# Barack Obama Retweet Network

Mateusz Zaremba

March 9, 2020

## 1 Introduction

In this report I will present a dataset obtained from the from http://networkrepository.com/rt-barackobama.php website; it is a retweet network of the US president Barack Obama. The network was downloaded in a form of an edge list.

I wanted to see if by performing a set of tests on the network, it would be possible to identify Barack Obama's vertex. I believe I succeded to identify the aformnetioned vertex with a considerable degree of confidence. I will present my reasoning and methods which led to me to such conclusion.

## 2 Methods

The network was analysed using R and Python programming languages, and Gephi software package. The data statistics were calculated using Python; the data preparation and manipulation was done using R; the data visualisation was done using R and Gephi.

### 2.1 Data Preparation

The data was prepared using R programming language and RStudio software; the data was downloaded, unziped and saved into a R table (within the code). *Timestamp* column was dropped (it would be useful if the dataset was an updatable one; then it could be used to update the set only from a certain timestamp but in this case the set was a complete one).

By default, R names the columns like this:

$$x = \{V1, V2, ..., VN \mid N \in \mathbb{N}\}$$

To make the data manipulation easier the columns were renamed such that:

| Previous column name | Changed column name |
|---|---|
| V1 | from |
| V2 | to |

Later, the data was saved in a `.csv` format so it could be easly loaded into memory using Python programming language.

## 2.2 Metadata

The following metadata (which can be found on the http://networkrepository.com/rt-barackobama.php website) were used in the network analysis:

**Metadata**

| | |
|---|---|
| **Category** | Sparse Network |
| **Collection** | Retweet network |
| **Tags** | • Twitter hashtag network • retweet network |
| **Short** | Twitter retweet network |
| **Vertex type** | User |
| **Edge type** | Retweet |
| **Edge weights** | Unweighted |
| **Description** | Nodes are twitter users and edges are retweets. These were collected from various social and political hashtags. |

The dataset creators stated that the data was collected from "various social and political hashtags". It is not clear what methods they used but I suspect that once a user (which later will become a vertex in the network) which retweeted Barack Obama's tweet, a retweet link (the edge in the network) was created. Later, in the dataset visualisation, it is clearly visible that some people retweeted the retweets of someone who retweeted Barack Obama's tweet. This would create a path in the network to Barack Obama. The creators did not state if the network was simple or not but the fact the the network is sparse tells us that the creators took the steps to improve the network's efficiency in loading into memory and computation time. The sparcity statistic was not given so it is not certain to what degree the network's connection weights were reduced to 0 or very close to 0. The creators did not also specifically state if the network is simple or not but their efforts to make it as efficient as possible lead me to assumption that the network is a simple one, meaning it contains no loops or multiple edges; the visual analysis as well as the size metrics confirm this assumption which I elaborate in the next section.

## 2.3 Network size metrics

The analysis conducted with Python relied on the correct data prepartion in R; it was crucial for further analysis for the statistics from the 'Network Repository' website, like *number of nodes*, *number of edges*, *average degree* etc. (Rossi and Ahmed, 2015) to match the ones I calculated. The statistics matched and the comparision of the reults is presented in the table below:

| | Calculated Statistics | Website Statistics |
|---|---|---|
| Number of nodes | 9631 | 9.6K |
| Number of edges | 9775 | 9.8K |
| Average degree | 2.0299 | 2 |

## 2.4 Network Structure Metrics

Our graph in undirected so it does not contain in- or out-degrees. Instead, we found the highest degree node - it had an id of 2506 and degree of 7655; this was likely Barack Obama's twitter account since it is the most connected.

## 2.5 Network Density

The network's density is equal to 0.000210789557302036, which is close to zero, meaning - our graph is close to being 'fully disconnected'.

## 2.6 Shortest Path Between Two Nodes

We will find the shortest path between top two nodes with the highest degree and between the nodes with the highest and the lowest degree.

### 2.6.1 Top Two

The top two nodes with the highest degree had ids: 2506 (Barack Obama) and 9302. Shortest path between these two nodes is: ['2506', '8474', '9302']; the length of this path is 2, which means they are not connected directly.

