Project I - Social Media Data Analysis

Abstract:

The project aims at crawling social media data and performing analysis on the extracted data. The social media platform selected for this is Twitter. Twitter is one of the most well known platforms and extracting the data from it is also simplified by them. They provide Twitter APIs through which we can extract the data. We will further perform analysis on in it and build a friendship network.

Data Collection:

- Using the twitter API, data of any user whose account is public can be extracted. For this, we created a twitter developer account and built an application for our project. Through this, we got our consumer key, consumer secret, access key and access secret.
- These keys are used for authentication.
- We extracted the data using the Twitter API by passing these authentication keys.
- We have collected the data of 6 influential users. 4 of them are related to politics where 2 are democrats and 2 are republicans and remaining 2 are news channels. The users we selected are as follows:

I. realDonaldTrump
II. JoeBiden
III. KamalaHarris
IV. Mike_Pence
V. FoxNewsSunday
VI. FaceTheNation

- We collected the data of their friends and stored them in each user's dictionary. The friends are stored as a key having a value as a sorted list of their friends ID.
- For each user, we calculated the total number of friends and calculated the number of people who follows them back.
- Using the above values, we calculated the friends overlap between the 6 users.
- A friendship network was created using the friends overlap and we showed it in the form of a graph.
- In the graph, only the 6 user nodes are assigned with values of screennames for reducing clutter.

Data Visualization:

- There are numerous types of network creating tools and packages. We have used NetworkX which is a Python package used for creating, manipulating, and studying and analyzing networks. In short by using NetworkX for creating networks in python enhances python features and creates self-explaining network graphs. NetworkX has inbuilt functions which help us to calculate different network measures very easily.
- Below is the representation of a friendship network between the 6 users, their friends and their friends overlap.

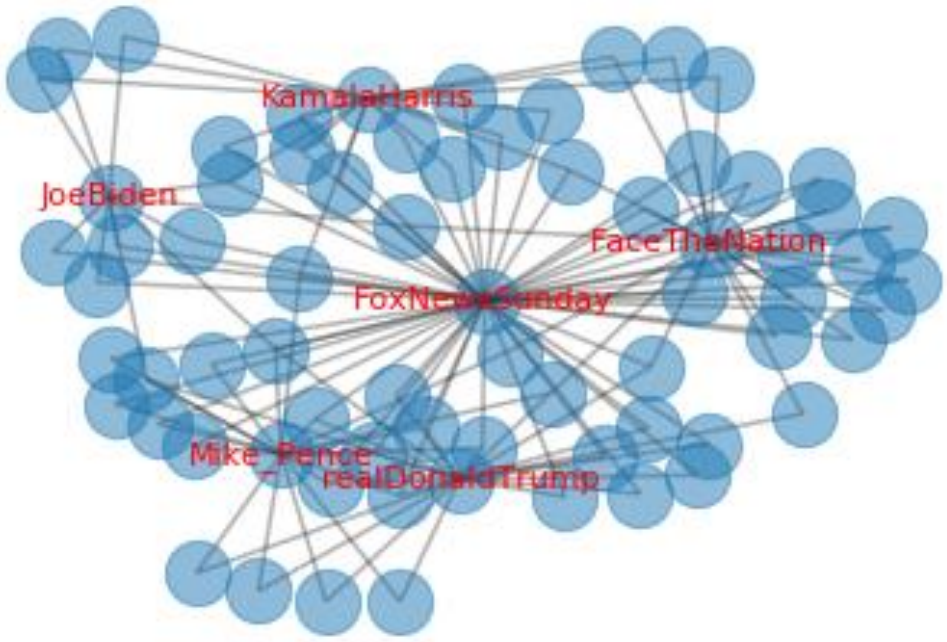


Fig1.Friendship network representation

- The 6 user nodes are highlighted with their screen names. We have only selected the nodes who have a friendship overlap with at least 2 users. For Visualizing this plot, we used nx.spring_layout(graph,scale = 5) inbuilt function of networkX. A spiral plot displays data along an Archimedean spiral. The main graph nodes (chapters) are placed in a spiral starting from its center.

Network measure Calculations:

Network centrality helps us a lot for performing online social network analysis. With help of this we can measure influential nodes in social network, Nodes which are more informative, important pages in web and nodes which prevent network from breaking up.

