Analysis of the Categorization of Twitter as a Hybrid Social and Information Network as a Tool for Politicians

Presented to the S. Daniel Abraham Honors Program in Partial Fulfillment of the Requirements for Completion of the Program

Stern College for Women
Yeshiva University
May 7, 2019

Avital Greenberg

Mentor: Professor Joshua Waxman, Computer Science
Introduction

Social media platforms today can be utilized for two main purposes, maintaining social connections and spreading information. Therefore, media platforms can be categorized as either social networks or information networks. According to the Merriam-Webster Dictionary, a social network is defined as “an online service or site through which people create and maintain interpersonal relationships,” implying that the main use of the service is for people to engage with friends, family, peers, and colleagues on a personal basis through written posts, pictures, or direct messages. According to Myers et al., an information network is defined as a “structure whose dominant interaction is the dissemination of information,” meaning that the primary use is either providing or gaining new information through written posts, pictures, and links to other sources (Myers et al., 2014).

The simplest and most superficial way to differentiate between these two categories of social media is through the type of connection that is made. Social networks have “friend” relationships wherein when a friend request is made, both users become friends of each other. Examples of this type of relationship are evident in Facebook and LinkedIn. On the other hand, information networks have a “following” connection, where a user can follow someone, but that user does not have to automatically follow back. Examples of this model are Twitter and YouTube. One could argue that the type of relationship model is indicative of its categorization as a form of social media. If a platform has a friend relationship, it maintains connections between social circles and keeps you up to date on friends’ personal lives, and if a platform has a following relationship, it allows a user to choose what kind of information he wants to see without any obligation of reciprocating an interpersonal relationship with other users (Parr, 2010).
However, Twitter as a media platform does not automatically determine itself as an information network simply on the basis of its connection model. The standard Twitter follow relationship is primarily about consuming information, but still many follows are built on social ties. A deeper understanding of the Twitter network is required in order to correctly categorize it and analyzing the graph representation of users can assist in the determination of the dominant Twitter interaction. Characterizing the properties of the Twitter follow graph such as degree discrepancy, connected components, shortest path lengths, clustering coefficients, two-hop neighborhoods, and degree assortativity is helpful in categorizing the Twitter network. The analysis of these topological features of the graph indicate that Twitter is a hybrid network; in some aspects Twitter allows users to maintain interpersonal relationships, while in other aspects it is conducive to spreading information far and wide.

In order to use a network effectively, it is important to understand its proper categorization. The hybrid behavior of Twitter as a network makes it a powerful resource, as it has the potential to achieve the two most important elements of social media. Therefore, many politicians utilize Twitter as a social media page because of this duality. Both during election campaigns and throughout their terms, politicians maintain seemingly personal relationships with constituents, party members, and opposition leaders as well as convey their platforms and agendas to their audience. However, the complexities that come with a hybrid network such as Twitter force politicians to balance increasing favorability amongst the electorate by maintaining social connections and effectively communicating their ideas in a way that will impact their constituents.

**Social and Information Network Properties**
Social networks and information networks have different trends in specific properties of their follow graphs that allow for the proper categorization of each. My research included analyzing properties of the Twitter graph such as degree discrepancy, connected components, shortest path lengths, clustering coefficients, two-hop neighborhoods, and degree assortativity.

A social network’s goal is sharing personal and professional experiences with a user’s friends. Therefore, a social network’s follow graph tends to exhibit properties such as high degree assortativity, small shortest path lengths, large connected components, high clustering coefficients, and a high degree of reciprocity. An information network’s goal is disseminating large amounts of information and the dominant interaction is following outlets that provide the information users are looking to receive. Its follow graph tends to exhibit properties such as large vertex degrees, lack of reciprocity, and large two-hop neighborhoods.

An example of a social network that we can draw conclusions and compare Twitter to is Facebook. Facebook is unarguably a social network; not only does it have the typical friend model connection, it’s main use is for maintaining interpersonal relationships. In fact, it is a poor forum for disseminating and receiving information since the self-built social groups, as well as personally tailored algorithms for commercial interests, create an echo chamber in which a user is only provided a narrow scope of information. Previous research similar to that of this paper has been conducted on the Facebook graph and gives insight as to the expected structure of a social network. The study “The Anatomy of the Facebook Social Graph” by Ugander et al., computed graph features including degree distributions, path lengths, clustering coefficients, two-hop neighborhoods, and assortativity. They found that
the average path length between users was 4.7; the largest connected component contained 99.91 percent of Facebook users; the clustering coefficient decreases as degree increases; the unique two-hop neighborhood of a user with 100 friends is 27,500 and the non-unique two-hop neighborhood of the same user is 40,300; and there is a positive correlation for degree assortativity, implying that the degree of your friends is similar to your own degree.