### 2.6.2 Highest and Lowest Degree

As already mentioned, the id of the node with the highest degree is: 2506 and the id of the node with the lowest degree is: 2709; the shortest path between these two nodes is: ['2506', '2709'] and its length is 1, meaning the nodes are connected directly.

## 2.7 Identifying Network Communities

We managed to identify 138 community groups:

```
Counter({1: 7414, 6: 611, 7: 158, 12: 114, 4: 106, 2: 102, 50: 84, 5: 69,
88: 66, 61: 61, 127: 53, 35: 49, 19: 39, 25: 38, 17: 27, 111: 27, 77: 26,
23: 25, 112: 25, 0: 22, 49: 22, 26: 20, 54: 15, 37: 14, 102: 14, 22: 13,
41: 12, 48: 12, 55: 12, 9: 11, 15: 10, 32: 10, 75: 10, 97: 10, 34: 9, 38:
9, 82: 9, 85: 9, 13: 8, 3: 7, 29: 7, 84: 7, 105: 7, 108: 7, 68: 6, 78: 6,
83: 6, 117: 6, 86: 5, 110: 5, 8: 4, 27: 4, 31: 4, 62: 4, 67: 4, 79: 4, 93:
4, 100: 4, 129: 4, 10: 3, 20: 3, 36: 3, 40: 3, 47: 3, 66: 3, 70: 3, 80: 3,
81: 3, 87: 3, 90: 3, 95: 3, 104: 3, 107: 3, 114: 3, 125: 3, 133: 3, 134:
3, 139: 3, 11: 2, 14: 2, 16: 2, 18: 2, 21: 2, 24: 2, 28: 2, 30: 2, 33: 2,
39: 2, 42: 2, 43: 2, 44: 2, 45: 2, 46: 2, 51: 2, 52: 2, 53: 2, 56: 2, 57:
2, 58: 2, 59: 2, 60: 2, 63: 2, 64: 2, 65: 2, 69: 2, 71: 2, 72: 2, 73: 2,
74: 2, 76: 2, 89: 2, 91: 2, 92: 2, 94: 2, 96: 2, 98: 2, 99: 2, 101: 2, 103:
2, 106: 2, 109: 2, 113: 2, 115: 2, 116: 2, 118: 2, 119: 2, 120: 2, 121:
2, 122: 2, 123: 2, 124: 2, 126: 2, 128: 2, 130: 2, 131: 2, 132: 2, 135: 2,
136: 2, 137: 2, 138: 2})
```

It was calculated that the sizes of the communities range from 2 to 7414. We can, again, assume that the biggest community is likely to be centred around Barack Obama.

## 2.8 Network Structure Connectivity

Investigation reveals that the Barack Obama network is fully connected, and it has no sub-components; this is not surprising because the edges are retweets, nodes are twitter users and the network consists only of users who retweeted Barack Obama's posts.

## 2.9 Network Hubs/Brokers

Betweenness and closeness centrality was successfully calculated (Although, it took almost 40 minutes to compute) and sorted from the highest to the lowest score for top 20 results; but a `(PowerIterationFailedConvergence(...), 'power iteration failed to converge within 100 iterations')` error kept occurring when calculating the nodes eigenvector centrality; replacing `nx.eigenvector_centrality` with `nx.eigenvector_centrality_numpy` solved the issue (Stack Overflow, 2019) and after the fix, the computation was almost instantaneous.

Barack Obama's node (id = 2506) had the highest score in closeness, betweenness and eigenvector centrality, which means:

- The highest closeness centrality score – it is the farthest away from all other nodes in the network or – it takes the most time to spread information sequentially from it to other nodes (Sciencedirect.com, 2019).
- The highest betweenness centrality score – it has the highest number of distinct paths that strictly contain it in-between (Sci.unich.it, 2019).
- The highest eigenvector centrality score – it is the most influential node in the network.