1. **Degree Distribution:**
   Degree of node is nothing but the number of edges it is connected to. Degree distribution histogram can be plotted using list of unique degrees of all nodes against number of nodes having same degrees.
   $P_{deg}(k)$=fraction of nodes in the graph with degree k.
   We have used 2etwork's to build degree distribution histogram which we save as png file.

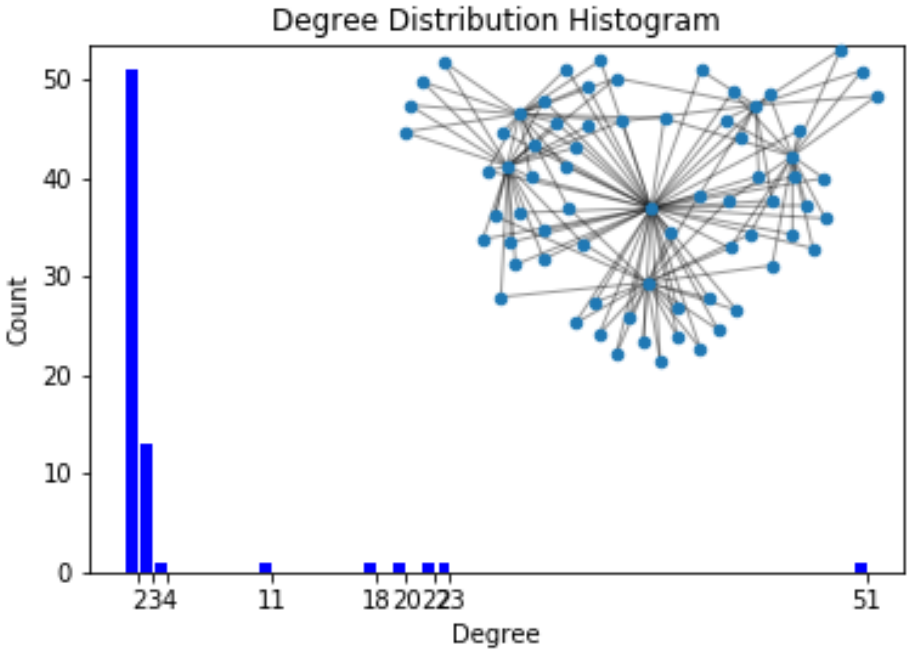


Fig2: Degree Distribution Histogram

2. **Degree Centrality:**

   Based on assumption that important nodes have many connection. In our friendship network important nodes are our six users.

   $$Centrality_{degree}(v) = d_v/(|N| - 1)$$

   Where, d_v → degree of node v

   N → set of all nodes of a graph

   We have used networkx's inbuilt function to calculate degree centrality which is **nx.degree_centrality(graph)**

3. **Closeness:**

   Assumption: Important nodes are close to other nodes

   this will return sum of path lengths from given nodes to all other nodes, and for nodes which are unreachable it is calculated using following formula,

   $$Centrality_{closeness}(v) = (|R(v)|/|N| - 1) * (|R(v)|/\Sigma_{u \epsilon R(v)} d(v, u))$$

   R(v) is the set of all nodes v can reach

   We have used networkx's inbuilt function to calculate closeness which is **nx.closeness_centrality(G)**

4. **Betweenness:**

   Assumption: Important nodes are close to other nodes

   It is a measure of sum of fraction of all shortest paths going through node v. can be calculated by the following formula,

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

Where , v→ set of nodes,

$\sigma(s,t)$ → number of shortest paths from s to t

$\sigma(s,t|v)$ → number of shortest paths from s to v passing from v

We have used networkx's inbuilt function to calculate Betweenness which is

nx.betweenness_centrality(graph, normalized = True, endpoints = False)

Where,

normalized= True → between ness values will get normalized by 2/((n-1)(n-2)) where n is number of nodes in graph.

