with the desired file and producing one visual for the network at a time, a simple, functional graphical user interface is created. Using Tkinter, the application allows the user to upload an appropriate file (file specifics layed out for the user within the app) and view different visualizations of their data based on the different centrality calculations [14]. In viewing the different visualizations, the user can see how individuals within a network interact with one another and their impact on the network as a whole. As mentioned in previous sections, the various measurements of centrality highlights an individual node's involvement in the graph, whether it be how central the node is or how much information comes from or passes through an individual, as a few examples.

After analyzing the visualizations created by the application, it is up to the user to tease out the humanistic questions that graphs prompt. The usefulness of the application is limited; however, the information that is provided by the app provides the user with a great base of understanding for their research or independent exploration. It is then up to the user to take the information they gathered and seek out other tools to deepen their exploration on their data.

This network analysis application is created with the intent of being a low barrier to entry tool, with a goal of presenting large amounts of textual data in visual forms, condensing the material into one connected graph. This process, in Digital Humanities is called ”distant reading”. A common term used in this field is nicely defined in the article *On Close and Distant Reading in Digital Humanities* [10] which states, “[Distant reading] aims to generate an abstract view by shifting from observing textual content to visualizing global features of a single or of multiple text(s)”. For example, a user could link the network analysis application to a novel that they have yet to read or have not read in awhile, look at the various visualizations, and come away with a general idea of who the most important characters are their connections with one another. When paired with close reading, or the standard approach text where you deeply read a text to uncover themes and finer details, distant reading can help paint and clear, detailed picture of the novel (or other source material).

4.2 Creating a Network Analysis Application

4.2.1 The GUI

The GUI for the accompanying network analysis application takes on a minimalist design with functionality at the forefront if its design. The user is first directed to upload a csv file that they would like analyzed. After successfully uploading the file, all of the buttons used to create graphs based on the various centrality measurements become available for the user to interact with. Each of the buttons calculate the desired centrality for each of the nodes in the graph and display the graph. The nodes are colored based on the centrality of each node, which is described to the user through clicking on the “Analysis Help” button shown in Figure 7 later in the paper. The figure of the graph appears on screen for the user to scan through, zoom in and out on, and even save the figure. A flowchart of the application is presented below to emphasize this explanation.

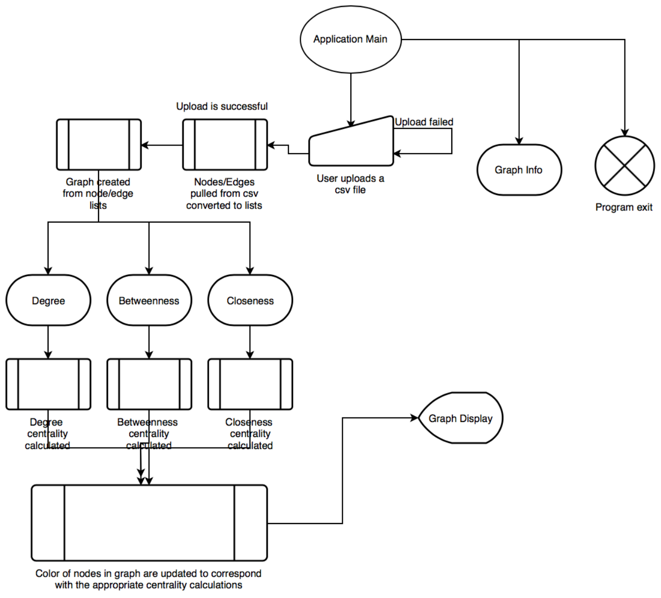
Application Main
Upload is successful
Upload failed
Graph created from node/edge lists
Nodes/Edges pulled from csv converted to lists
User uploads a csv file
Graph Info
Program exit
Degree
Betweenness
Closeness
Degree centrality calculated
Betweenness centrality calculated
Closeness centrality calculated
Graph Display
Color of nodes in graph are updated to correspond with the appropriate centrality calculations

Figure 6: A flowchart detailing how the application works, including the processes that occur at each step.

