Assignment 2 - Analysis of US Political Blogs

Kalyani Prashant Kawale

Student ID: 21237189

Reading the Given Graph

```
# Importing packages
library(igraph)
library(knitr)
library(kableExtra)

# Reading graph from gml file
blogs_graph <- read_graph("polblogs.gml", format = "gml")

# Adding names for nodes using the label attribute
V(blogs_graph)$name <- V(blogs_graph)$label
# Calculating and setting the degree for each node
V(blogs_graph)$deg <- degree(blogs_graph, mode = "all")
# Calculating and setting the authority score for each node
V(blogs_graph)$authority <- authority_score(blogs_graph)$vector

# Initializing results table
results <- c()
```

Question 1: What are the sizes of the strongly connected component (scc), in-component, out-component and tube component?

Solution 1

Finding SCC components: The strongly connected component is computed using the **component()** method of igraph using *mode=strong*. Following code is used to extract the component with maximum number of nodes with the help of *csize* vector returned by component() method.

```
# Getting components from graph
strong_components <- components(blogs_graph, mode = "strong")
# Finding the nodes in largest cluster of connected components
scc <- which(strong_components$membership == which.max(strong_components$csize))
# Getting the scc nodes using id given by membership vector
scc_nodes <- V(blogs_graph)[V(blogs_graph)$id %in% scc]
# Adding SCC component size to table
results <- rbind(results, c("SCC", length(scc_nodes)))
cat("SCC Size:", length(scc_nodes))
```

```
## SCC Size: 793
```

```
# Finding the nodes not in scc
non_scc_nodes <- V(blogs_graph)[!V(blogs_graph) %in% scc_nodes]
```

Finding IN, OUT and TUBE Components: The IN and OUT component sizes are computed by checking if path exists between the nodes and scc component, while for the TUBE component, paths are checked between nodes not connected to scc and the IN/OUT components, using the **bfs()** method of igraph.

```
# Note: Following code is taken from "Exploratory Analysis of Game of Thrones Network"
# worksheet
# Function to detect paths between nodes
path_exists <- function(g, start_node, finish_nodes, mode){
  dist <- bfs(g, root=start_node, neimode=mode, unreachable=F, dist=T)$dist
  return (any(!is.nan(dist[finish_nodes])))
}
```

The following code defines a function to get components based on existence of paths between different components:

```
# Function to get components based on the links that exist or do not exist between
# two set of nodes
get_components <- function(start_nodes, end_nodes, mode1, mode2) {
  # start_nodes: set of nodes being examined to be in, out or tube
  # end_nodes: set of nodes based on whose links, resultant components are decided
  # mode1: 'in'/'out' to check if the paths exist
  # mode2: 'in'/'out' to check if paths do not exist
  result_components <- c()
  for (node in start_nodes) {
    # checking if node has paths according to mode1 with end_nodes
    if(path_exists(blogs_graph, node, end_nodes, mode = mode1)) {
      # checking if node doesn't have a path according to mode2 with end_nodes
      if(!path_exists(blogs_graph, node, end_nodes, mode = mode2)){
        # adding node component if path exists according to mode1,
        # and no path exists according to mode2
        result_components <- c(result_components, node)
      }
    }
  }
  return (result_components)
}
```

To extract the **IN component**, for each node in *non_scc_nodes*, checking if **path exists from node to scc component**, and **no path exists from scc component to the node**.

```
# Finding IN component
in_component <- get_components(non_scc_nodes, scc_nodes, 'out', 'in')
# Adding IN component size to table
results <- rbind(results, c("IN", length(in_component)))
cat("IN Component Size:", length(in_component))
```

```
## IN Component Size: 232
```

```
# Getting nodes not in SCC and IN component
rem_nodes <- non_scc_nodes[!non_scc_nodes %in% c(in_component)]
```