Endpoints= False → will exclude endpoint from a shortest path

5. **Eigenvector_centrality:**
   It will help to decide importance of a node based on its connectivity with other nodes, if it is connected to more number of other important nodes then that node is considered as important.
   We have used networkx's inbuilt function to calculate eigenvector centrality which is
   nx.eigrnvector_centrality(graph)

6. **Eccentricity:** For a node n in a graph G, the eccentricity of n is the largest possible shortest path distance between n and all other nodes.
   Used function → **nx.eccentricity(graph)**

7. **Diameter:** The maximum shortest distance between a pair of nodes in a graph G is its Diamater. It is the largest possible eccentricity value of a node.
   Used function → **nx.diameter(graph)**

8. **Radius**: It is the minimum eccentricity value of a node.
   Used function → **nx.radius(graph)**

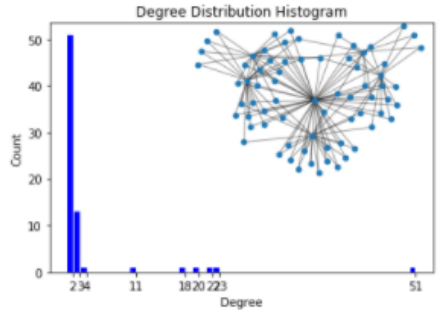
9. **Preiphery**: It is the set of nodes that have their eccentricity equal to their Diameter.
   Used function → **list(nx.periphery(G))**

10. **Center**: Center of a Graph is the set of nodes whose eccentricity is equal to the radius of the Graph.
    Used function → **list(nx.center(G))**

Results and code snippets for Network measurement are as follows:

In [16]:

```
#Degree Distribution Histogram

degree_sequence=sorted([d for n,d in graph.degree()], reverse=True) # degree sequence
#print "Degree sequence", degree_sequence
degreeCount=collections.Counter(degree_sequence)
deg, cnt = zip(*degreeCount.items())

fig, ax = plt.subplots()
plt.bar(deg, cnt, width=0.80, color='b')

plt.title("Degree Distribution Histogram")
plt.ylabel("Count")
plt.xlabel("Degree")
ax.set_xticks([d+0.4 for d in deg])
ax.set_xticklabels(deg)

#graph in inset
plt.axes([0.4, 0.4, 0.5, 0.5])
pos=nx.spring_layout(graph)
plt.axis('off')
nx.draw_networkx_nodes(graph, pos, node_size=20)
nx.draw_networkx_edges(graph, pos, alpha=0.4)

plt.savefig("degree__distribution_histogram.png")
plt.show()
```




```
#Network Measures for twitter friendship network
deg_centrality = nx.degree_centrality(graph)
print("Degree Centrality",deg_centrality)
print("\n")
eigenvector_centrality=nx.eigenvector_centrality(graph)
print("\n")
print("Eigenvector Centrality",eigenvector_centrality)
print("\n")
bet_centrality = nx.betweenness_centrality(graph, normalized = True,  endpoints = False)
print("Betweeness",bet_centrality)
print("\n")
close_centrality = nx.closeness_centrality(graph)
print("Closeness",close_centrality)
```