**Methods and Data**

A graph data structure is a set of vertices or nodes combined with a set of edges that connect pairs of vertices together. A graph can be either directed or undirected. A directed graph indicates the edges have a single direction, connecting vertices in only one way, while an undirected graph means the edges do not have a specific direction, rather connect both vertices to each other. The Twitter follow graph is a directed graph, where the vertices represent Twitter users and edges represent the following relationship. This edge is one way, meaning that an edge from vertex A to vertex B indicates that user A follows user B. An example of an undirected graph is a model representation of the Facebook network graph. Since a friend relationship on Facebook is reciprocal, meaning you cannot be friends with someone unless they are friends with you, the edge is undirected and represents the existence of a friend connection between users. Twitter differentiates between followers and following relationships, thereby making a directed graph a more accurate form of representation. A directed graph differentiates between inbound edges and outbound edges of a specific vertex. In the case of the Twitter graph, the inbound edges of a vertex represent all the followers that user has, and the outbound edges represent how many other users that user is following (referred to as friends by the Twitter API).
NetworkX is a Python module that allows a programmer to build and store a graph data structure. Hagberg et al., in their discussion of NetworkX in their article, “Exploring Network Structure, Dynamics and Function using NetworkX,” explain that NetworkX provides flexibility in representing connections between objects for easy graph analysis and implements many common algorithms for graph structure calculations (Hagberg et al., 2008). I used NetworkX to represent my Twitter follow graph in code and made use of many of the built-in methods that NetworkX provided for analyzing graph characteristics. Additionally, I further manipulated and implemented their methods within my own hand written code in order to acquire even more data and understand it from this perspective.

In order to conduct an analysis, I recreated a model of the Twitter follow graph as I did not have access to the actual Twitter graph. The results in this paper are based on algorithms that were run on a Twitter dataset from Stanford University’s Stanford Network Analysis Project (SNAP) which contained 81,306 nodes and 1,768,149 edges representing Twitter circles (McAuley and Leskovec, 2012). I began to create a more updated model of the Twitter follow graph to run the same tests on. Using a Python module called Tweepy, I connected to the Twitter API (Application Programming Interface) and accumulated real time Twitter users, their followers, and their friends.

```python
import tweepy
from tweepy import OAuthHandler

# import twitter keys from separate file
from keys import keys
ckey = keys['ckey']
csecret = keys['csecret']
atoken = keys['atoken']
asecret = keys['asecret']

# initialize and authorize access to Twitter
auth = OAuthHandler(ckey, csecret)
auth.set_access_token(atoken, asecret)
```
# connect to api with our OAuthHandler
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True, compression=True)

However, because Twitter limits the amount of data a single user can receive per every fifteen minutes and because of the limited computing machine I had access to, this model data set is not yet complete and has not yet been tested for the same graph characteristics. It is important to note the SNAP model data set represents less than 0.01 percent of all Twitter users worldwide, making this research is quite limited. Further improvements to this study would be to continue acquiring real Twitter data to build a more representative model of the Twitter follow graph and see if it provides the same results as an older version of the graph.

My approach to gathering a sample of real Twitter users was that of a breadth-first algorithm. I began with a random Twitter user, collected all his followers, stored them as edges by writing out to a text file, and added them to a global set of users. Similarly, I then collected the users that he is following, wrote them out as edges to a text file and added them to the global set. I then collected the same data for each user in the global user set.

```python
def get_followers(user, start=False):
    # use the global set users to add more users to go through
global users
followers = tweepy.Cursor(api.followers, screen_name=start,
count=200).items()
    while True:
        try:
            follower = next(followers)
            # if this is the starting user, add it's followers to a set
            # only the start user can do this because the rest of the
            # code loops through each user in the set
            if start:
                # add follower to set users
                users.add(follower.screen_name)
        except tweepy.TweepError:
            print("in a follower error")
            time.sleep(60 * 15)
            follower = next(followers)
        except StopIteration:
            break
        fout.write(str((follower.screen_name, follower.id_str)) + 't' + 
                   str((user.screen_name, user.id_str)) + 'n')
```
def get_friends(user, start=False):
    friends = tweepy.Cursor(api.friends, screen_name=start, count=200).items()
    while True:
        try:
            friend = next(friends)
            # if this is the starting user, add it's friends to a set
            # only the start user can do this because the rest of the
            # code loops through each user in the set
            if start:
                # add friend to set users
                users.add(friend.screen_name)
        except tweepy.TweepError:
            print("in a friends error")
            time.sleep(60 * 15)
            friend = next(friends)
        except StopIteration:
            break
        fout.write(str((user.screen_name, user.id_str)) + ' ' + \
                   str((friend.screen_name, friend.id_str)) + '

# start with random user
start = "noemineko"
user = api.get_user(start)
# a set of users already to go through
users = set()

get_followers(user, start=True)
get_friends(user, start=True)
for account in users:
    user = api.get_user(account)
    get_followers(user)
    get_friends(user)

Because of the restrictions to the amounts of data collected from Twitter at a given
time, this code ran for many hours; therefore, I only had this code go two levels deep from
the starting user. After the code completed running, I manually inserted a new starting user
by selecting a random user, whose data had not yet been collected, from the followers or
friends of a user in the second level. At other times, I inserted a new starting user by
randomly selecting a user on Twitter that I had not previously collected data on, nor was in
my dataset previously. This was done with the intention of avoiding creating a network that
was completely connected from the outset, rather so that a user that was originally
disconnected from the rest of the graph may or may not become connected through a similar
follower or friend.

