CT5113: Assignment 3 - Analysis of Word Association Network

Submitted By: Kalyani Prashant Kawale | Student ID: 21237189

Part 1

Task: Creation of Subgraph for Selected Target Words from the Word Association Network.

Solution:

Part 1.1. Data Setup:

- The word association network stored in the *pajek* file is loaded using igraph's **read_graph()** method and stored in the igraph variable *word_graph*.
- The graph is further converted into an undirected graph and simplified by removing the 4 self-loops present in the graph using igraph methods **as.undirected()** and **simplify()** respectively.
- A *deg* attribute is added to the graph to store the degree of each node, calculated using igraph **degree()** function, which is further used in selecting the target words.
- Each edge weight is re-scaled to fall in the range of 0 to 1 using **rescale()** function of scales package.

```r
# Loading necessary libraries
library(igraph)
library(ggrepel)
library(ggraph)
# library(RColorBrewer)
library(knitr)
library(kableExtra)
library(scales)

source("my_modularity.R")
source("cluster_measurements.R")

# Reading word pairs graph from pajek file
word_graph <- read_graph(file="WordPairs.txt",format="pajek")
# Removing edge direction from graph
word_graph <- as.undirected(word_graph)
# Removing self loops from graph
word_graph <- simplify(word_graph)
# Adding degree column for each node
V(word_graph)$deg <- degree(word_graph)
# Re-scaling the weights of each edge to be in the range 0 to 1
E(word_graph)$weight<-rescale(E(word_graph)$weight)
```

Part 1.2. Selection of 3 Target Words:

- As the requirement states that the target words need to be cue words, the text file containing the cue word information for each node is read using utils method **read.csv()**.
- The cue word information is then added as an attribute to the *word_graph* graph for each node.
- For selecting the target words, a separate list of graph nodes is created with cue words only and the nodes with degree higher than 100 are filtered to make the word selection list compact.
- The **sample()** method was used to get 3 random indices from the list of cue word nodes, the indices given by the sample method were then used to get the target words.

```r
# Selecting 3 target words and showing that they are cue words

# Reading the cue word indicator for each vertex stored in cue.txt
cue <- read.csv("cue.txt")
# Removing the information rows
cue <- cue[-c(1, 2, 3),]
# Adding cue value for each vertex in graph
V(word_graph)$cue <- cue
# Getting cue words from word pairs graph
cue_words <- V(word_graph)[V(word_graph)$cue == 1]
# Selecting cue words with degrees higher than 100
words_temp <- cue_words[cue_words$deg > 100]

# Following code was used to randomly select 3 target indices from list of
# cue word nodes with degree greater than 100
# selected_indices <- sample(1:length(words_temp), 3)
# Resultant indices of the sample method
selected_indices <- c(25, 54, 5)

# Getting the target word nodes for selected indices
selected_words <- words_temp[selected_indices]

knitr::kable(selected_words$name, align = "c", booktabs=T,
             col.names = c("Selected Target Words")) %>%
  kable_styling(full_width = F, latex_options = "HOLD_position")
```

Selected Target Words
TREE
SOFT
CHILD

Part 1.3. Creating the Subgraphs for Selected Target Words:

- A function **get_subgraph()** shown below is used to create the subgraphs for each target word.
- **get_subgraph()** uses the sampling strategy of multiple short random walks by allowing different settings of number of runs and number of steps to be taken of the random walk.
- The edges with low weights are removed from the subgraph created by the random walk sampling and the singleton vertices created after removing such edges are also remvoed from the graph.
- The threshold for edge weight removal is based on the analysis of minimum weights in the subgraph created for each target word.

- Further, to remove disconnected smaller components from the graph the **components** method of igraph is used to get the strongly connected component whose vertices are used to create the final subgraph, which ensures removal of such disconnected components from the subgraph.
- The number of runs of the **random_walk()** method, number of steps of random walk and the threshold for edge removal was decided for each target word by manual analysis of the graphs created and based on size of graph and the presence of both cue and non-cue words.

```r
# Using random walk sampling to create sub-graphs
get_subgraph <- function(start_node, steps=10, runs=50, threshold=0.005) {
  # Large number of small walks
  walk_rand <- c()
  for (i in 1:runs) {
    walk_rand <- c(walk_rand,
        random_walk(word_graph, start=start_node, steps=steps, stuck="return"))
  }
  # Removing duplicate nodes
  walk_rand <- unique(walk_rand)
  # Creating sub-graph with nodes visited in the random walk
  sub_graph <- induced_subgraph(word_graph, walk_rand)
  # Deleting edges with weight less than threshold
  sub_graph <- delete.edges(sub_graph, which(E(sub_graph)$weight < threshold))
  # Deleting singleton nodes in graph after removing low weighted edges
  sub_graph <- delete.vertices(sub_graph, which(degree(sub_graph)==0))
  # Getting components of sub_graph
  sub_comp <- components(sub_graph)
  # Getting the strongly connected component
  scc <- which(sub_comp$membership == which.max(sub_comp$csize))
  # Getting nodes from strongly connected component of the sub_graph
  scc_nodes <- V(sub_graph)[V(sub_graph) %in% scc]
  # Inducing sub graph after removing multiple unconnected smaller components,
  # to get a connected sub_graph
  sub_graph <- induced_subgraph(sub_graph, scc_nodes)
  return(sub_graph)
}
```

