Assignment3

Diksha Srivastava

23/03/2022

Part 1: Creating subgraph

```
# loading libraries
library(igraph)
library(kableExtra)
library(ggraph)
```

Reading the Graph

```
# reading the graph
g<- read_graph(file="E:/Users/Diksha/Desktop/NUIG/Web Science/WordPairs.txt",
               format="pajek")
# making the graph undirected
g<- as.undirected(g)
# removing self loop
g<-simplify(g)
# reading cue file
cue <- read.table("E:/Users/Diksha/Desktop/NUIG/Web Science/cue.txt",sep="\n")
```

Selecting three words

I chose three target words: MOVIE, HOME, LIFE. I chose them based on their degree. They had a degree of 107, 106, 110 respectively. I did this to ensure that the graph which will be created would be semantically similar to target words. If a less degree word was chosen then a completely different community with different context could have been created.

In order to check whether these target words are cue or not, I found the vertex ID of these target words and then this vertex ID acted as an index to the cue vector. On finding the cue ID, we can see that they have an id 1 which indicates that they are cue words.

```
# selecting three words.
target_words <- c("MOVIE", "HOME", "LIFE")
# creating a dataframe to show if these words are cue or not.
df_words <- data.frame(MOVIE =
                         cue[as.numeric(V(g)[which(V(g)$name == target_words[1])]),],
                       HOME =
                         cue[as.numeric(V(g)[which(V(g)$name == target_words[2])]),],
                       LIFE =
```

Table 1: Cue Words

	MOVIE	HOME	LIFE
Cue	1	1	1

```r
                         cue[as.numeric(V(g)[which(V(g)$name == target_words[3])]),])
row.names(df_words) <- "Cue"
# using kable to show the table.
df_words %>%
  kbl(align = "ccc", caption = "Cue Words")
```

Creating subgraphs

In order to create subgraphs, I have performed several short random walks on the graph. I took a step size of 10 and ran it 15 times, thus, we would get 150 vertices in total. I tried with the step size of 5 and 15 as well and ran it 30, 10 times respectively. I found that when I was using a step size of 5 then the communities created were quite big of around size 20 and many other communities of size 2/3. Thus, the communities created were either too big or too small which is not the ideal scenario. However, when I took a step size of 15 and ran it 10 times, the communities created were not semantically similar to the target word. Hence, I chose a step size that lies between them i.e., 10. Thus, the communities created from this were neither too large nor too small (with 1-2 exceptions).

The parameter stuck = "return" suggests that if the random walk gets stuck at a node that has no out-links, then it would return the partial walk till that node.

MOVIE subgraph

Below I am creating a subgraph of the target word 'MOVIE'.

```r
# performing random walk 12 times of step size 12.
walk_rand_movie <- vector(mode = "list")
set.seed(13)
for (i in 1:15){
  walk_rand_movie <- append(walk_rand_movie,
                    as.list(random_walk(g,
                                        # starting random walk from target word.
                                        start= target_words[1],
                                        steps=10,
                                        stuck = "return")))
}
```

I have chosen a threshold of 0.015 for the MOVIE subgraph. There are about 56 edges which have a weight less than 0.015 and the weights of these edges are quite low i.e., around 0.011, 0.013 and thus, they can be removed from the graph. Removing these edges and the vertices which have a degree 0 after removal of edges makes the subgraph more readable.