4.2.2 Converting Files to a Network

After the user links their csv file to the program, the application begins handling the file and converting it to a graph in the background. To do this, the program reads through the first two columns of the file, which are set up to contain the sender and receiver of an interaction between the two. All of the senders and receivers are placed in to separate lists that are later used to create the graph.

4.2.3 Creating the Graphs

Every individual recorded in the csv is represented by a node and their interaction is marked by an edge between their two nodes. Using Networkx, edges are added to the graph by pairing up entries from both the sender and receiver lists with matching list indices. After the graph is created, the program waits for the user to select which type of centrality they want presented. Each button, when pressed, calls a function that calculates the appropriate centrality measurements for all the nodes within the graph. This is done by utilizing the built-in centrality calculation functions of Networkx. These functions return a dictionary in Python that contains all nodes in the graph and their corresponding centrality measurements. Printing this dictionary would look like "(name of person): (centrality value)," for all people in the network. Using this dictionary, the program updates the color of each node in the previously created graph to correspond to the centrality value of the individual the node represents. The node with the highest centrality values are red, the node with the lowest centrality value is blue, and nodes in between are colored appropriately between those two colors.

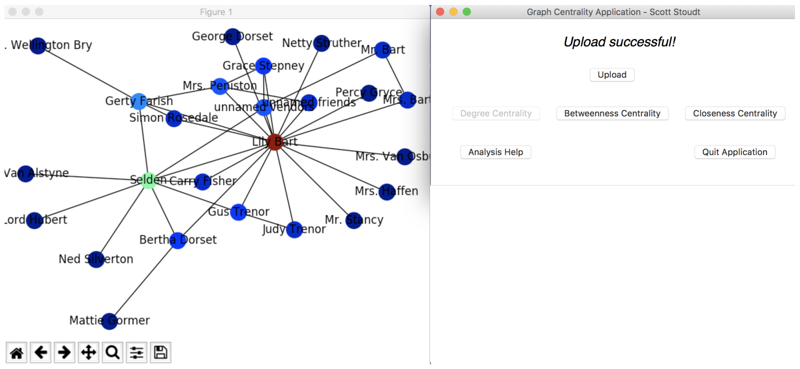

Figure 7: An example of the GUI and a graph that is created in terms of the measurements of degree centrality.

4.2.4 Possible Future Extensions to the Application

Although not a major issue, this application is heavily reliant on receiving an appropriately formatted csv file in order to produce the graph. Improving this application would happen by allowing the user to link different types of files, such as .txt files. In order for this to be possible, the program would have to be taught how to read such files and interpret the context of the words it is reading. This would place a heavy burden on the computer to figure out which characters are talking to each other and what the nature of their conversations are. By doing this, the program could tell a user more about the connections between the individuals by mapping the edges between nodes to appropriate natures of interaction. The current version of the software does its job of visualizing the types of centrality covered in this paper, but these ideas could take things to a more advanced level.

5 Conclusion

The goal of this paper was to introduce the concepts of graph centrality and network analysis. In doing so, we broke down three important types of centrality measurements: degree, betweenness,and closeness. After defining the types of centrality and breaking down the formulas for calculating them, the paper discussed their usefulness in different fields of study. Next, we looked at the application of graph centrality in investigating social structures, or social network analysis. Finally, this paper breaks down the network analysis application that was created to complement information that has been presented. This included the creation process, inner workings of the program, and how it is used to analyze graphs created from textual data sets.

Hopefully, after breaking down the logic behind these graph centrality calculations and grounding all the information with an application that produces comprehensive visuals, you have retained a better understand of graph centrality and its application in network analysis.