Graph Characteristics
I. Degree Discrepancy

The amount of edges a vertex has determines its degree. When analyzing a directed graph, such as the Twitter follow graph, there is a differentiation between in-degree and out-degree. In-degree is the amount of inbound edges of a vertex and the out-degree is the amount of outbound edges of a vertex. The difference between in-degree and out-degree of vertices in the graph shows whether Twitter is commonly used a social platform or more commonly used as an information gathering network. In a social network, one would expect a small difference between in-degrees and out-degrees because these relationships would be reciprocated, whereas in an information network there would be a large discrepancy between in-degrees and out-degrees because users are not maintaining personal relationships.

For example, in the graph in Figure 1, Cindy has an out-degree of two, because there is one outgoing edge to both Adam and Bill. Meanwhile, Cindy has an in-degree of 1 as there is only one incoming edge from Adam. Therefore, the difference between in-degree and out-degree is one, implying that within Cindy’s social circle, she has approximately the same number of followers as friends and her relationships are reciprocated.

Figure 1. In/out- degrees
For this analysis, I considered a large discrepancy between in-degrees and out-degrees to be more than absolute value of one thousand. A difference of one thousand between the number of followers and the number of friends of a single user implies that the Twitter relationship is not being used as a way to maintain a social connection, but rather to share information with or acquire information from many other users. In the NetworkX graph, in-degrees and out-degrees are saved as attributes of a vertex with an access of O(1) time complexity. The process of accessing the number of degrees and taking the difference took O(N) time, where N is the total number of vertices in the graph. Additionally, I found the average discrepancy of the graph in order to see if the average discrepancy is itself large.

```python
# function that returns the number of nodes that have a discrepancy of 1000 or greater
# between in and out degrees
def process_degrees(dg):
    # counter to count how many nodes have a large discrepancy in in vs out degrees
    bigDiff = 0
    # for every node in the graph
    for node in dg.nodes():
        # get its in and out degree
        in_degree = dg.out_degree(node)
        out_degree = dg.in_degree(node)
        # if the abs value of the difference is greater than or equal to 1000
        if abs(in_degree - out_degree) >= 1000:
            # add to the count of large discrepancies
            bigDiff += 1
    return bigDiff

# function that returns the average discrepancy of in and out degrees
# of the nodes in the graph
def avg_diff(dg):
    # counter
    totalDiff = 0
    # number of nodes in the graph
    nNodes = len(dg)
    # for each node in the graph
    for node in dg.nodes():
        # get its in and out degrees
        in_degree = dg.out_degree(node)
        out_degree = dg.in_degree(node)
        # add the abs value of the difference to the count
        totalDiff += abs(in_degree - out_degree)
    # return the average
    return totalDiff / nNodes
```

I found that the number of vertices with a difference between in and out-degrees more than one thousand was insignificant, less than 0.1 percent of vertices in the graph. The
average discrepancy was also small, implying that most users do not have many more followers than friends or vice versa. These findings are behaviors expected in a social network since many connections are reciprocated and users have the ability to maintain social relationships with the amount of connections they have.

II. Connected Components

A connected component of an undirected graph is a subgraph in which any two vertices are connected to each other by paths and which is connected to no additional vertices in the super-graph. The Twitter graph is a directed graph, so there is a distinction between strongly and weakly connected components. A directed subgraph is strongly connected if there is a path from every node to every other node. A directed subgraph is weakly connected if the underlying undirected graph is connected.

The super-graph of Figure 2 includes all the vertices Adam, Bill, Cindy, Deborah, Eloise, and Frank and there are two separate subgraphs. The first subgraph consisting of Adam, Bill, and Cindy is only weakly connected, because once edge direction is considered, there is no path from Bill to Cindy. On the other hand, the subgraph consisting of Deborah,
Eloise, and Frank is strongly connected since there is a path from Eloise to Frank and Eloise to Deborah via Frank; there is a path from Frank to Deborah and Frank to Eloise via Deborah; and there is a path from Deborah to Eloise and Deborah to Frank via Eloise.

In distinguishing between a social network and an information network, the largest strongly connected component is revealing. The percentage of how many vertices contained in the largest strongly connected component as compared to the total number of vertices in the graph is indicative of the nature of connections on Twitter. The expectation of the percentage of vertices in the largest weakly connected component is close to 100 percent since the direction of edges is not considered. Therefore, a user that has at least one connection will likely have a path to every other user, and this is not suggestive towards a social or information network specifically. Rather, the larger the percentage of total vertices in the largest strongly connected component, the more indicative of a social network. Since the edges are directional, a larger strongly connected component would represent greater reciprocity in connections.