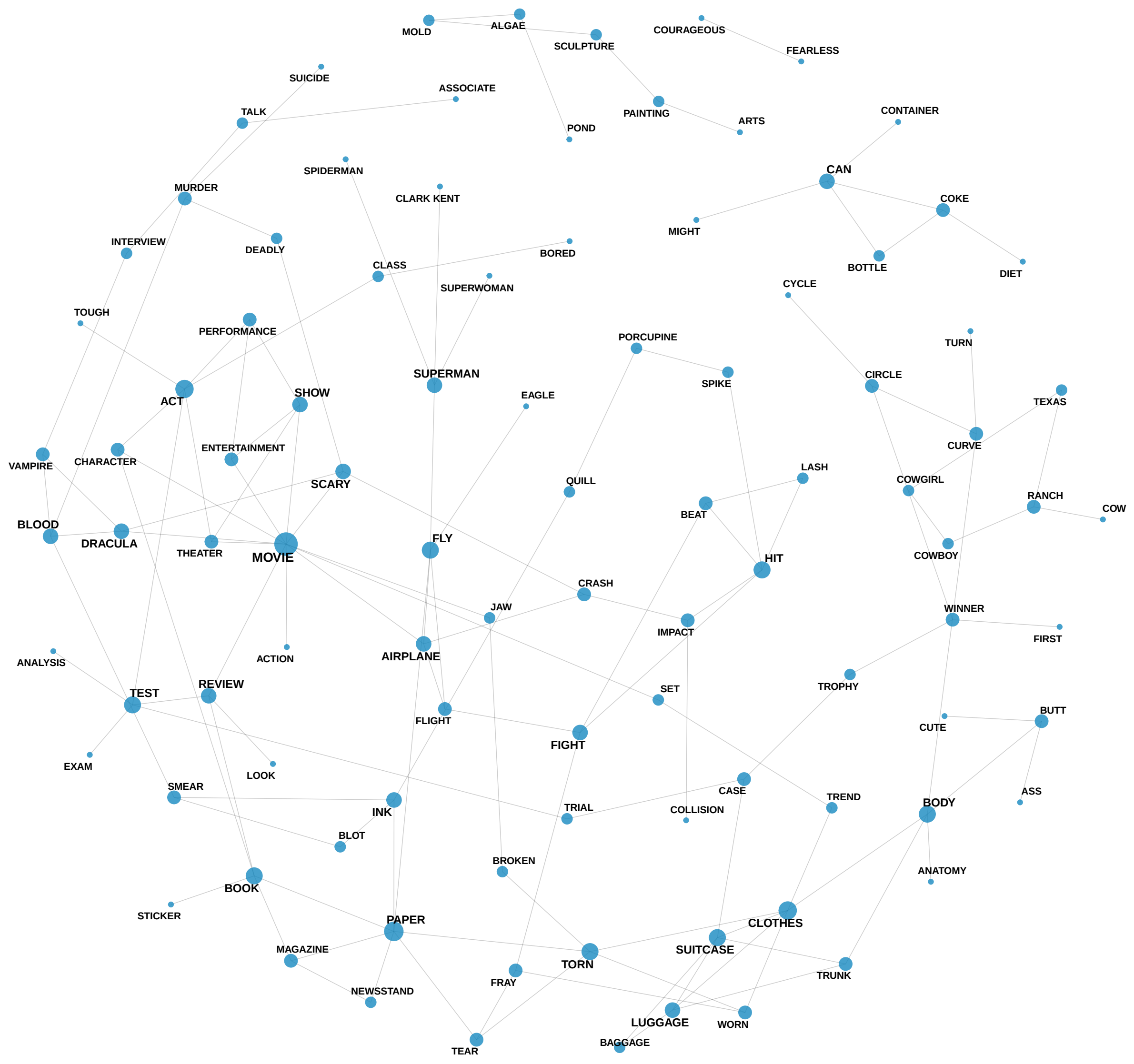
The vertex size and the label size of the vertices are also set based on their degree. First, we find the degree of node and then normalise it.

```r
# creating a subgraph from the nodes found from the random walk.
movie_subgraph <- induced.subgraph(g, walk_rand_movie)
# Removing edges which have very low weight i.e. below 0.002.
movie_subgraph<-delete.edges(movie_subgraph, which(E(movie_subgraph)$weight < 0.015))
# after removing edges, removing vertices which are not connected to any other node.
movie_subgraph<- delete.vertices(movie_subgraph,which(degree(movie_subgraph)==0))
# Plotting the vertices based on their degree.
vertex_size_movie <- 2.5 + degree(movie_subgraph)/1.5
# Plotting the label based on the degree of the vertex.
cex_size_movie <-2 + degree(movie_subgraph)/12

# creating a subgraph using ggraph.
subgraph1 <- ggraph(movie_subgraph, layout = "fr") +
                         # setting the width and alpha value of edge.
                         geom_edge_link(edge_width = 0.2,
                                        alpha = 0.2) +
                         # setting the size of vertex.
                         geom_node_point(aes(size = vertex_size_movie),
                                             alpha = 0.8,
                                             colour = "#188ac1") +
                         # setting the size of label of vertex.
                         geom_node_text(aes(label = name),
                                        size = cex_size_movie,
                                        fontface = "bold",
                                        repel = TRUE) +
                  theme(legend.position = "none",
                        panel.background = element_rect(fill = "white")) +
ggtitle("Figure 1: A subgraph around the word 'MOVIE'")
```

```r
subgraph1
```

Figure 1: A subgraph around the word 'MOVIE'