6 Appendix

```
'''
main.py
Jr. IS Software: Graph Centrality CentralityApplication
Spring '18 CS120- The College of Wooster

Writen by Scott Stoudt
'''
from Tkinter import *
import tkMessageBox
from tkFileDialog import *
import networkx as nx
import matplotlib
import csv
matplotlib.use("TkAgg")
from matplotlib import pyplot as plt

class CentralityApplication:
    global senderList
    senderList = []
    global receiverList
    receiverList = []
    global centralityList
    centralityList = []
    global G
    global degVals
    degVals = []
    global betVals
    betVals = []
    global closeVals
    closeVals = []
    def upload(self):
        #asks for file to be analyzed (csv in this case)
        filename = askopenfilename(initialdir = "/Desktop",title
    = "Select file",filetypes = (("csv files","*.csv"),("all
    files","*.*")))

        #reset all variables
        self.clearGraph()

        #checks if upload was successful or not
        if(filename != ''):
            self.label.configure(text = "Upload successful!")
            self.graph1_button.configure(state=NORMAL)
```

```python
            self.graph2_button.configure(state=NORMAL)
            self.graph3_button.configure(state=NORMAL)
            self.info_button.configure(state=NORMAL)
            #reads in senders/receivers from csv to create lists of each
            with open(filename, 'rU') as csvData:
                nr = csv.reader(csvData)
                for row in nr:
                    temp1 = str(row[0])
                    temp2 = str(row[1])
                    senderList.append(temp1)
                    receiverList.append(temp2)
                self.makeGraph(senderList, receiverList)
        else:
            self.label.configure(text = "Upload not successful!")
            self.graph1_button.configure(state=DISABLED)
            self.graph2_button.configure(state=DISABLED)
            self.graph3_button.configure(state=DISABLED)
            self.info_button.configure(state=DISABLED)

    #Creates the graph using senders/receivers as nodes
    #Then creates an edge from the sender to its appropriate receiver
    def makeGraph(self,senderList,receiverList):
        global G
        G = nx.Graph()

        for i in range(len(senderList)):
            G.add_edge(senderList[i], receiverList[i])

    #makes the matplotlib graph visible
    def showGraph(self, values):
        valMap = [values[node] for node in G.nodes()]
        nameMap = G.nodes()
        #assigns color of node based on desired centrality value
        nx.draw(G, with_labels=True, node_color=valMap , cmap=plt.cm.jet)
        plt.show()

    #calls for calculation of degree centrality values and sends the
    #values to be used to make the graph
    def degButton(self):
        plt.clf()
        self.graph1_button.configure(state=DISABLED)
        self.graph2_button.configure(state=NORMAL)
        self.graph3_button.configure(state=NORMAL)
        self.calculateDegree()
        self.showGraph(degVals)

    #calls for calculation of betweenness centrality values and sends the
    #values to be used to make the graph
    def betButton(self):
        plt.clf()
        self.graph1_button.configure(state=NORMAL)
        self.graph2_button.configure(state=DISABLED)
        self.graph3_button.configure(state=NORMAL)
        self.calculateBetween()
        self.showGraph(betVals)

    #calls for calculation of cloeness centrality values and sends the
    #values to be used to make the graph
    def closeButton(self):
        plt.clf()
        self.graph1_button.configure(state=NORMAL)
        self.graph2_button.configure(state=NORMAL)
        self.graph3_button.configure(state=DISABLED)
        self.calculateCloseness()
        self.showGraph(closeVals)

    #calculates degree centrality of all nodes in graph G
    def calculateDegree(self):
        #dictionary with 'node_name':'centrality value'
        global degVals
        #uses networkx function to calculate centrality values for each nodes
        #with respect to all other nodes in graph G
        ###exact formula used is discussed in paper portion
        degVals = nx.degree_centrality(G)

    #calculates betweenness centrality of all nodes in graph G
    def calculateBetween(self):
        #dictionary with 'node_name':'centrality value'
        global betVals
        betVals = nx.betweenness_centrality(G)

    #calculates cloeness centrality of all nodes in graph G
    def calculateCloseness(self):
        #dictionary with 'node_name':'centrality value'
        global closeVals
        closeVals = nx.closeness_centrality(G)

    #resets all lists, graph, etc. when a new csv is linked to the program
    def clearGraph(self):
        global senderList
        senderList = []
        global receiverList
        receiverList = []
        global centralityList
        centralityList = []
        global degVals
        degVals = []
        global betVals
        betVals = []
        global closeVals
        closeVals = []

    #Creates a pop up message to explain what the colors of the nodes in the graph mean
    def tipPopup(self):
        tkMessageBox.showinfo("Node Color Info", "The closer a node is to RED, the higher the centrality. The closer a node is to BLUE, the lower the centrality.")


    #creation of the tkinter GUI
    def __init__(self, master):
        self.master = master
        master.title("Graph Centrality Application - Scott Stoudt")

        self.label = Label(master, text="Please upload the csv file you would like to analyze!")
        self.label.configure(font=('helvetica', 20, 'italic'))
        self.label.grid(row=0, column=0, columnspan=3, pady=20)

        self.upload_button = Button(master, text="Upload", command=self.upload)
        self.upload_button.grid(row=1, column=1, columnspan=1)

        self.graph1_button = Button(master, text="Degree Centrality", command=self.degButton, state =DISABLED)
        self.graph2_button = Button(master, text="Betweenness Centrality", command=self.betButton, state =DISABLED)
        self.graph3_button = Button(master, text="Closeness Centrality", command=self.closeButton, state =DISABLED)

        self.graph1_button.grid(row=2, column=0, columnspan=1, pady=30, padx=(30,10))
        self.graph2_button.grid(row=2, column=1, columnspan=1, pady=10, padx=10)
        self.graph3_button.grid(row=2, column=2, columnspan=1, pady=10, padx=(10,30))

        self.info_button = Button(master, text= 'Analysis Help', command=self.tipPopup, state =DISABLED)
        self.info_button.grid(row=3, column=0, padx=(30,10))

        self.close_button = Button(master, text="Quit Application", command=root.destroy)
        self.close_button.grid(row=3, column=2, columnspan=1, padx=(10,30))


#Main window tkinter sizing/positioning
root = Tk()
RWidth = root.winfo_screenwidth()
RHeight = root.winfo_screenheight()
root.geometry(("%dx%d+%d+%d")%(RWidth/3,RHeight/4,RWidth-550,0))
root.minsize(550, 250)
root.maxsize(550, 250)
my_gui = CentralityApplication(root)
root.mainloop()
```