# 3 Results

The network is a retweet network of Barack Obama's tweets. It is not obvious that Barack Obama's **?** is in the network; the highest degree vertex could be someone who's retweeted Barack Obama a thousand of times. At the time, it could had been his officail **?** twitter account which used to retweet every tweet from his private **?** twitter account. This would mean that evey single person in the network has retweeted the POTUS account insted of Barack Obama's - **?** account gets thousands of retweets so it's possible. But given the fact that it is "Barack Obama's retweet network" I think it is more plausible that the highest degree vertex is Barack Obama's private twitter account.

## 3.1 Data Visualisation

It was assumed that the biggest community of 7414 would be Barack Obama's first-degree neighbours. This cannot be really well seen on the graph without any colouring or sizing:

But we can immediately see Barack Obama's node (The big red one, obstructed by a bunch of smaller, blue nodes) when we distinguish the node's degree using sizing and colouring:

The graph becomes even more readable when we use colouring to represent the node's score of closeness and the node's sizing to represent its measure of betweenness:

The same graph was plotted using `Large Graph Layout` function which slightly improved the readability of the graph:

Interesting results were achieved using `Gephi` (Gephi.org, 2019), were it is clearly visible that there is one central node in the network (Barack Obama); there is also a one bigger cluster which was marked with an orange frame and many smaller ones which were marked with a green frame:

The small cluster up close:

The same cluster showing the nodes id of 9302:

And for the final proof, here is a zoom of the same graph generated with Gephi and node labels on, where it is clearly visible that the most central node, has an id of 2506:

# 4 Conclusion

To my best knowledge, the raport presents complelling evidence for the conclusion that the node with an id of 2506 is Barack Obama's private twitter account.

find the shortest path between top two nodes with the highest degrees as well as the node with the highest and the lowest degree; identify 138 communities within the network and their range; calculate and analyse the network's closeness, betweenness and eigenvector centrality; conduct a comprehensive visual analysis.

The size of the network proved it quite difficult to work with; calculation of the node betweenness and closeness centrality took approx. 40 minutes. Plotting of the network in R took only a few moments but it was only due to the fact that the nodes' labels were not being displayed, otherwise, it was taking approx. 5-10 minutes to plot. Nonetheless, an attempt was made to speed up the process using `with_lgl(...)` (Rdrr.io, 2019) function for `Large Graph Layout` (See Appendix B) but no improvement in rendering speed was noticed apart from a slightly better visual representation of the communities.

One of the things that could be done in the future, to improve calculation and render times, when analysing as big or even bigger networks, could be a usage of a GPU for network analysis. (Mathworks.com, 2019); e.g. Geforce GTS 250 has 450 cores working in parallel versus 4 on the CPU (Kajan and Slačka, 2019).

Overall, the data was well described and of a good quality; the original analysis provided a lot of good insight (Look table 1) and the size of the data was appropriate for a comprehensive analysis.

# 5 References


Gephi.org. (2019). Gephi - The Open Graph Viz Platform. [online] Available at: https://gephi.org/ [Accessed 16 Dec. 2019].

Kajan, S. and Slačka, J. (2019). COMPUTING OF NEURAL NETWORK ON GRAPHICS CARD. BSc. Slovak University of Technology in Bratislava.

Mathworks.com. (2019). Neural Networks with Parallel and GPU Computing-MATLAB & Simulink. [online] Available at: https://www.mathworks.com/help/deeplearning/ug/neural-networks-with-parallel-and-gpu-computing.html;jsessionid=fad53e284f66e53b4d2665c85a87 [Accessed 16 Dec. 2019].

Rdrr.io. (2019). layout_with_lgl: Large Graph Layout in igraph: Network Analysis and Visualization. [online] Available at: https://rdrr.io/cran/igraph/man/layout_with_lgl.html [Accessed 16 Dec. 2019].

Rossi, R. and Ahmed, N. (2015). The Network Data Repository with Interactive Graph Analytics and Visualization. [online] Network Repository. Available at: http://networkrepository.com/rt-barackobama.php [Accessed 16 Dec. 2019].

Sci.unich.it. (2019). Betweenness Centrality. [online] Available at: https://www.sci.unich.it/~francesc/teaching/network/betweeness.html [Accessed 16 Dec. 2019].