HOME subgraph

Below I am creating a subgraph of the target word 'HOME'. In this also, I have used a step size of 12 and ran it 12 times. Here, I am using a threshold of 0.0131 as it deletes 32 edges having weights between 0.010 to 0.013. I have used 0.0131 as I wanted to remove edges having weight 0.013 but did not want to remove edges having weight 0.014 because most of the edge weights lie in the range 0.014 to 0.8. The vertex and label size are set based on their degree.

```
# performing random walk 10 times of step size 15.
walk_rand_home <- vector(mode = "list")
set.seed(5)
for (i in 1:12){
  walk_rand_home <- append(walk_rand_home,
```

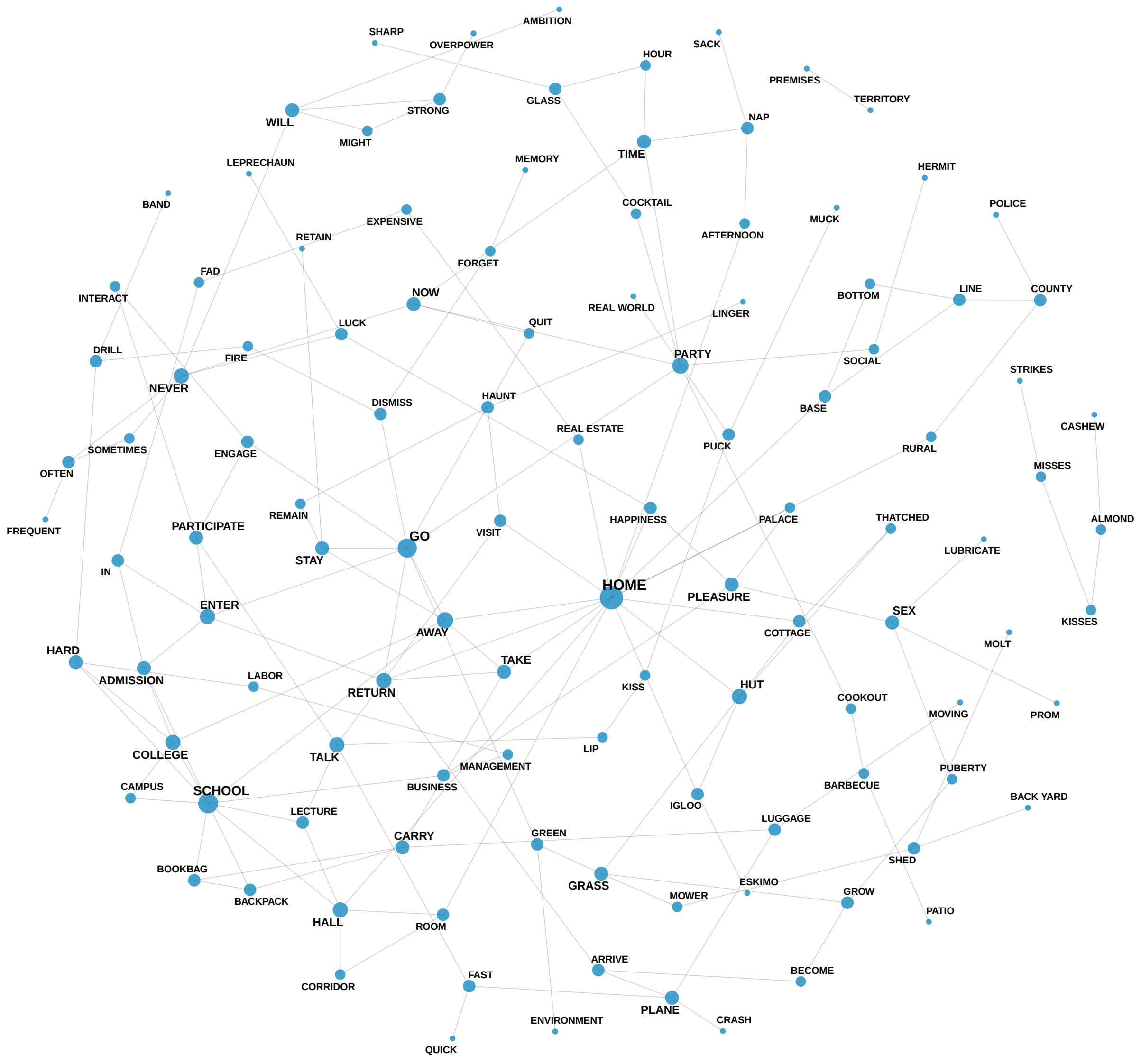
```r
                      # starting random walk from target word.
                      as.list(random_walk(g,
                                          start= target_words[2],
                                          steps=12,
                                          stuck = "return")))
}
```

```r
# creating a subgraph from the nodes found from the random walk.
home_subgraph <- induced.subgraph(g, walk_rand_home)
# Removing edges which have very low weight i.e. below 0.003.
home_subgraph<-delete.edges(home_subgraph, which(E(home_subgraph)$weight < 0.0131))
# after removing edges, removing vertices which are not connected to any other node.
home_subgraph<- delete.vertices(home_subgraph,which(degree(home_subgraph)==0))
# Plotting the vertices based on their degree.
vertex_size_home <- 2.5 + degree(home_subgraph)/1.5
# Plotting the label based on the degree of the vertex.
cex_size_home <-2 + degree(home_subgraph)/12

# creating a subgraph using ggraph.
subgraph2 <- ggraph(home_subgraph, layout = "fr") +
                         # setting the width and alpha value of edge.
                         geom_edge_link(edge_width = 0.2,
                                        alpha = 0.2) +
                         # setting the size of vertex.
                         geom_node_point(aes(size = vertex_size_home),
                                             alpha = 0.8,
                                             colour = "#188ac1") +
                         # setting the size of label of vertex.
                         geom_node_text(aes(label = name),
                                        size = cex_size_home,
                                        fontface = "bold",
                                        repel = TRUE) +
                  theme(legend.position = "none",
                        panel.background = element_rect(fill = "white")) +
ggtitle("Figure 2: A subgraph around the word 'HOME'")
```

```r
subgraph2
```