- Generating the sub-graphs for each target word using the **get_subgraph()** method defined above,

```r
set.seed(100)
# Initializing list for holding the 3 sub-graphs
sub_graphs <- list()
# Generating sub_graph for target word TREE
sub_graph1 <- get_subgraph(selected_words[1], steps=5, runs=34, threshold=0.002)
sub_graphs$sub_graph1 <- sub_graph1
# Generating sub_graph for target word SOFT
sub_graph2 <- get_subgraph(selected_words[2], steps=6, runs=20, threshold=0.002)
sub_graphs$sub_graph2 <- sub_graph2
# Generating sub_graph for target word CHILD
sub_graph3 <- get_subgraph(selected_words[3], steps=8, runs=24, threshold=0.005)
sub_graphs$sub_graph3 <- sub_graph3
```

- Following method was written to generate information table for the sub-graphs created in the preceding section.

```r
display_subgraph_info <- function(sub_graphs, target_words) {
  # Calculating and saving the average degree for all sub-graphs by taking mean
  # of degree function applied on sub-graphs
  avg_degree = round(as.vector(unlist(lapply(sub_graphs,
                    function(x){mean(degree(x))}))), 2)
  # Calculating average path length using mean_distance function for all sub-graphs
  avg_path_len = round(as.vector(unlist(lapply(sub_graphs, mean_distance))), 2)
  # Calculating average clustering coefficient using transitivity function
  # for all sub-graphs
  avg_clustering_ceoff = round(as.vector(unlist(lapply(sub_graphs,
                    function(x){transitivity(x, type="localaverage")}))), 2)

  # Saving all information for each sub-graph into data-frame information
  information <- data.frame(Number=1:length(sub_graphs),
                Target_Word=target_words,
                # using vcount to calculate total no. of nodes
                No_of_Nodes=as.vector(unlist(lapply(sub_graphs, vcount))),
                No_of_cue_words=as.vector(unlist(lapply(sub_graphs,
                  function(x){length(which(V(x)$cue == 1))}))),
                No_of_non_cue_words=as.vector(unlist(lapply(sub_graphs,
                  function(x){length(which(V(x)$cue == 0))}))),
                # using ecount to calculate total no. of edges
                No_of_Edges=as.vector(unlist(lapply(sub_graphs, ecount))),
                Average_Degree=avg_degree,
                Average_Path_Length=avg_path_len,
                # getting the maximum eccentricity to get the diameter
                Diameter=as.vector(unlist(lapply(sub_graphs,
                  function(x){max(eccentricity(x))}))),
                Average_Clustering_Coefficient=avg_clustering_ceoff)

  # Plotting the information in a table
  knitr::kable(information, align="c", booktabs=T,
             col.names = c("No.",
                           "Target Word",
                           "Nodes",
                           "Cue Nodes",
                           "Non-Cue Nodes",
                           "Edges",
                           "Average Degree",
                           "Average Path Length",
                           "Diameter",
                           "Average Clustering Coefficient")) %>%
    kable_styling(full_width = F, latex_options = "HOLD_position") %>%
    column_spec(1, width="0.5cm") %>%
    column_spec(c(3,4,6), width="1cm") %>%
    column_spec(c(2,5,8,7), width="1.25cm") %>%
    column_spec(10, width="2cm") %>%
    row_spec(0, bold=TRUE)
}
```

Part 1.4. Subgraph Results:

No.	Target Word	Nodes	Cue Nodes	Non-Cue Nodes	Edges	Average Degree	Average Path Length	Diameter	Average Clustering Coefficient
1	TREE	119	115	4	232	3.90	3.55	8	0.24
2	SOFT	91	86	5	153	3.36	4.05	10	0.24
3	CHILD	112	104	8	170	3.04	4.92	13	0.26

Part 2

Task: Community Detection for Subgraphs Created in Part 1.

