Social Media Intelligence Project

Elisabeth Putri - 20306250

27/05/2022

By including this statement, we the authors of this work, verify that: • I hold a copy of this assignment that we can produce if the original is lost or damaged. • I hereby certify that no part of this assignment/product has been copied from any other student’s work or from any other source except where due acknowledgement is made in the assignment. • No part of this assignment/product has been written/produced for us by another person except where such collaboration has been authorised by the subject lecturer/tutor concerned. • I am aware that this work may be reproduced and submitted to plagiarism detection software programs for the purpose of detecting possible plagiarism (which may retain a copy on its database for future plagiarism checking). • I hereby certify that we have read and understand what the School of Computing and Mathematics defines as minor and substantial breaches of misconduct as outlined in the learning guide for this unit.

Before we start the coding, importing all the library.

library('igraph')

## Warning: package 'igraph' was built under R version 4.1.3

##   
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':  
##   
## decompose, spectrum

## The following object is masked from 'package:base':  
##   
## union

library('rtweet')

## Warning: package 'rtweet' was built under R version 4.1.3

library('tm')

## Warning: package 'tm' was built under R version 4.1.3

## Loading required package: NLP

library('dplyr')

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:igraph':  
##   
## as\_data\_frame, groups, union

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

1. Gathering the Network

em <- search\_tweets(  
 "#elonmusktwitter", n=25000, include\_rts = FALSE, retryonratelimit = TRUE)  
  
names(em)

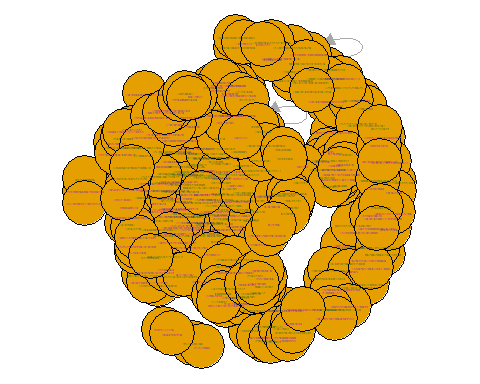
## [1] "user\_id" "status\_id"   
## [3] "created\_at" "screen\_name"   
## [5] "text" "source"   
## [7] "display\_text\_width" "reply\_to\_status\_id"   
## [9] "reply\_to\_user\_id" "reply\_to\_screen\_name"   
## [11] "is\_quote" "is\_retweet"   
## [13] "favorite\_count" "retweet\_count"   
## [15] "quote\_count" "reply\_count"   
## [17] "hashtags" "symbols"   
## [19] "urls\_url" "urls\_t.co"   
## [21] "urls\_expanded\_url" "media\_url"   
## [23] "media\_t.co" "media\_expanded\_url"   
## [25] "media\_type" "ext\_media\_url"   
## [27] "ext\_media\_t.co" "ext\_media\_expanded\_url"   
## [29] "ext\_media\_type" "mentions\_user\_id"   
## [31] "mentions\_screen\_name" "lang"   
## [33] "quoted\_status\_id" "quoted\_text"   
## [35] "quoted\_created\_at" "quoted\_source"   
## [37] "quoted\_favorite\_count" "quoted\_retweet\_count"   
## [39] "quoted\_user\_id" "quoted\_screen\_name"   
## [41] "quoted\_name" "quoted\_followers\_count"   
## [43] "quoted\_friends\_count" "quoted\_statuses\_count"   
## [45] "quoted\_location" "quoted\_description"   
## [47] "quoted\_verified" "retweet\_status\_id"   
## [49] "retweet\_text" "retweet\_created\_at"   
## [51] "retweet\_source" "retweet\_favorite\_count"   
## [53] "retweet\_retweet\_count" "retweet\_user\_id"   
## [55] "retweet\_screen\_name" "retweet\_name"   
## [57] "retweet\_followers\_count" "retweet\_friends\_count"   
## [59] "retweet\_statuses\_count" "retweet\_location"   
## [61] "retweet\_description" "retweet\_verified"   
## [63] "place\_url" "place\_name"   
## [65] "place\_full\_name" "place\_type"   
## [67] "country" "country\_code"   
## [69] "geo\_coords" "coords\_coords"   
## [71] "bbox\_coords" "status\_url"   
## [73] "name" "location"   
## [75] "description" "url"   
## [77] "protected" "followers\_count"   
## [79] "friends\_count" "listed\_count"   
## [81] "statuses\_count" "favourites\_count"   
## [83] "account\_created\_at" "verified"   
## [85] "profile\_url" "profile\_expanded\_url"   
## [87] "account\_lang" "profile\_banner\_url"   
## [89] "profile\_background\_url" "profile\_image\_url"