To extract the **OUT component**, for each node in *non_scc_nodes*, checking if **path exists to node from scc component**, and **no path exists from node to scc component**.

```
# Finding OUT component
out_component <- get_components(rem_nodes, scc_nodes, 'in', 'out')
# Adding OUT component size to table
results <- rbind(results, c("OUT", length(out_component)))
cat("OUT Component Size:", length(out_component))
```

```
## OUT Component Size: 165
```

```
# Getting nodes not in SCC, IN component, and OUT component
rem_nodes <- non_scc_nodes[!non_scc_nodes %in% c(in_component, out_component)]
```

To extract **TUBE component**, for each node in *rem_nodes* finding all nodes not connected to scc component with in-coming paths from IN component and out-going paths to OUT components using **path_exists()** method.

```
# Finding TUBE component size
# Getting nodes that are not connected to any nodes in SCC component
possible_tube <- c()
for (node in rem_nodes) {
  # checking if any incoming or outgoing paths exists between
  # remaining nodes and scc
  if(!path_exists(blogs_graph, node, scc_nodes, mode = 'out')
     && !path_exists(blogs_graph, node, scc_nodes, mode = 'in')) {
    # adding nodes not connected to scc in possible tube component
    possible_tube <- c(possible_tube, node)
  }
}
# Creating partial tube, by getting the nodes that have incoming edges
# from IN component, but do not have outgoing edges into IN component
partial_tube <- get_components(possible_tube, in_component, 'in', 'out')
# Creating the TUBE component, by getting the nodes that have outgoing edges
# into OUT component, but do not have incoming edges from OUT component
tube_component <- get_components(partial_tube, out_component, 'out', 'in')
# Adding TUBE component size to table
results <- rbind(results, c("TUBE", length(tube_component)))
cat("TUBE Component Size:", length(tube_component))
```

```
## TUBE Component Size: 0
```

Question 1 Results

```
# Printing SCC, IN, OUT, TUBE component sizes as a table
kable(results, col.names=c("Component", "Size"), caption = "Solution 1:",
      align = "c", booktabs=T) %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 1: Solution 1:

Component	Size
SCC	793
IN	232
OUT	165
TUBE	0

Question 2: What are the proportions of liberal/conservative blogs in the scc, in-component, out-component?

Solution 2

Following code is used to find the liberal and conservative nodes, which are then used to get their proportions in the SCC, IN component and OUT component.

```
# Getting nodes with value 0 i.e. liberal blogs
liberals <- V(blogs_graph)[V(blogs_graph)$value == 0]

# Getting nodes with value 1 i.e. conservative blogs
conservatives <- V(blogs_graph)[V(blogs_graph)$value == 1]
```

Extracting the amount of liberal/conservative nodes in each component using **intersect()** method.

```
# Initializing results for saving proportions
results <- c()
# Calculating the proportions and saving in results
l_in_scc <- round((length(intersect(liberals, scc_nodes))
                   / length(scc)) * 100, digits = 2)
results <- rbind(results, c("Proportion of liberal blogs in SCC",
                            paste(l_in_scc, "%")))

c_in_scc <- round((length(intersect(conservatives, scc_nodes))
                   / length(scc)) * 100, digits = 2)
results <- rbind(results, c("Proportion of conservative blogs in SCC",
                            paste(c_in_scc, "%")))

l_in_component <- round((length(intersect(liberals, in_component))
                         / length(in_component)) * 100, digits = 2)
results <- rbind(results, c("Proportion of liberal blogs in IN component",
                            paste(l_in_component, "%")))

c_in_component <- round((length(intersect(conservatives, in_component))
                         / length(in_component)) * 100, digits = 2)
results <- rbind(results, c("Proportion of conservative blogs in IN component",
                            paste(c_in_component, "%")))

l_out_component <- round((length(intersect(liberals, out_component))
                          / length(out_component)) * 100, digits = 2)
results <- rbind(results, c("Proportion of liberal blogs in OUT component",
```

```
                              paste(l_out_component, "%")))

c_out_component <- round((length(intersect(conservatives, out_component))
                          / length(out_component)) * 100, digits = 2)
results <- rbind(results, c("Proportion of conservative blogs in OUT component",
                              paste(c_out_component, "%")))
```

Question 2 Results