Sciencedirect.com. (2019). Closeness Centrality - an overview | ScienceDirect Topics. [online] Available at: https://www.sciencedirect.com/topics/computer-science/closeness-centrality [Accessed 16 Dec. 2019].

Stack Overflow. (2019). Using networkx to calculate eigenvector centrality. [online] Available at: https://stackoverflow.com/questions/43208737/using-networkx-to-calculate-eigenvector-centrality [Accessed 16 Dec. 2019].

Sadler, J. (2019). Introduction to Network Analysis with R. [online] Jesse Sadler. Available at: https://www.jessesadler.com/post/network-analysis-with-r/ [Accessed 16 Dec. 2019].

# 6 Appendix A

Python code

```
import networkx as nx
import csv
import matplotlib.pyplot as plt
from operator import itemgetter
from matplotlib.pyplot import figure

BO_graph = nx.Graph()
with open('barack_obama.csv','r') as BO:
  columns = next(BO, None)# skip the header row
  BO_graph = nx.parse_edgelist(BO, delimiter=',', create_using=nx.Graph(),
    nodetype=str, data=(('type',str),('weight', float),('book',int)))

# 1. Network size metrics
# This will print the network's:
# - Number of edges
# - Number of nodes
# - Average degree
print(nx.info(BO_graph))

# 2. Network structure metrics
# degree('node id', node degree)
highest_degree = max(BO_graph.degree(), key=lambda x : x[1])
sorted_degree = sorted(BO_graph.degree(), key=lambda x : x[1], reverse=True)
print('Top 10 highest degrees:', sorted_degree[:10])
print('Highest degree:', highest_degree)

# 3. Network Density
density = nx.density(BO_graph)
print('Network density is:', density)

# 4. Shortest Path Between Two Nodes
# 4.1 Shortest Path Between Top Two Highest Degree Nodes
# Find ids of top 2 nodes with the highest degree
top_two_degree_path = nx.shortest_path(
  BO_graph, source=sorted_degree[0][0], target=sorted_degree[1][0])

print(f'Shortest path between node {sorted_degree[0][0]}\
  and node {sorted_degree[1][0]}:', top_two_degree_path)
print('Length of this path:', len(top_two_degree_path)-1)

# 4.2 Shortest Path Between Nodes With The Highest and The Loewst Degree
highest_lowest_degree_path = nx.shortest_path(
  BO_graph, source=sorted_degree[0][0], target=sorted_degree[-1][0])
```

```python
print(f'Shortest path between node with the highest degree of\
  {sorted_degree[0][1]} and the node with the lowest degree of\
  {sorted_degree[-1][1]}:',
  highest_lowest_degree_path)
print('Length of this path:', len(highest_lowest_degree_path)-1)

# 5. Improving the network visualisation
import community

from collections import Counter

parts = community.best_partition(BO_graph)
values = [parts.get(node) for node in BO_graph.nodes()]

print(Counter(values))
print(f"Sizes of the communities range from\
  {min(Counter(values).items(), key=lambda i : i[1])}")
print(f"to {max(Counter(values).items(), key=lambda i : i[1])}")

# 6.    Network Structure Connectivity
if nx.is_connected(BO_graph):
    print("Barack Obama network is fully connected and has no sub-networks")
else:
    print("Barack Obama network is NOT fully connected")

# # 7. Network Hubs/Brokers
# 7.1 Node betweenness
betweenness_dict = nx.betweenness_centrality(BO_graph)

# Assign each to an attribute in your network
nx.set_node_attributes(BO_graph, betweenness_dict, 'betweenness')

sorted_betweenness = sorted(betweenness_dict.items(),
  key=itemgetter(1),
  reverse=True)

print("Top 20 nodes by betweenness centrality:")
for i in sorted_betweenness[:20]:
    print(i)
# ('2506', 0.9894963970281675)
# ('9302', 0.12282820504741454)
# ('4143', 0.03144816600031404)
# ...

# # 7.2 Node closeness
closeness_dict = nx.closeness_centrality(BO_graph)
```

```python
# Assign each to an attribute in your network
nx.set_node_attributes(BO_graph, closeness_dict, 'closeness')

sorted_closeness = sorted(closeness_dict.items(),
  key=itemgetter(1),
  reverse=True)

print("Top 20 nodes by closeness centrality:")
for i in sorted_closeness[:20]:
    print(i)
# ('2506', 0.6994988014818043)
# ('6675', 0.4373297002724796)
# ('8744', 0.4362993838347227)
# ...

# # 7.3 Node eigenvector centrailty
# using: nx.eigenvector_centrality()
# error: PowerIterationFailedConvergence:
# (PowerIterationFailedConvergence(...),
# 'power iteration failed to converge within 100 iterations')
# use: nx.eigenvector_centrality_numpy()
eigenvector_dict = nx.eigenvector_centrality_numpy(BO_graph)

nx.set_node_attributes(BO_graph, eigenvector_dict, 'eigenvector')

sorted_eigenvector = sorted(eigenvector_dict.items(),
  key=itemgetter(1),
  reverse=True)

print("Top 20 nodes by eigenvector centrality:")
for b in sorted_eigenvector[:20]:
    print(b)
# Top 20 nodes by eigenvector centrality:
# ('2506', 0.7070870193199195)
# ('8983', 0.00849854648835321)
# ('8744', 0.008303338474194982)
# ...
```