References

[1] https://www.sci.unich.it/~francesc/teaching/network/betweeness.html.

[2] (2015, May). Game of nodes: A social network analysis of game of thrones. https://gameofnodes.wordpress.com/2015/05/06/game-of-nodes-a-social-network-analysis-of-game-of-thrones/.

[3] (2018a, Feb). Betweenness centrality (centrality measure). https://www.geeksforgeeks.org/betweenness-centrality-centrality-measure/.

[4] (2018a, Feb). Centrality. https://en.wikipedia.org/wiki/Centrality.

[5] (2018b, Feb). Closeness centrality (centrality measure). https://www.geeksforgeeks.org/closeness-centrality-centrality-measure/.

[6] (2018c, Feb). Degree centrality (centrality measure). https://www.geeksforgeeks.org/degree-centrality-centrality-measure/.

[7] (2018b, Apr). Sociology. https://en.wikipedia.org/wiki/Sociology#Computational_sociology.

[8] Erkan, G. and D. R. Radev (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research 22*, 457–479.

[9] Hagberg, A. A., D. A. Schult, and P. J. Swart (2008). Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman (Eds.), *Proceedings of the 7th Python in Science Conference*, Pasadena, CA USA, pp. 11 – 15.

[10] Jnicke, S., G. Franzini, M. F. Cheema, and G. Scheuermann (2015). On Close and Distant Reading in Digital Humanities: A Survey and Future Challenges. In R. Borgo, F. Ganovelli, and I. Viola (Eds.), *Eurographics Conference on Visualization (EuroVis) - STARs*. The Eurographics Association.

[11] Kutty, S., R. Nayak, and L. Chen (2014). A people-to-people matching system using graph mining techniques. *World Wide Web 17*(3), 311–349.

[12] Kydros, D. and A. Anastasiadis (2015). Social network analysis in literature: The case of the great eastern. *Proceedings of the 5th European Congress of Modern Greek Studies of the European Society of Modern Greek Studies 4*, 681–702.

[13] Saryuce, A. E., E. Saule, K. Kaya, and U. V. Catalyurek (2015). Regularizing graph centrality computations. *Journal of Parallel and Distributed Computing] 76*, 106–119.

[14] Shipman, J. W. (2010). `http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html`.