tweettext = as.data.frame(em$text, em$screen\_name)  
print(head(tweettext))

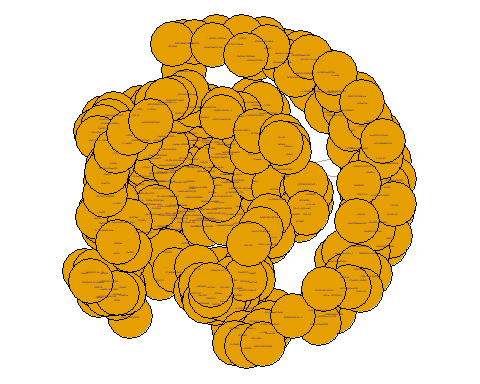
## em$text  
## CunningStunts19 goddammit I was promised an edit button. fuck all #elonmusktwitter  
## AChristhope Dear Social Media Users:\n\n#Twitter #instagram #Facebook #socialmediamanager #YouTube #YouTubers #socialjusticewarrior #politician #Leaders #followers #walkthetalk @elonmusk #elonmusktwitter https://t.co/rMynzvjzmx  
## Gaurav\_GK99 Am creating a illustrated photo of you and going to make it as nft . Will you buy it @elonmusk #ElonMusk #ElonMuskTwitter if you retweet or like this. I will give you the nft link. Guys please to make a change in my life. Am going to post this daily untill @elonmusk shares.  
## Gaurav\_GK99.1 Am creating a illustrated photo of you and going to make it as nft . Will you buy it @elonmusk #ElonMusk #ElonMuskTwitter if you retweet or like this. I will give you the nft link. Guys please to make a change in my life. Am going to post this daily untill @elonmusk shares.  
## Gaurav\_GK99.2 Am creating a illustrated photo of you and going to make it as nft . Will you buy it @elonmusk #ElonMusk #ElonMuskTwitter if you retweet or like this. I will give you the nft link. Guys please to make a change in my life. Am going to post this daily untill @elonmusk shares.  
## Gaurav\_GK99.3 Am creating a illustrated photo of you and going to make it as nft . Will you buy it @elonmusk #ElonMusk #ElonMuskTwitter if you retweet or like this. I will give you the nft link. Guys please to make a change in my life. Am going to post this daily untill @elonmusk shares.

1. Mention Graph

# Create graph on the mentions in each tweet  
datatw = network\_data(em, "mention")  
  
gnet <- graph\_from\_data\_frame(datatw, directed = TRUE)  
  
par(mar = c(0, 0, 0, 0))  
V(gnet)$label.cex=0.2  
plot(gnet, layout = layout.fruchterman.reingold, vertex.size = 30)



# Convert the directed graph into undirected graph  
convertem <- graph\_from\_data\_frame(datatw, directed = FALSE)  
  
par(mar = c(0, 0, 0, 0))  
V(convertem)$label.cex=0.1  
plot(convertem, layout = layout.fruchterman.reingold, vertex.size = 30)