Solution:

Part 2.1 Analysis and Display Methods:

The three community detection algorithms ran on each target word to get communities are,

1. Girvan-Newman Edge Betweenness Algorithm
2. Louvain Algorithm
3. Label Propagation Algorithm

- The three community detection algorithms were applied using the **cluster_edge_betweenness()**, **cluster_louvain()** and **cluster_label_prop()** functions of igraph.
- Following function is used to apply the three algorithms to each target word and return the results,

```r
apply_detection <- function(sub_graph, target) {
## sub-graph community detection
communities <- list()
eb.community <- cluster_edge_betweenness(sub_graph)
communities$`eb` <- eb.community

louvain.community <-cluster_louvain(sub_graph)
communities$`louvain` <- louvain.community

labprop.community <- cluster_label_prop(sub_graph)
communities$`labprop` <- labprop.community

# Getting the measurements for each algorithm's results
algo_results <- algorithm_comparison(sub_graph, communities)

return(list(algo_results=algo_results, communities=communities))
}
```

- To display the results of all three algorithms on each sub-graph and then displaying the result of the selected algorithm following two methods have been written,

```r
# Method to create a data-frame object to display results of three
# community detection algorithms
# Note: The cluster_measurement method given in week 10 materials has been used [6]
algorithm_comparison <- function(graph, communities_data) {
  # getting the cluster measurements for community object of edge betweenness
  cluster.measurement <- cluster_measurements(graph, communities_data[[1]])
  communities_results <-
    # saving the community detection algorithm name obtained using algorithm function
    data.frame(Algorithm=algorithm(communities_data[[1]]),
    # saving the sizes of all communities
    Size=length(communities_data[[1]]),
    # saving the global modularity of communities
    Modularity=round(modularity(communities_data[[1]]), 3),
    # checking and saving the number of communities with edge density higher than
    # the total edge density of the sub-graph to enable
    # internal evaluation of the communities formed
    no.clusters.higher.density=length(which(cluster.measurement$density >

  # Saving the same information as above to remaining algorithm results
  # into the data-frame
  for (i in 2:(length(communities_data))) {
    cluster.measurement <- cluster_measurements(graph, communities_data[[i]])
    communities_results <- rbind(communities_results, data.frame(
            Algorithm=algorithm(communities_data[[i]]),
            Size=length(communities_data[[i]]),
            Modularity=round(modularity(communities_data[[i]]), 3),
            no.clusters.higher.density=length(which(cluster.measurement$density >
                cluster.measurement$density[cluster.measurement$cluster==-1]))))
  }
  return(communities_results)
}

# Method to select an algorithm from the three trialled on sub-graph and
# create data-frame to display communities for the selected algorithm
selected_community_results <- function(sub_graph, communities, interpretations) {
  # Selecting the algorithm results with the maximum modularity
  selected_algo <- sub_graph$Algorithm[which.max(sub_graph$Modularity)]
  # Getting the communities for the selected algorithm from the communities list
  selected_community <- NULL
  for (i in 1:(length(communities))) {
    if(algorithm(communities[[i]])==selected_algo) {
      selected_community <- communities[[i]]
    }
  }
  # Getting the names of nodes in all communities
  clusters <- communities(selected_community)
  for (x in 1:length(clusters)) {
    # Converting the communities into strings
    clusters[x] <- tolower(toString(clusters[[x]]))
  }
  # Creating data-frame to display all communities,
  # with their nodes, sizes and interpretation
  community_df <- data.frame(no=1:length(selected_community),
                             cluster=clusters,
                             size=as.data.frame(sizes(selected_community))$Freq,
                             interpretation=interpretations)
  return(list(community_df=community_df, selected_community=selected_community))
}
```

Part 2.2 Applying Community Detection on Target Word TREE:

```
# Applying the three community detection algorithms
results1 <- apply_detection(sub_graph1, selected_words[1]$name)

# Displaying the results for each algorithm
knitr::kable(results1$algo_results, booktabs=TRUE, align = "c",
             caption = "Community Detection Algorithm Results for TREE Sub-Graph",
             col.names = c("Algorithm", "Size of Community", "Modularity",
      "No. of communities with edge density higher than total edge density")) %>%
  kable_styling(full_width=T, latex_options = "HOLD_position") %>%
  row_spec(0, bold=TRUE)
```

Table 1: Community Detection Algorithm Results for TREE Sub-Graph

Algorithm	**Size of Community**	**Modularity**	**No. of communities with edge density higher than total edge density**
edge betweenness	11	0.445	11
multi level	13	0.738	13
label propagation	25	0.719	25

- The algorithm selected from the given three algorithms is Louvain Algorithm (labeled multi level) because it has the maximum modularity among all other algorithms, the selection is done automatically using **selected__community__results** function which selects the algorithm with maximum modularity and returns the results as follows.
- Maximum modularity is chosen to select community detection algorithm as it presents an intuitive sense and measure of valid communities, in the absence of the ground truth.

```
community_results1 <-
  selected_community_results(results1$algo_results, results1$communities, interpretations1)

knitr::kable(community_results1$community_df,
  caption = "Selected Community Detection Algorithm Results for TREE Sub-Graph") %>%
  kable_styling(full_width = F, latex_options = "HOLD_position") %>%
    column_spec(c(1, 3), width="0.5cm") %>%
    column_spec(2, width="10cm") %>%
  column_spec(4, width="5cm") %>%
    row_spec(0, bold=TRUE)
```