```
# Printing proportions of liberals/conservatives in SCC, IN, and OUT component
kable(results, caption = "Solution 2:", booktabs=T) %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 2: Solution 2:

Proportion of liberal blogs in SCC	44.26 %
Proportion of conservative blogs in SCC	55.74 %
Proportion of liberal blogs in IN component	58.62 %
Proportion of conservative blogs in IN component	41.38 %
Proportion of liberal blogs in OUT component	46.06 %
Proportion of conservative blogs in OUT component	53.94 %

Question 3: From the perspective of liberal bloggers, what are the 3 most influential conservative bloggers?

Solution 3

1. To find the influential conservative bloggers, from liberal perspective, the authority of each conservative blog (ego) is computed in a sub-graph, created using all the liberal nodes and conservative neighbors of the ego node, along with the ego node.

2. These authorities are compared and the 3 blogs with highest authority are extracted.

3. The sub-graph is created to get the influence of any conservative node only from the perspective of liberal nodes, however, the use of the 1.5 network is based on the assumption that many liberal blogs may be linking to a conservative blog, which highly references or is based on another conservative blog (immediate neighbor), which may be of higher importance and vice-versa.

```
# Defining function influential_blogs to calculate influential blogs
# from a community's perspective
influential_blogs <- function(src_community, target_community) {
  # src_community: community from whose perspective influential
  #                blogs are to be calculated
  # target_community: community from which the influential
  #                   blogs are to be fetched
  authorities <- list()
  for (node in target_community$name) {
```

```
    # getting neighbours of the target node using neighbors() igraph function
    neighbours <- neighbors(blogs_graph, node, mode="all")
    # adding the ego node to neighbours to create a 1.5 ego network
    ego_nodes1.5 <- c(node, neighbours$name)
    # creating sub-graph with all source nodes and ego network
    sub_net <- induced_subgraph(blogs_graph, c(src_community$name,ego_nodes1.5))
    # calculating the authorities of each node in the sub-graph
    V(sub_net)$authority <- authority_score(sub_net, scale = TRUE)$vector
    # getting the authority of the ego node in the sub-graph
    authorities[[node]] <- V(sub_net)[V(sub_net)$name == node]$authority
  }
  # Removing nodes with zero authority
  authorities <- authorities[authorities != 0]
  # Using order and unlist to sort the list from highest to lowest,
  # and returning the top 3 blogs
  return (authorities[order(unlist(authorities), decreasing=TRUE)][1:3])
}
# Saving 3 most influential conservative nodes from perspective of liberals
influential_c <- influential_blogs(liberals, conservatives)
```

Question 3 Results

```
# Initializing results to save influential conservative blogs
results <- c()
# Saving blog names, authority in sub-graph and overall authority in full graph
for(i in 1:length(influential_c)) {
  sub_auth <- round(as.double(influential_c[i]), digits = 4)
  overall_auth <- round(as.double(V(blogs_graph)[V(blogs_graph)$name ==
                          names(influential_c[i])]$authority), digits = 4)
  results <- rbind(results, c(names(influential_c[i]), sub_auth, overall_auth))

}
# Printing 3 most influential conservative nodes from perspective of liberals
kable(results, col.names = c("Blog", "Sub-Graph Authority", "Overall Authority"),
      caption = "Solution 3: Three most influential conservative bloggers,
      from the perspective of liberal bloggers, ", booktabs=T, align="c") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 3: Solution 3: Three most influential conservative bloggers, from the perspective of liberal bloggers,

Blog	Sub-Graph Authority	Overall Authority
instapundit.com	0.5103	0.6408
andrewsullivan.com	0.2952	0.4237
powerlineblog.com	0.2656	0.499

Question 4: From the perspective of conservative bloggers, what are the 3 most influential liberal bloggers?

Solution 4

Following the same approach as solution 3, the 3 most influential liberal bloggers were calculated as follows:

```
# Saving 3 most influential liberal nodes from perspective of conservatives
influential_l <- influential_blogs(conservatives, liberals)
```

Question 4 Results:

```
# Initializing results to save influential liberal blogs
results <- c()
# Saving blog names, authority in sub-graph and overall authority in full graph
for(i in 1:length(influential_l)) {
  sub_auth <- round(as.double(influential_l[i]), digits = 4)
  overall_auth <- round(as.double(V(blogs_graph)[V(blogs_graph)$name ==
                        names(influential_l[i])]$authority), digits = 4)
  results <- rbind(results, c(names(influential_l[i]), sub_auth, overall_auth))
}
# Printing 3 most influential liberal nodes from perspective of conservatives
kable(results, col.names = c("Blog", "Sub-Graph Authority", "Overall Authority"),
      caption = "Solution 3: Three most influential liberal bloggers, from the
      perspective of conservative bloggers, ", booktabs=T, align="c") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 4: Solution 3: Three most influential liberal bloggers, from the perspective of conservative bloggers,

Blog	Sub-Graph Authority	Overall Authority
atrios.blogspot.com	1	0.9361
dailykos.com	1	1
talkingpointsmemo.com	0.8575	0.9617

Question 5: Identify the 3 most 'neutral' blogs in the dataset.

Solution 5

- To identify 3 most neutral blogs, the nodes with equal number of liberal and conservative neighbors with same number of incoming and outgoing links are identified.
- The nodes found using this process mostly have one or two neighbours, hence the nodes with highest degree are selected to be most neutral, because these link to more nodes in the graph and yet have equal number of links for both communities.

```
neutrals <- c()
# For all nodes in the graph checking if they are neutral nodes
for(node in V(blogs_graph)$name) {
  # getting the neighbours that have links into the node
  alters <- neighbors(blogs_graph, node, mode = 'in')
  # setting alters to the names of the nodes
  alters <- V(blogs_graph)[V(blogs_graph) %in% alters]$name
  # getting the conservative neighbour nodes
  c_alters_in <- intersect(alters, conservatives$name)
  # getting the liberal neighbour nodes
  l_alters_in <- intersect(alters, liberals$name)
  # checking if the node has equal number of liberal and conservative
  # neighbours with links coming into the node
  if(length(c_alters_in) == length(l_alters_in)){
    # getting the neighbours that have links from the node
    alters <- neighbors(blogs_graph, node, mode = 'out')
    # setting alters to the names of the nodes
    alters <- V(blogs_graph)[V(blogs_graph) %in% alters]$name
    # getting the conservative neighbour nodes
    c_alters_out <- intersect(alters, conservatives$name)
    # getting the liberal neighbour nodes
    l_alters_out <- intersect(alters, liberals$name)
    # checking if the node has equal number of liberal and conservative
    # neighbours with links coming from the node
    if(length(c_alters_out) == length(l_alters_out)){
      # getting degree of each node with equal number of
      # conservative and liberal neighbours
      degree <- V(blogs_graph)[V(blogs_graph)$name == node]$deg
      # saving the node with its degree if not zero
      if(degree != 0) {
        neutrals[[node]] <- degree
      }
    }
  }
}
# Sorting the neutral nodes by degree and fetching the top 3 from the list
neutral_nodes <- neutrals[order(unlist(neutrals), decreasing=TRUE)][1:3]
cat("3 most neutral nodes are:",
    toString(names(neutral_nodes)))
```

```
## 3 most neutral nodes are: lonewacko.com, blog.mintruth.com, awards5.tripod.com/tarasblog
```