Figure 2: A subgraph around the word 'HOME'



LIFE subgraph

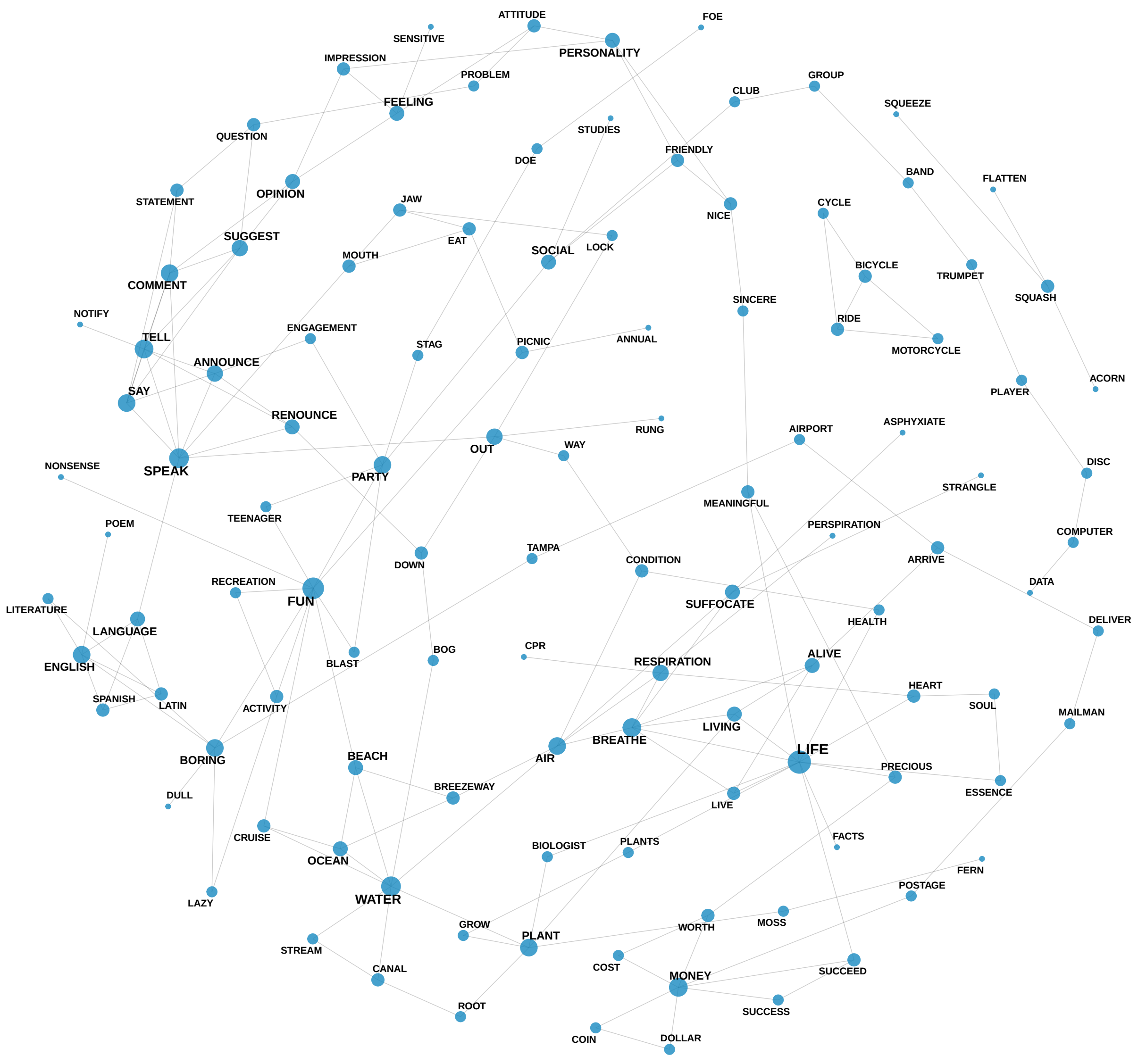
Below, I am creating a subgraph of the target word 'LIFE' using random walk. Here, I have used a step size of 15 as a step size below this value created communities of size 20-21 which is quite large and were semantically heterogeneous. Here, I have used a threshold of 0.0141 as I wanted to remove edges having weight 0.014. There were 27 edges having weight 0.014 and these were the lowest in the subgraph. Hence, I removed them from the subgraph to make the graph more interpretable.

```r
# performing random walk 10 times of step size 15.
set.seed(6)
walk_rand_life <- vector(mode = "list")
for (i in 1:10){
  walk_rand_life <- append(walk_rand_life,
                      # starting random walk from target word.
                      as.list(random_walk(g,
                                          start= target_words[3],
                                          steps=15,
                                          stuck = "return")))
}
```

```
# creating a subgraph from the nodes found from the random walk.
life_subgraph <- induced.subgraph(g, walk_rand_life)
# Removing edges which have very low weight i.e. below 0.001.
life_subgraph<-delete.edges(life_subgraph, which(E(life_subgraph)$weight < 0.0141))
# after removing edges, removing vertices which are not connected to any other node.
life_subgraph<- delete.vertices(life_subgraph,which(degree(life_subgraph)==0))
# Plotting the vertices based on their degree.
vertex_size_life <- 2.5 + degree(life_subgraph)/1.5
# Plotting the label based on the degree of the vertex.
cex_size_life <-2 + degree(life_subgraph)/12
# creating a subgraph using ggraph.
subgraph3 <- ggraph(life_subgraph, layout = "fr") +
                         # setting the width and alpha value of edge.
                         geom_edge_link(edge_width = 0.2,
                                        alpha = 0.2) +
                         # setting the size of vertex.
                         geom_node_point(aes(size = vertex_size_life),
                                             alpha = 0.8,
                                             colour = "#188ac1") +
                         # setting the size of label of vertex.
                         geom_node_text(aes(label = name),
                                        size = cex_size_life,
                                        fontface = "bold",
                                        repel = TRUE) +
                  theme(legend.position = "none",