Degree Cntrality {'FaceTheNation': 0.32857142857142857, 'FoxNewsSunday': 0.7285714285714285, 'JoeBiden': 0.15714285714285714, 'KamalaHarris': 0.2571428571428571, 'Mike_Pence': 0.285714285714285, 'realDonaldTrump': 0.3142857142857143, 81891097567344128: 0.05714285714285714, 939091: 0.04285714285714286, 18166778: 0.04285714285714286, 30354991: 0.04285714285714286, 81892713188335616l: 0.04285714285714286, 17685258: 0.04285714285714286, 39349894: 0.04285714285714286, 172858784: 0.04285714285714286, 620571475: 0.04285714285714286, 2353605901: 0.04285714285714286, 729676086632656900: 0.04285714285714286, 822215679726100480: 0.04285714285714286, 823367015830323201: 0.04285714285714286, 822215673812119553: 0.04285714285714286, 837008 2: 0.02857142857142857, 15324851: 0.02857142857142857, 17539497: 0.02857142857142857, 17906632: 0.02857142857142857, 19455864: 0.02857142857142857, 85606078: 0.02857142857142857, 86697288: 0.02857142857142857, 87048122: 0.02857142857142857, 150078976: 0.02857142857142857, 208120290: 0.02857142857142857, 217543151: 0.02857142857142857, 394351710: 0.02857142857142857, 963480595: 0.02857142857142857, 1330457336: 0.02857142857142857, 1967216306: 0.02857142857142857, 2167097881: 0.02857142857142857, 799764016885329921: 0.02857142857142857, 820452522494226433: 0.02857142857142857, 830177829736345600: 0.02857142857142857, 11134252: 0.02857142857142857, 15007149: 0.02857142857142857, 15985455: 0.02857142857142857, 17494010: 0.02857142857142857, 19471123: 0.02857142857142857, 22203756: 0.02857142857142857, 22703645: 0.02857142857142857, 25101996: 0.02857142857142857, 44196397: 0.02857142857142857, 125453969: 0.02857142857142857, 325830217: 0.02857142857142857, 409486555: 0.02857142857142857, 471672239: 0.02857142857142857, 609476852: 0.02857142857142857, 778798705: 0.02857142857142857, 1013672084: 0.02857142857142857, 1084375028: 0.02857142857142857, 1249982359: 0.02857142857142857, 1339835893: 0.02857142857142857, 1615463502: 0.02857142857142857, 2359926157: 0.02857142857142857, 4121225056: 0.02857142857142857, 818876014390603776: 0.02857142857142857, 822127086194348032: 0.02857142857142857, 964032914626359296: 0.02857142857142857, 260759782: 0.02857142857142857, 821784477076750338: 0.02857142857142857, 1129095209772552192: 0.02857142857142857, 39344374: 0.02857142857142857, 52544275: 0.02857142857142857, 75541946: 0.02857142857142857, 108471631: 0.02857142857142857}

Eigenvector Centrality {'FaceTheNation': 0.22704686724674367, 'FoxNewsSunday': 0.56425612096251, 'JoeBiden': 0.09101910116551435, 'KamalaHarris': 0.14940792226638136, 'Mike_Pence': 0.209987508983464, 'realDonaldTrump': 0.2353007237631662, 818910970567344128: 0.12859886634605114, 939091: 0.10992038315598103, 18166778: 0.11995680300117999, 30354991: 0.10309775410374517, 818927131883356161: 0.07856115090938313, 17685258: 0.11796344757588598, 39349894: 0.11796344757588598, 172858784: 0.09402578259260762, 620571475: 0.11796344757588598, 2353605901: 0.11796344757588598, 729676086632656900: 0.11796344757588598, 822215679726100480: 0.10792702773068702, 823367015830323201: 0.11796344757588598, 822215673812119553: 0.06266655034600974, 837008 2: 0.09246233533357999, 15324851: 0.04398806715593963, 17539497: 0.09246233533357999, 17906632: 0.09246233533357999, 19455864: 0.09246233533357999, 85606078: 0.09246233533357999, 86697288: 0.04398806715593963, 87048122: 0.09246233533357999, 150078976: 0.04398806715593963, 208120290: 0.05106668324178314, 217543151: 0.09246233533357999, 394351710: 0.09246233533357999, 963480595: 0.05402448700113857, 1330457336: 0.09246233533357999, 1967216306: 0.09246233533357999, 2167097881: 0.09246233533357999, 799764016885329921: 0.09246233533357999, 820452522494226433: 0.09246233533357999, 830177829736345600: 0.09246233533357999, 11134252: 0.09046897990828598, 15007149: 0.08339036382244246, 15985455: 0.09046897990828598, 17494010: 0.08339036382244246, 19471123: 0.08339036382244246, 22203756: 0.09342678366764141, 22703645: 0.09342678366764141, 25101996: 0.09342678366764141, 44196397: 0.08339036382244246, 125453969: 0.08339036382244246, 325830217: 0.07656773477020658, 409486555: 0.07656773477020658, 471672239: 0.09342678366764141, 609476852: 0.07656773477020658, 778798705: 0.08339036382244246, 1013672084: 0.09342678366764141, 1084375028: 0.09046897990828598, 1249982359: 0.09342678366764141, 1339835893: 0.07656773477020658, 1615463502: 0.08339036382244246, 2359926157: 0.08339036382244246, 4121225056: 0.09342678366764141, 818876014390603776: 0.09046897990828598, 822127086194348032: 0.09046897990828598, 964032914626359296: 0.08339036382244246, 260759782: 0.02809346659256622, 821784477076750338: 0.02809346659256622, 1129095209772552192: 0.02809346659256622, 39344374: 0.05203113157584456, 52544275: 0.05203113157584456, 75541946: 0.05203113157584456, 108471631: 0.05203113157584456}