# Compute the number of components and size of each components  
# number of components  
count\_components(convertem)

## [1] 30

# size of each components  
aa = components(convertem)  
aa$csize

## [1] 216 2 2 2 2 3 3 2 2 2 2 3 2 1 2 2 2 2 3  
## [20] 2 2 2 2 4 2 2 2 6 2 2

# Plot the largest component of the graph  
## First split the components of the graph  
deco = decompose(convertem)  
  
## nodes number in each component  
nc = sapply(deco, function(x) {length(V(x))})  
  
## index of largest component  
lc = which(nc == max(nc))  
  
## largest component's edges  
lce = deco[[lc]]  
  
## largest component graph  
lcgraph = cluster\_edge\_betweenness(lce, directed = FALSE)  
  
# check all partition  
print(lcgraph$membership)

## [1] 1 1 2 3 4 5 1 6 7 1 1 1 1 1 1 8 1 9 7 1 1 2 1 1 1  
## [26] 5 10 1 1 11 10 1 2 12 13 2 14 1 15 4 4 7 16 1 17 1 1 1 3 1  
## [51] 1 1 1 1 4 1 1 18 7 1 17 1 19 2 1 14 20 20 21 1 22 7 21 1 20  
## [76] 23 20 24 1 1 25 1 1 26 2 7 27 1 28 1 1 29 30 23 1 31 1 32 7 2  
## [101] 7 1 33 34 5 1 2 3 3 3 3 3 3 3 3 3 3 4 23 5 6 7 8 8 9  
## [126] 1 5 5 5 5 5 5 5 5 5 5 5 5 5 5 10 11 11 10 10 10 10 10 12 13  
## [151] 14 15 7 16 16 17 17 18 18 18 18 18 18 20 17 17 17 17 17 17 17 17 17 17 17  
## [176] 17 17 17 17 17 17 17 17 19 14 20 20 20 20 20 21 22 20 20 20 20 24 25 25 25  
## [201] 26 27 28 29 30 31 32 32 7 33 33 33 33 34 5 5

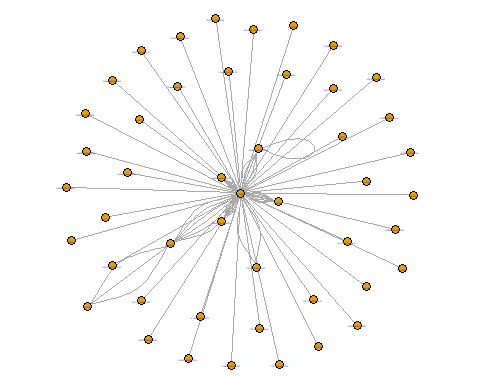
# check all the edges  
table(lcgraph$membership)

##   
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26   
## 47 8 12 5 20 2 11 3 2 8 3 2 2 4 2 3 23 7 2 14 3 2 3 2 4 2   
## 27 28 29 30 31 32 33 34   
## 2 2 2 2 2 3 5 2

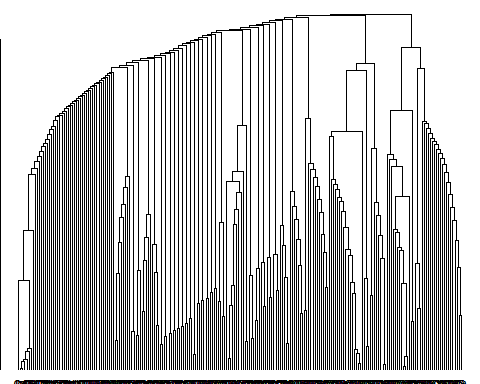
## By checking all the edges, the highest number is 1, so we use 1 as the largest component  
lcno = 1  
indexlc = which(lcgraph$membership == lcno)  
  
## Taking the nodes from largest component to be plotted  
lcgraphed = subgraph(deco[[lc]], indexlc)

## Warning in subgraph(deco[[lc]], indexlc): At  
## structural\_properties.c:2051 :igraph\_subgraph is deprecated from igraph 0.6, use  
## igraph\_induced\_subgraph instead

plot(lcgraphed, vertex.size=5)