Table 2: Selected Community Detection Algorithm Results for TREE Sub-Graph

no	cluster	size	interpretation
1	apple, red, fruit, pear, peach, computer, core, berry, blueberry, cherry, lemon, crab, nuts	13	Fruits that grow on trees
2	animal, squirrel, chipmunk, beaver, doe, elk, possum	7	Animals known to live on trees
3	muscle, walk, leg, thigh, crawl, limp, sneak, steps, slither	9	Organs and activities with heirarchical order like a tree structure
4	top, mountain, climb, pin, shirt, ladder, winding, rung, twisting	9	Things related to climbing, like climbing trees
5	water, land, clean, wash, dry, ditch, gully, raft, pure	9	Water and related adjectives, water and land are necessary for growth of trees
6	car, tall, small, park, growth, erect, central, compact, meter	9	Heights and characteristics for trees
7	log, tree, environment, fungus, scum, moss, gross, bite, nature, limb, ax, bark, leaf, foul, mint, gel, timber, grove, lumber	19	Parts of trees, including by-products and type of vegetation
8	add, paper, love, grass, heart, worry, nice, operation, knife, cut, rub, scratch, threat, tear, caress, divide, share, wasted, giving	19	Things related to grass and cutting of grass or trees
9	burn, ash, cigarette, smoky, smelt, delt	6	Things related to fire, perhaps relating to forest fires
10	fix, decorate, room, cabbage, lettuce, patch, cornbeef, decoration, tasty, parsley	10	Leafy vegetables
11	flower, bud, stem	3	Parts of plants and trees
12	eraser, depletion, erase	3	Words indicating degradation, perhaps relating to forest degradation
13	click, tick, lice	3	Bugs that could be present in trees

Part 2.3 Applying Community Detection on Target Word SOFT:

```r
# Applying the three community detection algorithms
results2 <- apply_detection(sub_graph2, selected_words[2]$name)

# Displaying the results for each algorithm
knitr::kable(results2$algo_results, booktabs=TRUE, align = "c",
             caption = "Community Detection Algorithm Results for SOFT Sub-Graph",
             col.names = c("Algorithm", "Size of Community", "Modularity",
      "No. of communities with edge density higher than total edge density")) %>%
  kable_styling(full_width=T, latex_options = "HOLD_position") %>%
  row_spec(0, bold=TRUE)
```

Table 3: Community Detection Algorithm Results for SOFT Sub-Graph

Algorithm	**Size of Community**	**Modularity**	**No. of communities with edge density higher than total edge density**
edge betweenness	9	0.552	9
multi level	12	0.782	12
label propagation	24	0.757	24

- The algorithm selected from the given three algorithms is Louvain Algorithm because it has the maximum modularity among all other algorithms, similar to previous target word the selection is done automatically using **selected_community_results()** function which selects the algorithm with maximum modularity and returns the results as follows,

```
community_results2 <-
  selected_community_results(results2$algo_results, results2$communities, interpretations2)

knitr::kable(community_results2$community_df,
  caption = "Selected Community Detection Algorithm Results for SOFT Sub-Graph") %>%
  kable_styling(full_width = F, latex_options = "HOLD_position") %>%
    column_spec(c(1, 3), width="0.5cm") %>%
    column_spec(2, width="10cm") %>%
  column_spec(4, width="5cm") %>%
    row_spec(0, bold=TRUE)
```

Table 4: Selected Community Detection Algorithm Results for SOFT Sub-Graph

no	**cluster**	**size**	**interpretation**
1	strong, tough, scared, nerves, wrestling, board, fake, stiff, harsh, terrified	10	Opposites to soft
2	people, boy, pretty, cute, animals, bill, nice, fine, expensive, mix, lace, chore, ordinary, nephew, extravagant, extraordinary, satin	17	Soft materials like lace, satin and other words relating to traditional dance balls
3	fight, river, scar, struggle, conflict, clash, riot	7	Things opposing softness
4	slow, hyper, allergy, cold, sleep, blanket, serene, tranquil, elderly, driving, grandma, drift, iceberg, flying, quilt, mellow	16	Things that could be described as soft
5	give, teach, send, launch	4	Providing or starting something, with a soft start perhaps
6	achieve, earn, make	3	Words related to earning
7	paper, note, material, pad	4	Writing material which has soft quality
8	woman, mother, legs, hands, female, skirt	6	Softness often associated with female qualities
9	bra, touch, skin, soft, weave, velvet, rug, plush, cushion, flannel, harp, padding, paddy, daddy	14	Soft clothing material
10	kiss, lips	2	Softness invovled in the act
11	beer, tavern, draft	3	Words related to drinking
12	chair, comfort, bench, sit, discomfort	5	Things providing softness and relaxation