Betweeness {'FaceTheNation': 0.12847098772240031, 'FoxNewsSunday': 0.6409823425081171, 'JoeBiden': 0.04854902022361914, 'KamalaHarris': 0.09767292359720813, 'Mike_Pence': 0.10056827750806763, 'realDonaldTrump': 0.1137771523743352, 818910970567344128: 0.01972812664207881, 939091: 0.008946837099567 62, 18166778: 0.009412331726798825, 30354991: 0.010415974392495734, 818927131883356161: 0.008647863350279575, 17685258: 0.006774127850007667, 39349894: 0.006774127850007667, 172858784: 0.009429131695536011, 620571475: 0.006774127850007667, 2353605901: 0.006774127850007667, 729676086632656900: 0.006774127850007667, 822215679726100480: 0.015208896143075133, 823367015830323201: 0.006774127850007667, 822215673812119553: 0.009262994786123682, 8370082: 0.0022096638500532055, 15324851: 0.0028012149563024754, 17539497: 0.0022096638500532055, 17906632: 0.0022096638500532055, 19455864: 0.0022096638500532055, 85606078: 0.0022096638500532055, 86697288: 0.0028012149563024754, 87048122: 0.0022096638500532055, 150078976: 0.0028012149563024754, 208120290: 0.004136326040927484, 217543151: 0.0022096638500532055, 394351710: 0.0022096638500532055, 963480595: 0.004094071436306379, 1330457336: 0.0022096638500532055, 1967216306: 0.0022096638500532055, 2167097881: 0.0022096638500532055, 799764016885329921: 0.0022096638500532055, 820452522494226433: 0.0022096638500532055, 830177829736345600: 0.0022096638500532055, 11134252: 0.0032480655365227083, 15007149: 0.0039359582932119 34, 15985455: 0.0032480655365227083, 17494010: 0.0039359582932119 34, 19471123: 0.0039359582932119 34, 22203756: 0.0031085964404392427, 22703645: 0.0031085964404392427, 25101996: 0.0031085964404392427, 44196397: 0.0039359582932119 34, 125453969: 0.0039359582932119 34, 325830217: 0.0041084696878993169, 409486555: 0.0041084696878993169, 471672239: 0.0031085964404392427, 609476852: 0.0041084696878993169, 778798705: 0.0039359582932119 34, 1013672084: 0.0031085964404392427, 1084375028: 0.0032480655365227083, 1249982359: 0.0031085964404392427, 1339835893: 0.0041084696878993169, 1615463502: 0.0039359582932119 34, 2359926157: 0.0039359582932119 34, 4121225056: 0.0031085964404392427, 818876014390603776: 0.0032480655365227083, 822127086194348032: 0.0032480655365227083, 964032914626359296: 0.0039359582932119 34, 260759782: 0.0013847035233309068, 821784477076750338: 0.0013847035233309068, 1129095209772552192: 0.0013847035233309068, 39344374: 0.0004174658730457148, 52544275: 0.0004174658730457148, 75541946: 0.0004174658730457148, 108471631: 0.0004174658730457148}

```
Closeness {'FaceTheNation': 0.44025157232704404, 'FoxNewsSunday': 0.6796116504854369, 'JoeBiden': 0.3825136612021858, 'Kamala
laHarris': 0.4093567251461988, 'Mike_Pence': 0.42424242424242425, 'realDonaldTrump': 0.4294478527607362, 81891097056734412
8: 0.4861111111111111, 939091: 0.4666666666666667, 18166778: 0.4794520547945205, 30354991: 0.47297297297297297, 81892713188
3356161: 0.40229885057471265, 17685258: 0.4605263157894737, 39349894: 0.4605263157894737, 172858784: 0.45454545454545453, 6
20571475: 0.4605263157894737, 2353605901: 0.4605263157894737, 729676086632656900: 0.4605263157894737, 822215679726100480:
0.49295774647887325, 823367015830323201: 0.4605263157894737, 822215673812119553: 0.36082474226804123, 8370082: 0.4430379746
835443, 15324851: 0.35353535353535354, 17539497: 0.4430379746835443, 17906632: 0.4430379746835443, 19455864: 0.443037974683
5443, 85606078: 0.4430379746835443, 86697288: 0.35353535353535354, 87048122: 0.4430379746835443, 150078976: 0.3535353535353
5354, 208120290: 0.3684210526315789, 217543151: 0.4430379746835443, 394351710: 0.4430379746835443, 963480595: 0.37234042553
19149, 1330457336: 0.4430379746835443, 1967216306: 0.4430379746835443, 2167097881: 0.4430379746835443, 799764016885329921:
0.4430379746835443, 820452522494226433: 0.4430379746835443, 8301778297363456000: 0.4430379746835443, 11134252: 0.44871794871
794873, 15007149: 0.4430379746835443, 15985455: 0.44871794871794873, 17494010: 0.4430379746835443, 19471123: 0.443037974683
5443, 22203756: 0.44871794871794873, 22703645: 0.44871794871794873, 25101996: 0.44871794871794873, 44196397: 0.443037974683
5443, 125453969: 0.4430379746835443, 325830217: 0.43209876543209874, 409486555: 0.43209876543209874, 471672239: 0.448717948
71794873, 609476852: 0.43209876543209874, 778798705: 0.4430379746835443, 1013672084: 0.44871794871794873, 1084375028: 0.448
71794871794873, 1249982359: 0.44871794871794873, 1339835893: 0.43209876543209874, 1615463502: 0.4430379746835443, 235992615
7: 0.4430379746835443, 4121225056: 0.44871794871794873, 81887601439060377 6: 0.44871794871794873, 822127086194348032: 0.4487
1794871794873, 964032914626359296: 0.4430379746835443, 260759782: 0.3153153153153153, 821784477076750338: 0.315315315315315
3, 1129095209772552192: 0.3153153153153153, 39344374: 0.32710280373831774, 52544275: 0.32710280373831774, 75541946: 0.32710
280373831774, 108471631: 0.32710280373831774}
```