## Graph partition by hierarchical relationship  
plot(as.dendrogram(lcgraph))



1. Graph Statistics

# graph diameter  
diameter(lcgraphed)

## [1] 3

# graph density  
graph.density(lcgraphed)

## [1] 0.08233117

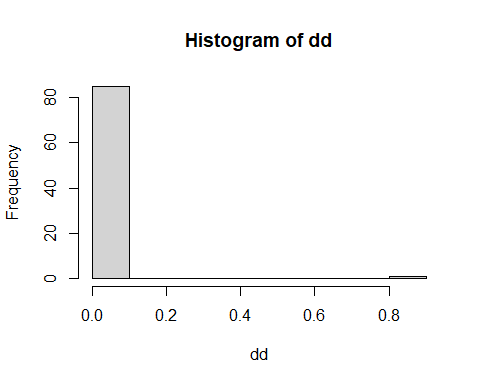
# plotting the degree distribution of the graph  
degree(lcgraphed)

## 1334440824608706562 1464529166557237253 710008378945355776 1523178461560418304   
## 1 18 1 2   
## 886822126145196033 158553324 1176088924894257152 1436568756331839492   
## 8 7 1 1   
## 3165824401 97297046 1331224968407891970 1316630499738021889   
## 1 1 1 1   
## 363762075 1519396939736829952 1483070768980316160 1475554039890952196   
## 9 1 1 1   
## 1455186931759910918 1484854743470309377 45152326 901313610806251520   
## 1 7 1 1   
## 1450820441803829256 837126781282852864 2977485983 908701589170352128   
## 1 1 1 1   
## 16222574 1480481160820109319 156198185 40628724   
## 1 1 1 1   
## 1488327858518999044 174451807 1522034108788281344 1445628693376684040   
## 1 1 1 1   
## 1408056463331758081 1512608806575898624 1512001189034217481 1521059182451036161   
## 1 1 1 1   
## 1531579300947759105 1256866059098955777 1345668916261965824 1241807750432206848   
## 1 1 1 1   
## 927032191233568768 1399302045413294085 2648989015 1511636208325169155   
## 1 1 1 1   
## 1021241294 44196397 1291945442   
## 1 85 3

max(degree(lcgraphed))

## [1] 85

dd = degree.distribution(lcgraphed)  
hist(dd)



# estimating the Power Law coefficient(c) from the degree distribution  
pl = fit\_power\_law(dd, xmin = 0.000000000000000001)  
print(pl)

## $continuous  
## [1] TRUE  
##   
## $alpha  
## [1] 1.026219  
##   
## $xmin  
## [1] 1e-18  
##   
## $logLik  
## [1] -10.68584  
##   
## $KS.stat  
## [1] 0.6268302  
##   
## $KS.p  
## [1] 0.003721941

print(pl)$alpha

## $continuous  
## [1] TRUE  
##   
## $alpha  
## [1] 1.026219  
##   
## $xmin  
## [1] 1e-18  
##   
## $logLik  
## [1] -10.68584  
##   
## $KS.stat  
## [1] 0.6268302  
##   
## $KS.p  
## [1] 0.003721941