The NetworkX module has generator method for directed graphs that takes in a directed graph G and generates sets of vertices, each one representing a strongly connected component of G. This generator method in NetworkX uses a non-recursive version of Tarjan’s algorithm with Nuutila’s modifications. Tarjan’s algorithm uses depth first search to find strongly connected components and runs in O(V + E) where V is the number of vertices and E is the number of edges in the graph. Nuutila’s modifications to Tarjan’s algorithm mainly improves efficiency in handling sparse graphs and graphs with many trivial components (Hagberg et al., 2008).

# For either weak or strong, takes in parameter char 's' for strong or 'w' for weak # use built-in generators for finding connected components, returns list of
# generator items

def connected_component(dg, type):
    if type == "s":
        return list(nx.strongly_connected_components(dg))
    else:
        return list(nx.weakly_connected_components(dg))

# method to see the percentage of nodes in each component

def percentage_connected(dg, type):
    # use method to get either all the weakly or strongly connected components
    components = connected_component(dg, type)
    # get the largest component from that list
    largest_component = max(components, key=len)
    numComponents = len(largest_component)
    total = dg.number_of_nodes()
    # to get the percentage of nodes in the largest connected component, divide the
    # num of nodes by the total amount of nodes in the graph and multiply by 100
    return numComponents / total * 100

I analyzed the largest strongly connected component and largest weakly connected component of the Twitter follow graph and found that the largest weakly connected component contained 100 percent of the nodes in the super-graph. However, the largest strongly connected component contained 84 percent of all the nodes in the graph. This implies that many edges in the graph are not closely connected, meaning that users only follow other users for information dissemination or consumption and not to maintain a social relationship.

III. Shorest Path Lengths

Determining shortest path lengths means determining a path between two vertices in a graph such that the sum of the weights of its constituent edges is minimized. The distribution of shortest path lengths quantifies how tightly users are connected; the shorter the path lengths, the more indicative of a social network. According to Leskovec et al., Facebook and other social networks have short average shortest path length that decrease with an increase of the size of the graph (Leskovec et al., 2005). In the example graph in Figure 3, each edge is a weight of one. If we were looking for the shortest path between Adam and Frank, there are two possible traversals. The first is to go from Adam to Bill, which is a weight of one,
and then from Bill to Eloise, from Eloise to Deborah, and finally from Deborah to Frank, which gives a total sum of four. However, there is a shorter path, from Bill directly to Frank, making the sum of the shortest path length two.

The NetworkX module has a method that takes in a graph G and returns the average shortest path length. It determines the average shortest path length using the formula found in Formula 1, where V is the set of all vertices in the graph, s and t are vertices, d(s, t) is the shortest path length between s and t, and n is the total number of vertices in the graph.

$$a = \sum_{s,t\in V} \frac{d(s, t)}{n(n - 1)}$$

Formula 1

def shortest_path_lengths(dg):
    # compute average shortest path lengths in the graph
    return nx.average_shortest_path_length(dg)

I found that the average shortest path length was 4.1. This result is shorter than the average shortest path length of Facebook which is 4.7 (Ugander et al., 2011). Because the
average shortest path length of Twitter is surprisingly less than Facebook, and because it has been previously determined that this average would continue to decrease with a larger graph to test on, Twitter exhibits closely connected communities with few degrees of separation and in this way it behaves like a social network.

IV. Clustering Coefficient

Clustering coefficient is the measure of the degree to which vertices in a graph tend to cluster together. In social networks, this measures the fraction of users whose friends are friends themselves; a high clustering coefficient is a property attributed to social networks.

Ugander et al., has determined average clustering coefficients expected in a social network and has also concluded that the average clustering coefficient decreases as the degree of a vertex increases (Ugander et al., 2011). The conclusions from this research were done using data from Facebook, established as a social network, as discussed above.

However, the friend connection on Facebook is not directed, thereby making the graph undirected. In order to compare the clustering coefficients of Twitter to those of Facebook, it is necessary to extract the mutual graph of the Twitter follow graph. The mutual graph is an undirected graph that consists of users whose edges are reciprocated.

![Figure 4. Mutual Graph](image-url)
In Figure 4, Graph B represents the mutual graph of Graph A, meaning that the edges in
Graph B are only those that are reciprocal edged in Graph A. For example, the edge between
Adam and Bill is not reciprocated; they are only connected in one direction and therefore not
included in the mutual graph. On the other hand, Adam and Cindy have a bidirectional edge,
meaning their connection is reciprocated and included in the mutual graph.