In [18]:

```
#Calculating graph properties
print("Eccentricity: ", nx.eccentricity(graph))
print("\n")
print("Diameter: ", nx.diameter(graph))
print("\n")
print("Radius: ", nx.radius(graph))
print("\n")
print("Preiphery: ", list(nx.periphery(graph)))
print("\n")
print("Center: ", list(nx.center(graph)))
print("\n")
```

```
Eccentricity:  {'FaceTheNation': 3, 'FoxNewsSunday': 3, 'JoeBiden': 3, 'KamalaHarris': 4, 'Mike_Pence': 3, 'realDonaldTrum
p': 4, 818910970567344128: 4, 939091: 4, 18166778: 4, 30354991: 4, 818927131883356161: 4, 17685258: 4, 39349894: 4, 1728587
84: 4, 620571475: 4, 2353605901: 4, 729676086632656900: 4, 822215679726100480: 4, 823367015830323201: 4, 82221567381211955
3: 4, 8370082: 4, 15324851: 4, 17539497: 4, 17906632: 4, 19455864: 4, 85606078: 4, 86697288: 4, 87048122: 4, 150078976: 4,
208120290: 4, 217543151: 4, 394351710: 4, 963480595: 4, 1330457336: 4, 1967216306: 4, 2167097881: 4, 799764016885329921: 4,
820452522494226433: 4, 8301778297363456000: 4, 11134252: 4, 15007149: 4, 15985455: 4, 17494010: 4, 19471123: 4, 22203756: 4,
22703645: 4, 25101996: 4, 44196397: 4, 125453969: 4, 325830217: 4, 409486555: 4, 471672239: 4, 609476852: 4, 778798705: 4,
1013672084: 4, 1084375028: 4, 1249982359: 4, 1339835893: 4, 1615463502: 4, 2359926157: 4, 4121225056: 4, 818876014390603776:
4, 822127086194348032: 4, 964032914626359296: 4, 260759782: 4, 821784477076750338: 4, 1129095209772552192: 4, 39344374:
4, 52544275: 4, 75541946: 4, 108471631: 4}


Diameter:  4


Radius:  3


Preiphery:  ['KamalaHarris', 'realDonaldTrump', 818910970567344128, 939091, 18166778, 30354991, 818927131883356161, 1768525
8, 39349894, 172858784, 620571475, 2353605901, 729676086632656900, 822215679726100480, 823367015830323201, 8222156738121195
53, 8370082, 15324851, 17539497, 17906632, 19455864, 85606078, 86697288, 87048122, 150078976, 208120290, 217543151, 3943517
10, 963480595, 1330457336, 1967216306, 2167097881, 799764016885329921, 820452522494226433, 830177829736345600, 11134252, 15
007149, 15985455, 17494010, 19471123, 22203756, 22703645, 25101996, 44196397, 125453969, 325830217, 409486555, 471672239, 6
09476852, 778798705, 1013672084, 1084375028, 1249982359, 1339835893, 1615463502, 2359926157, 4121225056, 818876014390603777
6, 822127086194348032, 964032914626359296, 260759782, 821784477076750338, 1129095209772552192, 39344374, 52544275, 7554194
6, 108471631]


Center:  ['FaceTheNation', 'FoxNewsSunday', 'JoeBiden', 'Mike_Pence']
```