# 7 Appendix B

R code

```
library(igraph)
library(tidyverse)
library(ggraph)
library(tidygraph)

# download
pth <- "http://nrvis.com/download/data/rt/rt_barackobama.zip"
download.file(pth, destfile = "rt_barackobama.zip")

# see file names
unzip("rt_barackobama.zip", list = TRUE)

# unzip
unz <- unzip("rt_barackobama.zip", "rt_barackobama.edges")

dat <- read.table(unz, sep=",")

# Drop 3rd column (V3) of the dataframe because it's only a timestamp
dat <- select(dat,-c(3))

# Rename
dat <- dat %>%
  rename(
    from = V1,
    to = V2
  )

# Save as csv to use in python
write.csv(dat,"barack_obama.csv", row.names = FALSE)

# Create graph
g <- graph_from_data_frame(dat, directed=FALSE)
summary(g)

# Ensure that that the plots take most of the available page space
par(oma=c(0,0,0,0),mar=c(0,0,0,0))

# Colour the nodes based on their degree
colors.new=rev(rainbow(max(degree(g))+1,end=2/3))
V(g)$color=colors.new[degree(g)+1]
plot.igraph(g,vertex.label=NA)

# Set vertex size according to degree of each vertex (min. size = 5)
V(g)$size=5+(15)/diff(range(degree(g)))*degree(g)
```

```r
# Plot the graph without node labels
plot.igraph(g,vertex.label=NA)

# Set nodes' colour palette for clossness centrality
# creates a color palette for us to use
colors.new=rev(rainbow(10,end=4/6))
# calculates closeness metric for all nodes
net.close=as.numeric(closeness(g))
# normalises the closeness value
net.close=floor((net.close-min(net.close))/
  diff(range(net.close))*(length(colors.new)-1)+1)
# sets the color of each node according to the closenss score
V(g)$color=colors.new[net.close]
# Plot the graph without node labels
plot.igraph(g,vertex.label=NA)

# Ser node size for betweenness centrality
# calculates betweenness of each node
net.between=as.numeric(betweenness(g))
# "normalises" the score
net.between=floor((net.between-min(net.between))/
  diff(range(net.between))*(length(colors.new)-1)+1)
# sets the node size accoring the betweenness score
V(g)$size=5+(20)/diff(range(net.between))*net.between
# Plot the graph without node labels
plot.igraph(g,vertex.label=NA)

# Layout for big graphs
area <- vcount(g)^2
layout_with_lgl(g, maxiter = 150, maxdelta = vcount(g),
  area = vcount(g)^2, coolexp = 1.5, repulserad = area *
  vcount(g), cellsize = sqrt(sqrt(area)), root = NULL)

# Plot g with lgl and without node labels
with_lgl(plot.igraph(g, vertex.label=NA))
```