                        panel.background = element_rect(fill = "white")) +
ggtitle("Figure 3: A subgraph around the word 'LIFE'")
```

```
subgraph3
```

Figure 3: A subgraph around the word 'LIFE'



Creating table of subgraphs

I am creating a table here containing several information of subgraphs like number of nodes, number of edges, average path length, diameter etc. To find average path length, I have used mean_distance() and used attributes directed = FALSE, but since it is already an undirected graph, this will be ignored. I have also set unconnected parameter as TRUE that finds the path length excluding the unconnected graphs. To find an unweighted diamtere, I have set the parameter weights = NA. To find average clustering coefficient, I have set the type to 'localaverage' in transitivity method.

```r
# creating a list of subgraphs to be used for further calculations.
subgraph_list <- list(movie_subgraph, home_subgraph, life_subgraph)
df_subgraph <- data.frame()
for (index in 1:length(subgraph_list)) {
  df_subgraph <- rbind(df_subgraph, c(index, target_words[index],
                     # counting number of vertices.
                     vcount(subgraph_list[[index]]),
                     # counting number of edges.
                     ecount(subgraph_list[[index]]),
                     # finding the average degree of nodes.
                     round(mean(degree(subgraph_list[[index]])),2),
                     # finding unweighted average path length.
                     round(mean_distance(subgraph_list[[index]],directed=FALSE,
                                         unconnected=TRUE), 2),
                     # finding unweighted diameter.
                     diameter(subgraph_list[[index]], weights=NA),
                     # finding average clustering coefficient by using type
                     # localaverage.
                     round(transitivity(subgraph_list[[index]],
                                        type = "localaverage"), 2)))
}
names(df_subgraph) <- c("No.",
                        "Target Word",
                        "Number of nodes",
                        "Number of edges",
                        "Average degree",
                        "Average Path length",
                        "Diameter",
                        "Average Clustering Coefficient")

df_subgraph %>% kbl(align = "llcccccc",
                      caption = "Subgraph of MOVIE, HOME, and LIFE") %>%
  kable_styling(latex_options = c("scale_down", "HOLD_position"))
```