Analysis and conclusion:

We calculated total number of shortest paths from each user to remaining users using **nx.shortest_path(graph).**

```
#all possible Shortest paths between our users
ls = (list(graph.nodes)[:6])
length = len(ls)
print("all possible Shortest paths between our users")
print("\n")
for i in range(0,length):
    for j in range(0,length):
        print(nx.shortest_path(graph,ls[i],ls[j]))
```

```
all possible Shortest paths between our users


['FaceTheNation']
['FaceTheNation', 939091, 'FoxNewsSunday']
['FaceTheNation', 30354991, 'JoeBiden']
['FaceTheNation', 939091, 'KamalaHarris']
['FaceTheNation', 818927131883356161, 'Mike_Pence']
['FaceTheNation', 18166778, 'realDonaldTrump']
['FoxNewsSunday', 939091, 'FaceTheNation']
['FoxNewsSunday']
['FoxNewsSunday', 818910970567344128, 'JoeBiden']
['FoxNewsSunday', 939091, 'KamalaHarris']
['FoxNewsSunday', 818910970567344128, 'Mike_Pence']
['FoxNewsSunday', 818910970567344128, 'realDonaldTrump']
['JoeBiden', 30354991, 'FaceTheNation']
['JoeBiden', 818910970567344128, 'FoxNewsSunday']
['JoeBiden']
['JoeBiden', 172858784, 'KamalaHarris']
['JoeBiden', 818910970567344128, 'Mike_Pence']
['JoeBiden', 818910970567344128, 'realDonaldTrump']
['KamalaHarris', 939091, 'FaceTheNation']
['KamalaHarris', 939091, 'FoxNewsSunday']
['KamalaHarris', 172858784, 'JoeBiden']
['KamalaHarris']
['KamalaHarris', 822215679726100480, 'Mike_Pence']
['KamalaHarris', 939091, 'FaceTheNation', 18166778, 'realDonaldTrump']
['Mike_Pence', 818927131883356161, 'FaceTheNation']
['Mike_Pence', 818910970567344128, 'FoxNewsSunday']
['Mike_Pence', 818910970567344128, 'JoeBiden']
['Mike_Pence', 822215679726100480, 'KamalaHarris']
['Mike_Pence']
['Mike_Pence', 818910970567344128, 'realDonaldTrump']
['realDonaldTrump', 18166778, 'FaceTheNation']
['realDonaldTrump', 818910970567344128, 'FoxNewsSunday']
['realDonaldTrump', 818910970567344128, 'JoeBiden']
['realDonaldTrump', 18166778, 'FaceTheNation', 939091, 'KamalaHarris']
['realDonaldTrump', 818910970567344128, 'Mike_Pence']
['realDonaldTrump']
```

From above results and our visualization using spral layout we came up with conclusion that people truly interested in politics and following all 4 candidates on twitter tend to follow foxNewsSunday on twitter. But people who are following only 2 candiatdates that is Donald trump and Kamala Harris (can say interested partially or reading trending tweets) tend to follow FaceTheNation. To support this we came up with explanation as, may be foxNewsSunday tweets more related to politics which is liked by people following all 4 candidates as it is election period. But compared to foxNewsSunday, FaceTheNation's tweets contents are not being liked by the followers of most of our users.