Part 2.4 Applying Community Detection on Target Word CHILD:

```
# Applying the three community detection algorithms
results3 <- apply_detection(sub_graph3, selected_words[3]$name)

# Displaying the results for each algorithm
knitr::kable(results3$algo_results, booktabs=TRUE, align = "c",
             caption = "Community Detection Algorithm Results for CHILD Sub-Graph",
             col.names = c("Algorithm", "Size of Community", "Modularity",
      "No. of communities with edge density higher than total edge density")) %>%
  kable_styling(full_width=T, latex_options = "HOLD_position") %>%
  row_spec(0, bold=TRUE)
```

Table 5: Community Detection Algorithm Results for CHILD Sub-Graph

Algorithm	**Size of Community**	**Modularity**	**No. of communities with edge density higher than total edge density**
edge betweenness	12	0.626	12
multi level	13	0.788	13
label propagation	25	0.731	25

- The algorithm selected from the given three algorithms is Louvain Algorithm because it has the maximum modularity among all other algorithms, similar to previous target words the selection is done automatically using **selected__community__results()** function which selects the algorithm with maximum modularity and returns the results as follows,

```
# Creating the result data frame for selected community
community_results3 <-
  selected_community_results(results3$algo_results, results3$communities, interpretations3)

knitr::kable(community_results3$community_df,
  caption = "Selected Community Detection Algorithm Results for CHILD Sub-Graph") %>%
  kable_styling(full_width = F, latex_options = "HOLD_position") %>%
    column_spec(c(1, 3), width="0.5cm") %>%
    column_spec(2, width="10cm") %>%
  column_spec(4, width="5cm") %>%
    row_spec(0, bold=TRUE)
```

Table 6: Selected Community Detection Algorithm Results for CHILD Sub-Graph

no	cluster	size	interpretation
1	child, adolescent, teenager, kid, juvenile, diaper, growth, swing, delinquent	9	Stages and growing up of a child
2	bad, criminal, poor, complain, behavior, mood, vagrant, unhappy, grumble, dissatisfaction, bicker, punish, jobless, unpleasant, treason	15	Negative feelings relating to activities of children
3	race, green, hospital, white, hose, looks, color, focus, eyes, flower, rod, bright, long, crayon, days, optimistic	16	Objects relating to drawing activity carried out by children
4	paper, fold, print, writing, rip	5	Things related to writing, perhaps learning to write
5	fly, ticket, plane, soup, spider, creep, sneak	7	Things children might do or use while playing
6	busy, tired, stress, hurry, burden, load, impatience	7	Feelings that might be experienced in handling difficult children
7	fun, enjoy, fair, festival, prank, high school	6	Playful things invovled in activities of children
8	girl, beautiful, attractive, female, tease, blonde, virgin, virtue	8	Words associated with female children
9	next, mother, friend, look, letters, student, relative, brother, education, peer, kin, daughter, visitor, stranger, guardian, folk, tangent, sine	18	People surrounding children
10	hate, mad, anger, kill, incense, insult	6	Show of anger, maybe presented by or in front of children
11	almond, chocolate, candy, bar, crave	5	Sweets enjoyed by children
12	hand, raise, glove, paw, grip	5	Words related to holding, that children learn as they grow
13	scapegoat, bully, coward, wimp, butthead	5	Relating to bullying that young children may face

Part 3

Task: Visualizing Sub-Graphs with their Communities

Solution:

Part 3.1 Method for Plotting the Sub-Graph

- Following method has been referred from Assignment 3 worksheet [9] and customized for given task,

```
# The first 10 colors in the palette are taken from Data Visualization Assignment 2,
# additional colors have been added to cover all communities
custom_palette <- c('#9F69E1','#59C7DB','#F1954D','#E562A5','#6C7AEA', '#59DCB2',
                    '#CB464E','#D065DF', '#601B52','#005437','#964B00',
                    '#deb887', '#F19CBB')

# Adjusting alpha value of the colors
add.alpha <- function(cols, alpha) rgb(t(col2rgb(cols)/255), alpha = alpha)
colours <- add.alpha(custom_palette, 1)
```

```r
# Function to plot sub-graph
plot_subgraph <- function(sub_graph, community_result,
                          graph_number, selected_word, colours=c('#0077b2')) {
  # Setting the degree of each node in the sub_graph
  V(sub_graph)$deg <- degree(sub_graph)
  # Selecting the central nodes in each community of the sub-graph based on
  # degree centrality
  central_nodes <- c()
  for(x in communities(community_result$selected_community)) {
    community <- V(sub_graph)[which(V(sub_graph)$name %in% as.vector(unlist(x)))]
    central_nodes <- c(central_nodes, community[which.max(community$deg)])
  }
  # Setting the vertex size relative to the degree of the nodes
  vertex_size <- 0.85 + degree(sub_graph) * 0.5
  # Plotting the sub-graph using ggraph library
  ggraph(sub_graph, layout = layout_with_graphopt(sub_graph)) +
    geom_edge_link(start_cap = circle(1.5, "mm"),
                   end_cap = circle(1.5, "mm"),
                   edge_width = 0.05,
                   alpha = 0.11) +
    geom_node_point(aes(size = vertex_size),
                    alpha = 0.7,
                    colour = colours) +
    geom_node_text(
      # Setting labels only for the most central node in each community
      aes(label = ifelse(name %in% names(central_nodes), name, NA)),
      size = 6.5,
      fontface = "bold",
      repel = TRUE,
      color=colours,
      alpha=1
    ) +
    ggtitle(paste0("TARGET WORD ", graph_number, ": ", selected_word)) +
    theme(legend.position = "none",
          panel.background = element_rect(fill = "white"),
          plot.title = element_text(size=18, hjust = 0.5, face="bold"))
}
```

Part 3.2 Sub-Graph for Target Word TREE:

```r
plot_subgraph(sub_graph1, community_results1, 1, selected_words[1]$name,
              colours =
        colours[community_results1$selected_community$membership])
```

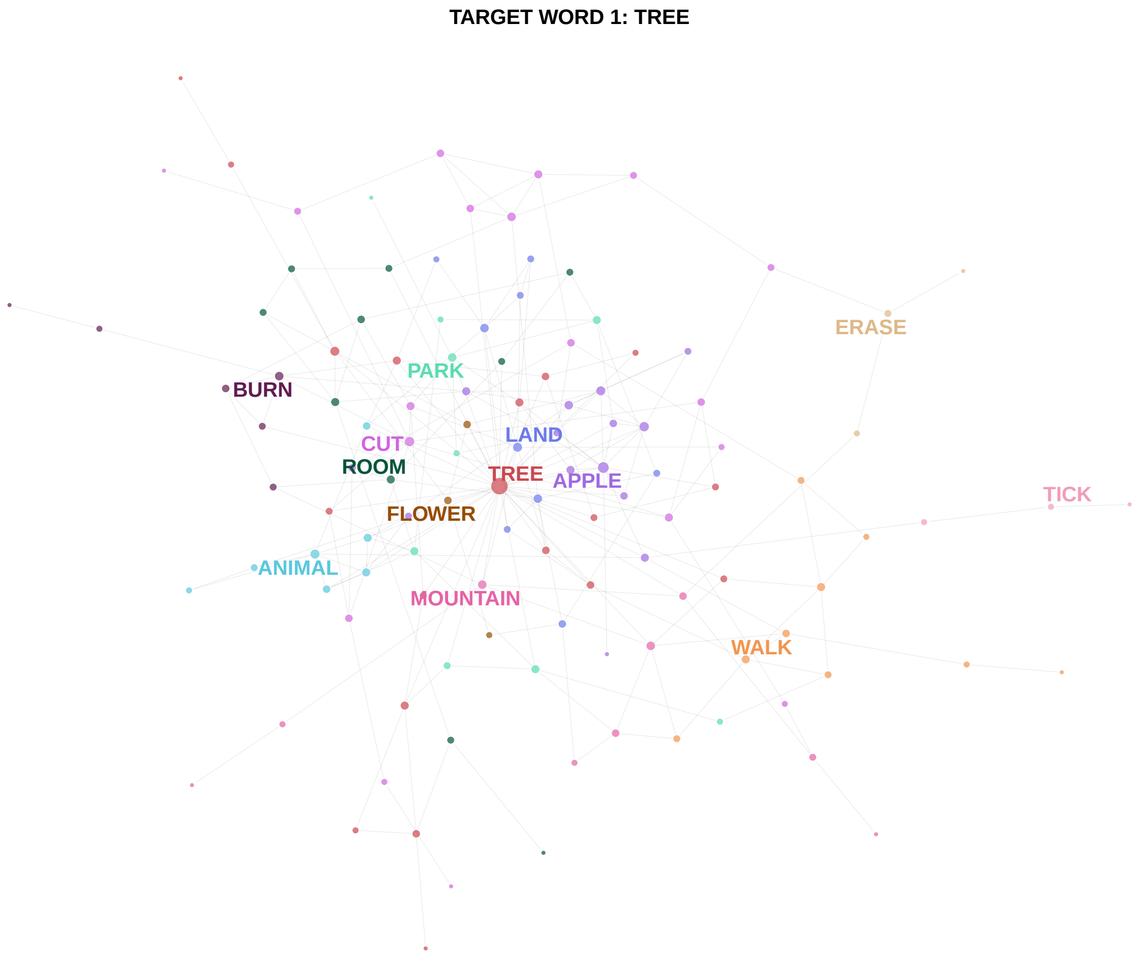

Figure 1: Sub-graph of Target Word Tree along with the Detected Communities Colored with Different Colors and Labelled by the Most Central Node in Each Community