Table 2: Subgraph of MOVIE, HOME, and LIFE

No.	Target Word	Number of nodes	Number of edges	Average degree	Average Path length	Diameter	Average Clustering Coefficient
1	MOVIE	98	128	2.61	4.87	12	0.19
2	HOME	114	156	2.74	4.93	12	0.19
3	LIFE	116	174	3	5.72	16	0.28

Part 2: Community Detection

For community detection, I have tried with three algorithms namely: Girvan Newman, Fast greedy modularity maximisation, and Louvain algorithm.

Girvan Newman Algorithm is based on the heuristic that communities have more links internally and few links netween each other. It recursively removes the weakest link using edge betweeness. The edge that have higher edge betweeness will be weakest link and thus can be removed. This process continues till the network is fully decomposed and finally the partition which is the best is chosen.

Fast greedy method tries to maximise modularity recusively. Modularity is a measure of subgraph being a community. However, this modularity maximisation approach cannot detect communities smaller than the resolution limit. Hence, it's possible that two smaller communities might get merged.

Louvain Algorithm is a clustering approach which first utilises modularity maximisation method to discover small communities and merge them to together to discover larger communities.

In the below code, I have created a generic code to find the communities of the three subgraphs. I have used method edge.betweenness.community(), cluster_fast_greedy(), and cluster_louvain() to implement these algorithms. I have stored the length of community, and modularity for each word in a dataframe and I am appending the dataframe to a final list which will be used further to print the tables. I am simultaneously also finding the words in a community for Louvain algorithm which is the final algorithm that I want to show (details given further). This is also done for all three words and the result is stored in a list to be printed further.

```r
communities_list <- vector(mode = "list")
community_names <- vector(mode = "list")
louvain_community_object <- vector(mode = "list")
edge_bet_community_object <- vector(mode = "list")
greedy_community_object <- vector(mode = "list")

# iterating over the three subgraphs.
set.seed(101)
for (index in 1: length(subgraph_list)) {
  # community detection using Girvan Newman Algorithm.
  community1<- edge.betweenness.community(subgraph_list[[index]])
  df <- data.frame("Girvan-Newman Algorithm",
                       length(community1),
                       round(modularity(community1), 2))
  names(df) <- c("Algorithm name",
                      "Number of communities",
                      "Modularity")

  # community detection using greedy modularity maximisation algorithm.
  community2 <- cluster_fast_greedy(subgraph_list[[index]])
  df <- rbind(df, c("Fast greedy modularity maximisation",
                              length(community2),
                              round(modularity(community2), 2)))

  # community detection using Louvain algorithm.
  community3 <- cluster_louvain(subgraph_list[[index]])
  df <- rbind(df, c("Louvain Algorithm",
                              length(community3),
                              round(modularity(community3), 2)))
  # creating a list of communities for three words to be used further.
  communities_list <- append(communities_list, list(df))

  # finding the words of each community for louvain algorithm.
  comm_df <- data.frame()
  communities <- communities(community3)
  for (commIndex in 1:length(communities)) {
    # storing the words and length of each community for all three words.
    comm_df <- rbind(comm_df,
                       c(commIndex, toString(communities[[commIndex]]),
                         length(communities[[commIndex]])))
  }
  names(comm_df) <- c("number", "cluster", "size")
  community_names <- append(community_names, list(comm_df))

  louvain_community_object <- append(louvain_community_object, list(community3))
  edge_bet_community_object <- append(edge_bet_community_object, list(community1))
  greedy_community_object <- append(greedy_community_object, list(community2))
}
```