I extracted the mutual graph from the original graph by finding the vertices that are
predecessors of each other; in other words, if there was a directed edge from the first vertex v
to the second vertex t and there was a directed edge from t to v, then their connection is
mutual and that edge was included in the mutual graph. This may not be the most efficient
algorithm as it is a double for loop, making the run time $O(N^2)$. However, I checked if an
edge was in my set of mutual edges before continuing the inner loop, allowing the it to break
if the connection has already been found earlier.

```python
def mutual_graph(dg):
    mutuals = set()
    # for each node in the graph
    for node in dg:
        # use built-in method to find the predecessors of the node
        pred = dg.predecessors(node)
        # go through each of the predecessors
        for node2 in pred:
            # check if not in the set, if it is, does not continue
            if (node, node2) not in mutuals or (node2, node) not in mutuals:
                # and get their predecessors
                pred2 = dg.predecessors(node2)
                # if the original node is in the predecessors of the second node
                if node in pred2:
                    # add it to the set of tuples for edges
                    mutuals.add((node, node2))
                    mutuals.add((node2, node))
    # now create a new mutual graph
    mg = nx.Graph()
    for edge in mutuals:
        mg.add_edge(edge[0].strip(), edge[1].strip())
    return mg
```

I then found the average clustering coefficient of vertices of the same degree. The
NetworkX module has a built-in clustering method that computes the clustering coefficient
for all the vertices in a graph and returns a dictionary with the key being the vertex of the
As shown in Formula 2, clustering for unweighted directed graphs is defined by Hagberg, et al. in the NetworkX documentation as “the fraction of all possible directed triangles,” where \( T(u) \) is the number of directed triangles through vertex \( u \), \( \deg^\text{tot}(u) \) is the sum of in and out-degree of \( u \) and \( \deg^\leftrightarrow(u) \) is the reciprocal degree of \( u \) (Hagberg, et al., 2008).

\[
c_u = \frac{1}{\deg^\text{tot}(u)(\deg^\text{tot}(u) - 1) - 2\deg^\leftrightarrow(u)}T(u),
\]

Formula 2

Using the dictionary returned from this method, I then calculated the average clustering coefficient of all vertices of the same degree.

```python
def avg_clustering_per_degree(dg):
    # get the nodes of the mutual graph by calling mutual_graph() function
    mg = mutual_graph(dg)
    # coefficients is a dict with the nodes in mutuals as keys and the clustering coefficient as the data
    coefficients = nx.clustering(mg)
    # now go through each node, get the degree and it to new dict, calculating the # average clustering coefficient for that degree
    coefficientByDegree = dict()
    for k in coefficients:
        # get the degree of the node
        degree = mg.degree(k)
        # if the degree is not already in the new dict then add it with the value # being the coefficient
        if degree not in coefficientByDegree:
            coefficientByDegree[degree] = coefficients[k]
        # if it is already in the new dict, then add to the value the coefficient # and divide by 2 to take average
        else:
            coefficientByDegree[degree] += coefficients[k]
            coefficientByDegree[degree] /= 2
    return coefficientByDegree
```

My results are consistent with the expectations of a social network; as degree increases, the average clustering coefficient decreases. At the lowest degree of 2, the average clustering coefficient is 0.66, while at the highest degree of 747, the average clustering coefficient is 0.0039. These findings indicate Twitter is like a social network as it has the
characteristic of having close communities in which a user’s friends are also friends amongst themselves.

V. Two-hop Neighborhoods

Two-hop neighborhoods refer to the set of vertices that are neighbors with a vertex’s neighbors. A neighbor of a vertex is any vertex that is one edge away from the starting vertex, or all the followers and friends of a Twitter user. A two-hop neighbor of a vertex is any vertex that is two edges away from the starting vertex, or all the followers and friends of a Twitter user’s followers and friends. The outbound one-hop neighborhood of Bill in the graph in Figure 5 includes Bill’s outgoing edges, or friends, namely, Cindy and Eloise. The two-hop neighborhood consists of both Cindy’s and Eloise’s outgoing edges, namely Adam and Frank. Bill’s inbound two-hop neighborhood would consist only of Adam’s inbound edges, namely Cindy.

![Figure 5. Two-Hop Neighborhoods](image)

For this research, analyzing the difference between unique two-hop neighborhoods and non-unique two-hop neighborhoods is informative in characterizing a graph as social or informational. The unique two-hop neighborhood indicates the removal of any duplicate users found in the neighborhood. If the unique and non-unique neighborhoods are close in
number of vertices, that implies that the number of edges between users of that neighborhood is low, which exhibits a lack of community structure. Also, it implies that adding a vertex, or gaining a follower, can dramatically increase the size of the two-hop neighborhood, which is more beneficial for information dissemination or collection. On the other hand, if the difference between the number of vertices in the unique and non-unique neighborhoods is not close in value, that implies that there are many duplicate edges between users of the neighborhood and indicates a community structure.