Part 3.3 Sub-Graph for Target Word SOFT:

```
plot_subgraph(sub_graph2, community_results2, 2, selected_words[2]$name,
              colours =
        colours[community_results2$selected_community$membership])
```

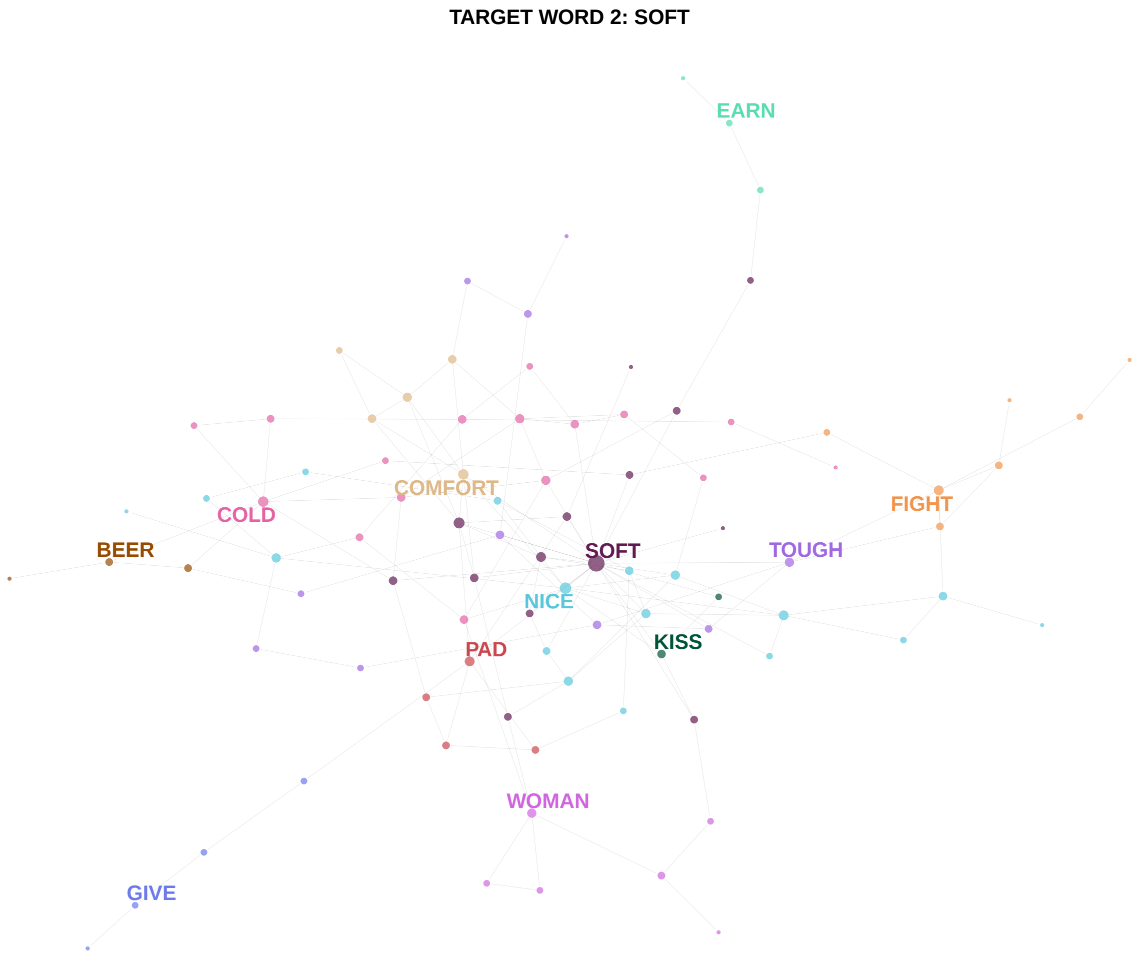

Figure 2: Sub-graph of Target Word SOFT along with the Detected Communities Colored with Different Colors and Labelled by the Most Central Node in Each Community

Part 3.4 Sub-Graph for Target Word CHILD:

```
plot_subgraph(sub_graph3, community_results3, 3, selected_words[3]$name,
              colours =
        colours[community_results3$selected_community$membership])
```