Question 5 Result: Neutral nodes and their Neigbhours

```
# Finding neighbours of first neutral node and printing the result
results <- c()
alters <- neighbors(blogs_graph, names(neutral_nodes[1]))$name
for(alter in 1:length(alters)) {
  value <- V(blogs_graph)[V(blogs_graph)$name == alters[alter]]$value
  results <- rbind(results, c(alter, alters[alter],
                              ifelse(value==0, "Liberal", "Conservative")))
```

```
}
kable(results, col.names = c("Sr. No.", "Blog", "Type"),
      caption = paste("Neighbour of neutral node ",
                      names(neutral_nodes[1])), booktabs=T, align="c") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 5: Neighbour of neutral node lonewacko.com

Sr. No.	Blog	Type
1	atrios.blogspot.com	Liberal
2	bluelemur.com	Liberal
3	buzzmachine.com	Liberal
4	dailykos.com	Liberal
5	democraticunderground.com	Liberal
6	juancole.com	Liberal
7	maxspeak.org/mt	Liberal
8	mydd.com	Liberal
9	oliverwillis.com	Liberal
10	presidentboxer.blogspot.com	Liberal
11	talkingpointsmemo.com	Liberal
12	talkleft.com	Liberal
13	thetalkingdog.com	Liberal
14	blogsforbush.com	Conservative
15	dailypundit.com	Conservative
16	drudgereport.com	Conservative
17	freerepublic.com	Conservative
18	gopbloggers.org	Conservative
19	instapundit.com	Conservative
20	littlegreenfootballs.com/weblog	Conservative
21	michellemalkin.com	Conservative
22	polipundit.com	Conservative
23	redstate.org	Conservative
24	samizdata.net/blog	Conservative
25	thatliberalmedia.com	Conservative
26	truthlaidbear.com	Conservative

```
# Finding neighbours of second neutral node and printing the result
results <- c()
alters <- neighbors(blogs_graph, names(neutral_nodes[2]))$name
for(alter in 1:length(alters)) {
  value <- V(blogs_graph)[V(blogs_graph)$name == alters[alter]]$value
  results <- rbind(results, c(alter, alters[alter],
                              ifelse(value==0, "Liberal", "Conservative")))
}
kable(results, col.names = c("Sr. No.", "Blog", "Type"),
      caption = paste("Neighbour of neutral node ",
                      names(neutral_nodes[2])), booktabs=T, align="c") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 6: Neighbour of neutral node blog.mintruth.com

Sr. No.	Blog	Type
1	blog.johnkerry.com	Liberal
2	smirkingchimp.com	Liberal
3	talkleft.com	Liberal
4	blogsforbush.com	Conservative
5	deanesmay.com	Conservative
6	outsidethebeltway.com	Conservative

```
# Finding neighbours of third neutral node and printing the result
results <- c()
alters <- neighbors(blogs_graph, names(neutral_nodes[3]))$name
for(alter in 1:length(alters)) {
  value <- V(blogs_graph)[V(blogs_graph)$name == alters[alter]]$value
  results <- rbind(results, c(alter, alters[alter],
                              ifelse(value==0, "Liberal", "Conservative")))
}
kable(results, col.names = c("Sr. No.", "Blog", "Type"),
      caption = paste("Neighbour of neutral node ",
                      names(neutral_nodes[3])), booktabs=T, align="c") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 7: Neighbour of neutral node awards5.tripod.com/tarasblog

Sr. No.	Blog	Type
1	farleft.blogspot.com	Liberal
2	scottcsmith.net	Liberal
3	martinipundit.com	Conservative
4	truthlaidbear.com	Conservative

Acknowledgments:

Following resources were referred to perform above tasks:

1. Lada A. Adamic and Natalie Glance. 2005. The political blogosphere and the 2004 U.S. election: divided they blog. In Proceedings of the 3rd international workshop on Link discovery (LinkKDD '05). Association for Computing Machinery, New York, NY, USA, 36–43. DOI:https://doi.org/10.1145/1134271.1134277
2. Dr. Conor Hayes. (2022). Centrality Week 3
3. Dr. Conor Hayes. (2022). Connectivity
4. Dr. Conor Hayes. 2022. Week 1 - Analysing the Adjacency Matrix of a Food Web
5. Dr. Conor Hayes. 2022. Exploratory Analysis of Game of Thrones Network
6. Dr. Conor Hayes. 2021. Centrality
7. Dr. Conor Hayes. 2022. Analysing a Twitter Graph