I looked at both the average difference between unique and non-unique neighborhoods of the graph for inbound edges, or the two-hop neighborhood of followers, and for outbound edges, or the two-hop neighborhood of friends, for each vertex in the graph. My approach to accumulating these neighborhoods was O(N^2) time complexity.

```python
# function that returns two dictionaries- one with the uniq nodes in all the inbound # two-hop neighborhoods of the graph as well as the amount of nodes, and one with the # non-uniq nodes in all the inbound two-hop neighborhoods with the amount of nodes
def inbound_twoHopNeighborhood(dg):
    # save all the two-hop neighborhoods for each node in a dict
    allUniq = {}
    allNonUniq = {}
    for node in dg:
        # unique can be a set because does not allow for duplicates
        uniq = set()
        # non unique is a list to allow for duplicates
        nonUniq = []
        numUniq = 0
        numNonUniq = 0
        # get the predecessors of origin node
        oneHop = dg.predecessors(node)
        # for each predecessor
        for oneHopNeighbor in oneHop:
            # get the predecessors of the second node
            twoHop = dg.predecessors(oneHopNeighbor)
            for twoHopNeighbor in twoHop:
                # if this is not already in the uniq set
                if twoHopNeighbor not in uniq:
                    # add it and add 1 to the count of uniq nodes
                    uniq.add(twoHopNeighbor)
                    numUniq += 1
                # either way, add it to the non unique list and add one to count
                nonUniq.append(twoHopNeighbor)
                numNonUniq += 1
        allUniq[node] = uniq, numUniq
```

allNonUniq[node] = nonUniq, numNonUniq
return allUniq, allNonUniq

# same as inbound, but for outbound two hop neighborhoods
def outbound_twoHopNeighborhood(dg):
    allUniq = {}
    allNonUniq = {}
    for node in dg:
        uniq = set()
        nonUniq = []
        numUniq = 0
        numNonUniq = 0
        oneHop = dg.neighbors(node)
        for oneHopNeighbor in oneHop:
            twoHop = dg.neighbors(oneHopNeighbor)
            for twoHopNeighbor in twoHop:
                if twoHopNeighbor not in uniq:
                    uniq.add(twoHopNeighbor)
                    numUniq += 1
                    nonUniq.append(twoHopNeighbor)
                    numNonUniq += 1
            allUniq[node] = uniq, numUniq
            allNonUniq[node] = nonUniq, numNonUniq
    return allUniq, allNonUniq

# this method will tell us the difference in nodes between unique and non-unique neighborhoods. takes in a graph and string if want inbound or outbound
def process_neighborhoods(dg, bound):
    totalDiff = 0
    if bound == "o":
        # get the unique and non-unique dictionaries
        unique, nonUnique = outbound_twoHopNeighborhood(dg)
        # for each key in graph
        for k in nonUnique:
            a = nonUnique[k][1]
            b = unique[k][1]
            diff = abs(a - b)
            totalDiff += diff
        return totalDiff / len(nonUnique)
    else:
        unique, nonUnique = inbound_twoHopNeighborhood(dg)
        for k in nonUnique:
            a = nonUnique[k][1]
            b = unique[k][1]
            diff = abs(a - b)
            totalDiff += diff
        return totalDiff / len(nonUnique)

I found the average difference between unique and non-unique two-hop neighborhoods to be about 1000 vertices for both inbound and outbound two-hop neighborhoods. With the average degree being about 44, this difference is relatively small compared to the expected difference of a social network; in the Facebook graph, the difference between unique and non-unique two-hop neighborhoods at a degree of 100 is about 12,800. These findings therefore indicate that Twitter is more conducive to
information dissemination and consumption since many vertices within the two-hop neighborhood are not connected themselves and addition of a single follower or friend has the potential to create a two-hop neighborhood of many new followers or friends to gain access to.

VI. Degree Assortativity

Degree assortativity is “the preference for a graph’s vertices to attach to others that are similar (or dissimilar) in degree” (Myers, et al., 2014). When analyzing the Twitter follow graph, it is important to evaluate the correlation for both in-degree and out-degree. There are four cases associated with each connection: the source vertex out-degree (SOD), the source vertex in-degree (SID), the destination vertex out-degree (DOD), and the destination vertex in-degree (DID).

As an example, from Figure 6, analyzing the edge between Bill and Eloise would present Bill as the source and Eloise as the destination. The source in-degree refers to all the inbound edges of the source, in this case the edges of Adam and Cindy. The source out-degree is all the outbound edges, in this case the edges of Eloise and Frank. Similarly, the destination in-degree would be the edge of Bill and the destination out-degree would be the edge of

![Figure 6. Degree Assortativity](image)
Deborah. Therefore, there are four different comparisons for degree assortativity, SOD vs. DOD, SID vs. DOD, SOD vs. DID, and SID vs. DID. Understanding the correlation for each of these comparisons can provide different insight on how Twitter functions as a network.