After finding the communities in the above code, I intend to find edge density of each community as well to see if the clusters are dense or not.

```r
# this method is taken from the notes to find edge density of each community.
subgraph_density <- function(graph, vertices) {
graph %>%
induced_subgraph(vertices) %>%
edge_density()
}
```

```r
# finding the edge density for the target word MOVIE.
movie_edge_density1 <- c()
movie_comm_len1 <- c()
# finding edge density of clusters formed from edge betweeness algorithm.
for (comm in communities(edge_bet_community_object[[1]])){
  movie_edge_density1 <- c(movie_edge_density1,
                           round(subgraph_density(movie_subgraph, comm), 4))
  movie_comm_len1 <- c(movie_comm_len1, length(comm))
}

movie_edge_density2 <- c()
movie_comm_len2 <- c()
# finding edge density of clusters formed from fast greedy algorithm.
for (comm in communities(greedy_community_object[[1]])){
  movie_edge_density2 <- c(movie_edge_density2,
                           round(subgraph_density(movie_subgraph, comm), 4))
  movie_comm_len2 <- c(movie_comm_len2, length(comm))
}

movie_edge_density3 <- c()
movie_comm_len3 <- c()
# finding edge density of clusters formed from louvain algorithm.
for (comm in communities(louvain_community_object[[1]])){
  movie_edge_density3 <- c(movie_edge_density3,
                           round(subgraph_density(movie_subgraph, comm), 4))
  movie_comm_len3 <- c(movie_comm_len3, length(comm))
}
```

```r
# showing the modularity/edge density of three algorithms on first word.
# adding the edge density calculated above in the data frame.
communities_list[[1]] <- communities_list[[1]]  %>%
  cbind(c(toString(movie_edge_density1),
          toString(movie_edge_density2),
          toString(movie_edge_density3)))
# naming the columns of data frame.
names(communities_list[[1]]) <- c("Algorithm name",
                        "Number of communities",
                        "Modularity", "Edge Density")
# printing the table for MOVIE community.
communities_list[[1]] %>%
  kbl(align = "lccl",
      caption = "Community Detection on subgraph of target word 'MOVIE'") %>%
  kable_styling(latex_options = c("scale_down", "HOLD_position")) %>%
  column_spec(4, width = "30em")
```

Table 3: Community Detection on subgraph of target word 'MOVIE'

Algorithm name	Number of communities	Modularity	Edge Density
Girvan-Newman Algorithm	17	0.58	0.4, 0.4, 0.3214, 0.3333, 0.381, 0.5, 0.3333, 0.4286, 0.5, 0.25, 0.2857, 0.2857, 0.6667, 0.2778, 1, 0.5, 0.8333
Fast greedy modularity maximisation	14	0.84	0.4, 0.1667, 0.2889, 0.4, 0.2857, 0.4286, 0.4, 0.4, 1, 0.2222, 0.3333, 0.381, 0.5, 0.4
Louvain Algorithm	14	0.84	0.4, 0.4, 0.1667, 0.3333, 0.381, 0.381, 0.2222, 0.4, 0.2857, 0.4, 1, 0.5, 0.2889, 0.4

For the word MOVIE, I chose Louvain Algorithm because of the following reasons:

1. Modularity is a measure of how likely a subgraph will be a community. Thus, if higher the modularity, the community structure will be better in network partioning. As, we can see from the table that the modularity of Louvain and fast greedy is better than Girvan Newman. Thus, I first chose them over Girvan Newman.
2. The edge density of all algorithms are almost similar, thus, this does not help in deciding the choice of algorithm.
3. Fast greedy algorithm are biased to merge small communities together. However, louvain first finds small communities and then recursively increase the community size. Thus, the communities are not merged rather a hierarchical structure is created. This difference is more clear on a larger network than in the current graph used.
4. The complexity of fast greedy algorithm is O(n^2) while that of louvain is O(L) which is less than fast greedy and hence louvain is preferred here and for larger networks.

```
# finding the edge density for the target word HOME
home_edge_density1 <- c()
home_comm_len1 <- c()
home_comm_df <- data.frame()
# finding edge density of clusters formed from edge betweeness algorithm.
i = 1
for (comm in communities(edge_bet_community_object[[2]])){
  home_edge_density1 <- c(home_edge_density1,
                          round(subgraph_density(home_subgraph, comm), 4))
  home_comm_len1 <- c(movie_comm_len1, length(comm))
  # storing the words and length of each community for target word home.
  home_comm_df <- rbind(home_comm_df,
                        c(i, toString(comm),
                          length(comm)))
  i = i+1
}

names(home_comm_df) <- c("number", "cluster", "size")

home_edge_density2 <- c()
home_comm_len2 <- c()
```

```r
# finding edge density of clusters formed from fast greedy algorithm.
for (comm in communities(greedy_community_object[[2]])){
  home_edge_density2 <- c(home_edge_density2,
                          round(subgraph_density(home_subgraph, comm), 4))
  home_comm_len2 <- c(movie_comm_len2, length(comm))
}

home_edge_density3 <- c()
home_comm_len3 <- c()
# finding edge density of clusters formed from louvain algorithm.
for (comm in communities(louvain_community_object[[2]])){
  home_edge_density3 <- c(home_edge_density3,
                          round(subgraph_density(home_subgraph, comm), 4))
  home_comm_len3 <- c(movie_comm_len3, length(comm))
}
```

```r
# showing the modularity of three algorithms on second word.
# adding the edge density calculated above in the data frame.
communities_list[[2]] <- communities_list[[2]]  %>%
  cbind(c(toString(home_edge_density1),
          toString(home_edge_density2),
          toString(home_edge_density3)))

# naming the columns of data frame.
names(communities_list[[2]]) <- c("Algorithm name",
                       "Number of communities",
                       "Modularity", "Edge Density")
# printing the table for HOME community.
communities_list[[2]] %>%
  kbl(align = "lccl",
      caption = "Community Detection on subgraph of target word 'HOME'") %>%
  kable_styling(latex_options = c("scale_down", "HOLD_position")) %>%
  column_spec(4, width = "30em")
```