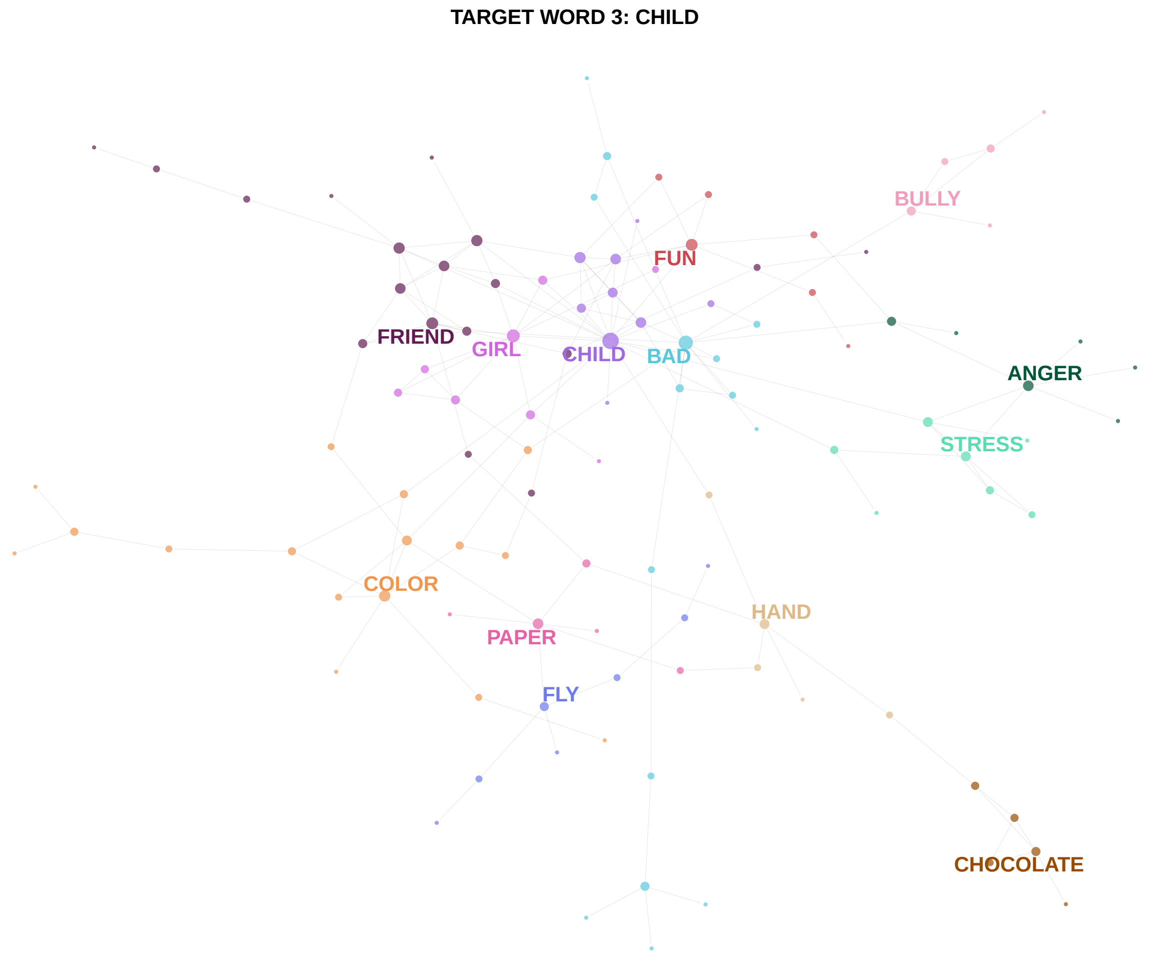

Figure 3: Sub-graph of Target Word CHILD along with the Detected Communities Colored with Different Colors and Labelled by the Most Central Node in Each Community

Acknowledgements

Following resources were referred to complete the given tasks,

[1] How To Remove Rows From an R Data Frame – With Examples. Available at: https://www.programmingr.com/examples/r-dataframe/add-delete-rows/ [Accessed on:19/03/2022]

[2] sample: Random Samples and Permutations. Available at: https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/sample [Accessed on:19/03/2022]

[3] Neighboring (adjacent) vertices in a graph. Available at: https://igraph.org/r/doc/neighbors.html [Accessed on:19/03/2022]

[4] Plot labels for specific geom_node_text. Available at: https://stackoverflow.com/questions/47175541/plot-labels-for-specific-geom-node-text [Accessed on: 20/3/2022]

[5] lapply: Apply a Function over a List or Vector Available at: https://rdrr.io/r/base/lapply.html [Accessed on: 26/3/2022]

[6] Dr. Conor Hayes. (2022). Week 10 - R Code

[7] Dr. Conor Hayes. (2022). Week 9 Lecture Notes

[8] Dr. Conor Hayes. (2022). Week 10 Lecture Notes

[9] Dr. Conor Hayes. (2022). Assignment 3 - CT5113- Web andNetwork Science