I used NetworkX’s built-in method that has the ability to distinguish between in-degree and out-degree on a directed graph for both the source and destination vertex of the edge and returns a float of assortativity of the graph by degree. This number is obtained by calculating the joint probability distribution, or mixing matrix, by degree according to M. E. J. Newman (Newman, 2003).

```python
# generator that yields the degree assortativity for each combo of SID, SOD, DID, and # DOD
def degree_assortativity(dg):
x = "in"
y = "out"
# built in method takes in optional parameters of string "in" and "out" for in # degree or out degree of the source and destination nodes
yield "degree assortativity SID vs. DID: ", nx.degree_assortativity_coefficient(dg, x, x)
yield "degree assortativity SID vs. DOD: ", nx.degree_assortativity_coefficient(dg, x, y)
yield "degree assortativity SOD vs. DID: ", nx.degree_assortativity_coefficient(dg, y, x)
yield "degree assortativity SOD vs. DOD: ", nx.degree_assortativity_coefficient(dg, y, y)
```

I found that there was a positive correlation for SOD vs. DOD, meaning that more people you follow, the more people those users are likely to follow. This makes sense from a social network perspective, because social users engage with other social users. SID vs. DOD also had a positive correlation, implying that the more followers you have, the more those followers follow other users. This is also logical in terms of a social network, since the more popular you are, the more those that engage with you will have bigger social circles. I found a negative correlation for SOD vs. DID, meaning that the more people you follow, the less followers those users have. This is very surprising since the presence of this edge would increase both the SOD and the DID by one, and it is not the behavior we would expect in a
social network. Additionally, there was a positive correlation for SID vs. DID, implying that
the more followers you have, the more popular the people you follow are. This is also
expected in a social network, since we expect popular users to be follow other popular users
(Myers et al., 2014). The correlations in three out of four of the cases are in line with
expectations of a social network, showing that Twitter does have properties that could
categorize it as a social network. The surprising result that is counterintuitive from what a
social network should behave as also serves to show that Twitter is more complex than a
typical social network.

Results

The results of analyzing these topological characteristics of the Twitter follow graph
present Twitter behaving like a social network in some ways and an information network in
other ways. Characteristics expected in a social network that are evident in Twitter include
high degree reciprocity, short average path lengths, decreasing clustering coefficient as
degree increases, and degree assortativity. These properties demonstrate close connections
between Twitter users and the ability to maintain interpersonal relationships. On the other
hand, Twitter also has characteristics expected of an information network such as a small
maximum strongly connected component and large two-hop neighborhoods. These
properties indicate a lack of community structure and facilitate the dissemination or
attainment of information from a large audience. This hybrid network characterization
implies that Twitter users can employ dual motivations with this platform. Users can utilize
Twitter to absorb and circulate information and ideas, as well as connect with friends and
establish communities.

Application to Politicians
Politician Graph Characteristics

To further understand how politicians use Twitter, I created a model Twitter graph with a focus on politicians to determine if this subgraph behaved similarly to the original graph. I specifically focused on top United States politicians including the president, vice president, senators and congressional representatives. I added these politicians, their followers, and friends to the existing graph and tested properties such as degree discrepancy, connected components, clustering coefficients, two-hop neighborhoods, and degree assortativity. As a note, I was not able to test average shortest path lengths because of the computational complexity and the limited computing resources available.

Unlike the average Twitter user, politicians presented a very large discrepancy between in-degree and out-degree, with the largest being a difference of more than a million. This is expected because millions of Twitter users follow these politicians’ accounts, but politicians do not reciprocate that connection with all their millions of followers. This trend is logical, as it is impossible for politicians to maintain personal relationships with every single follower. This behavior is different than that of the average Twitter user and is more indicative of politicians using Twitter as an information disseminating platform to impart their platforms and ideology to wide audience.

Other than that major difference, the combined graph of politicians and typical Twitter users is similar in behavior to the original testing graph. The addition of politicians still displayed the trend of decreasing clustering coefficient with increasing degree and did not significantly change degree assortativity that show Twitter’s social network behaviors, nor did it present a large maximum strongly connected component and large two-hop neighborhoods that portray Twitter’s behavior as an information network. These
characteristics indicate how even for politicians, Twitter has a dual power of providing effective communication between politicians and constituents through social relationships and information propagation.

**Politicians Use of Twitter**

Around the world, politicians use Twitter as a media resource outlet, often having specific professional accounts to engage with constituents and communicate with the electorate. The use of Twitter amongst politicians began as early as 2009, three years after Twitter was created, when verified accounts were introduced. The number of politicians that employ Twitter has increased significantly over the past ten years to the point where a politician must have a Twitter account in order to effectively campaign and participate with the public (Lee and Shin, 2012). In their study, “Sociology of Hyperlink Networks of Web 1.0, Web 2.0, and Twitter: A Case Study of South Korea,” Hsu and Park explain:

“[Twitter’s] high capacity in terms of delivering information across boundaries is undeniable, and the content of Tweets has generally remained intact… Furthermore, politicians can use Twitter as a bridge between themselves and the general public to establish a closer relationship and to extend their reach to broader audiences during nonelection years” (Hsu and Park, 2009).