## [1] 1.026219

4. Information Flow  
  
  
```r  
# neighourhood overlap between each pair of connected nodes in the twitter graph   
# changing the name of the vertex to easier analysis process  
bb = set.vertex.attribute(lcgraphed, "name", value = paste("A",1:146, sep = ""))

## Warning in vattrs[[name]][index] <- value: number of items to replace is not a  
## multiple of replacement length

en = ends(bb, E(bb), names = FALSE)  
  
ne.over = function(no1, no2, graphed) {  
 i = intersection(neighbors(graphed, no1), neighbors(graphed, no2))  
 u = union(neighbors(graphed, no1), neighbors(graphed, no2))  
 a = length(i)/ (length(u))   
 return(a)  
}  
  
nodes = list()  
  
for (i in seq(nrow(en))){  
 node1 = en[i, 1]  
 node2 = en[i, 2]  
 nover = ne.over(no1 = node1, no2 = node2, graphed = bb)  
 nodes[i] = nover  
}  
  
  
# identify the pair with the greatest and least neighborhood overlap  
d = c()  
  
for (i in 1:length(nodes)){  
 d[i] = nodes[[i]][1]  
}  
  
l = d[order(d,decreasing=TRUE)]  
#l

5. Account Popularity

# Measuring popularity of each account by Scaled PageRank

hu = rep(1, length(names(V(bb))))

names(hu) = names(V(bb))

au = hu

M = as\_adjacency\_matrix(bb)

M = as.matrix(M)

## iterate K times

ik = 100

for (i in 1:ik) {

au = t(M) %\*% hu

hu = M %\*% au

}

au = au/sum(au)

hu = hu/sum(hu)

# eigenvalue solution

ed1 = eigen(M %\*% t(M))

hub.ed = ed1$vectors[,1]

ed2 = eigen(t(M) %\*% M)

au.ed = ed2$vectors[,1]

hub.ed = hub.ed/sum(hub.ed)

au.ed = au.ed/sum(au.ed)

# Result checking

au - au.ed

hu - hub.ed

## PageRank

P = M %\*% diag(1/colSums(M))

nonodes = nrow(P)

R = matrix(1/nonodes, nonodes, nonodes)

ld = 0.8

S = ld\*P + (1-ld)\*R

colSums(S)

iter = rep(1/nonodes, nonodes)

for (k in 1:ik) {

iter = S %\*% iter

}

e.val = eigen(S)

pr = e.val$vector[,1]/sum(e.val$vector[,1])

pr = Re(pr)

#checking method

iter - pr

# Ten highest page rank

pro = sort(pr, index.return=TRUE)

pro$ix[1:10]

[1] 141 142 143 1 8 9 10 11 13 16

prorder = pr[order(pr,decreasing=TRUE)]

prorder[1:10]

[1] 0.308518527 0.039552618 0.026090744 0.020782265 0.017171617 0.013849264 0.012462664 0.010517661 0.010349100 0.009315947

V(lcgraphed)[141, 142, 143, 1, 8, 9, 10, 11, 13, 16]

+ 10/146 vertices, named, from 6efdbdc:

[1] 6182852 128372940 51827346 1513921541918138372 1316181534 1021241294 1450520650335072262

[8] 1452304054206488576 1510000127745970186 1423355927302975489

em %>% filter\_at(vars(user\_id, reply\_to\_user\_id), any\_vars(. %in% c("6182852", "128372940", "51827346", "1513921541918138372", "1316181534", "1021241294", "1450520650335072262", "1452304054206488576", "1510000127745970186", "1423355927302975489")))

```

6. Account Selection

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | Competitor | | |
| q1 | q2 | 1-q1-q2 |
| SpaceX | Tesla | Boring |
| Monitor | p1 | SpaceX | 1, 0.3 | 0.1, 3 | 0.3, 2 |
| p2 | Tesla | 0.2, 1 | 1, 0.2 | 0.3, 3 |
| 1- p1- p2 | Boring | 0.1, 3 | 0.1, 2 | 1, 0.1 |

Monitor

Competitor

From those equations, the probability distribution for each strategy is given below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | Competitor | | |
| 0.315 | 0.28 | 0.405 |
| SpaceX | Tesla | Boring |
| Monitor | 0.553 | SpaceX | 1, 0.3 | 0.1, 3 | 0.3, 2 |
| 0.298 | Tesla | 0.2, 1 | 1, 0.2 | 0.3, 3 |
| 0.149 | Boring | 0.1, 3 | 0.1, 2 | 1, 0.1 |