In other words, even as early as 2009, politicians understood the capabilities that Twitter provides. It has the ability to convey to the public important information clearly, concisely, and authentically, and it has the forum to create a more interpersonal connection with constituents. Despite this known power of Twitter, many politicians had yet to take advantage of it; at the time of this study, only 22 of 296 assembly members in South Korea had Twitter accounts preceding elections in 2009, less than 10 percent of member of the National Assembly. However, by 2012, members of the National Assembly that were active on Twitter increased to 86.4 percent (Lee and Shin, 2012). The number of politicians on
Twitter has continued to increase until today; for example, all 100 United States Senators of the 115th Congress, active from January 2017 until January 2019, had a professional Twitter account (McGuinness, 2018).

With increased political use of Twitter and an understanding of the hybrid categorization of the network, politicians must learn how to productively retain both social connections and groups conducive to the spread of information. In their study “Are They Talking to Me? Cognitive and Affective Effects of Interactivity in Politicians' Twitter Communication,” Lee and Shin conclude that a politician’s social presence amongst the electorate generates more favorable reactions towards him but can hinder the ability for a person to recall his platforms and agendas (Lee and Shin, 2012). This finding further highlights the complexity and contradictory effects of Twitter’s hybrid network. Politicians may use Twitter to effectively communicate information on their platforms and ideas and they may interact with followers by responding to posts and personal messaging, but there is a cost to over-utilizing one aspect of this social media network. Those politicians that are highly interactive with followers on Twitter increase the positive attitude the user will have towards them but will decrease the user’s ability to recall information on the politicians’ platform or beliefs. Similarly, the politicians that keep distance from a personal relationship with followers, and instead provide mostly information, increase their credibility amongst constituents but not favorability. Politicians must find the fine balance between both parts of Twitter’s duality in order to successfully utilize its greatest capacities.

**Contributions**

There has been extensive research previously done on this topic and the contributions of my work to this discussion are the following:
- Developed and manipulated software to measure graphs and extract metrics on graph properties relevant to determining if it is a social or information network.

- Innovated a novel measure of categorizing these graphs, namely finding the difference between in-degree and out-degree of a single vertex. The implication of a small difference would mean that many edges of a vertex are reciprocated and there is a social connection. On the other hand, a large difference implies that these edges are not reciprocated, and connections are used mainly for accumulating or disseminating information to many people.

- Analyzed how these measure transfer to the subgraph of politicians and explored how politicians use Twitter.

- For the reader unfamiliar with these concepts, described some typical measures of social networks and the intuition behind them.

**Conclusion**

Twitter as a social media platform cannot be categorized as solely a social network or an information network, but rather a hybrid of these two types of networks. Analysis of the topological features of a model of the Twitter follow graph indicate that it in some ways behaves as one would expect a social network to behave, and in other ways behaves as one would expect of an information network. Using the Python module NetworkX and handwritten Python code, I evaluated different graph characteristics on model Twitter data from SNAP. The graph properties considered that were indicative social network behavior include high reciprocity, short average shortest path lengths, decreasing clustering coefficients as degree increases, and degree assortativity. The low average difference between in-degree and out-degree of a vertex shows high reciprocity and the existence of a
personal relationship made through Twitter connections. Short path lengths represent the close community structure that Twitter has and the ability to reach most other users in a small number of separations. Decreasing average clustering coefficients as degree increases is consistent with the trend of other social networks such as Facebook. Most of the results of degree assortativity are behaviors we expect in a social network, such as the likelihood for a popular user to follow and be followed by other popular users. The graph properties considered that were indicative information network behavior include a low maximum strongly connected component and large two-hop neighborhoods. The lack of many vertices in the largest strongly connected component portray the lack of community structure with many unconnected or unreciprocated edges. Large two-hop neighborhoods demonstrate the capability for information distribution at a large scale. The complexities these topological features present in Twitter's behavior establish Twitter as a hybrid network with proficiency in preserving both social and informational interactions.

The knowledge of Twitter’s ability to function as both a social and information network is vitally important in using it successfully as a social media site, particularly for politicians. Analysis of the model Twitter graph combined with United States politicians’ connections presents graph characteristics similar to that of the regular Twitter graph model; it behaves more similar to a social network in some ways and more similar to an information network in others. The major difference from the average Twitter user is that politicians have a very large discrepancy between in-degrees and out-degrees since millions of users follow them and they are unable to reciprocate the following to that many users. Besides for this trend, Twitter’s dual power is applicable to politicians as well; they can circulate large amounts of information clearly and concisely to constituents, party members, and opposition,
as well as preserve social interactivity with their audience. However, over-utilizing one aspect can have negative effects on a politician’s favorability and credibility amongst his followers. Therefore, politicians must learn to balance these two aspects of the Twitter network in order effectively use Twitter at its greatest potential.
Works Cited