Table 4: Community Detection on subgraph of target word 'HOME'

Algorithm name	Number of communities	Modularity	Edge Density
Girvan-Newman Algorithm	18	0.74	0.3333, 0.3929, 0.5, 0.3111, 0.25, 0.3333, 0.4, 0.2857, 0.381, 0.3333, 0.2222, 0.5, 0.2857, 0.3333, 0.4, 0.3333, 1, 0.5
Fast greedy modularity maximisation	17	0.83	0.1978, 0.2, 0.1818, 0.2857, 0.3333, 0.4, 0.5, 0.4, 0.2, 0.2857, 0.4, 0.6667, 0.4, 0.3929, 0.5, 1, 1
Louvain Algorithm	17	0.83	0.2, 0.2051, 0.5, 0.3929, 1, 0.1818, 0.4, 0.4, 0.2857, 0.3214, 0.2857, 0.5, 0.2, 0.4, 0.4, 0.6667, 1

For the word HOME, I chose Girvan Newman Algorithm because of the following reasons:

1. We can see from the table that the modularity of all three is not much different. While the modularity of Girvan is slightly less than the other two algorithms but it is still high.
2. We can see that edge density of all three algorithms is similar hence, do not help in deciding.
3. I also calculated the size of each community for each algorithm. I found that Girvan Newman had an ideal community size with an exception on one which had a community size of 2. Louvain and Fast greedy had a community size of 16 which will be semantically heterogeneous and 2 communities of size 2 as well which is too small.

```r
print(home_comm_len1)
```

```
##  [1] 5 6 8 6 7 4 6 7 4 8 7 7 3 9 2 5 4 5
```

```r
print(home_comm_len2)
```

```
##  [1]  5 16 10  6  8  7  6  6  2  9  6  7  5  5  2
```

```r
print(home_comm_len3)
```

```
##  [1]  5  6 16  6  7  7  9  5  8  6  2  5 10  6  2
```

```r
life_edge_density1 <- c()
life_comm_len1 <- c()
# finding the edge density for the target word LIFE
# finding edge density of clusters formed from edge betweeness algorithm.
for (comm in communities(edge_bet_community_object[[3]])){
  life_edge_density1 <- c(life_edge_density1,
                          round(subgraph_density(life_subgraph, comm), 4))
  life_comm_len1 <- c(life_comm_len1, length(comm))
}

life_edge_density2 <- c()
life_comm_len2 <- c()
# finding edge density of clusters formed from fast greedy algorithm.
for (comm in communities(greedy_community_object[[3]])){
  life_edge_density2 <- c(life_edge_density2,
                          round(subgraph_density(life_subgraph, comm), 4))
  life_comm_len2 <- c(life_comm_len2, length(comm))
}

life_edge_density3 <- c()
life_comm_len3 <- c()
# finding edge density of clusters formed from louvain algorithm.
for (comm in communities(louvain_community_object[[3]])){
  life_edge_density3 <- c(life_edge_density3,
                          round(subgraph_density(life_subgraph, comm), 4))
  life_comm_len3 <- c(life_comm_len3, length(comm))
}
```

```r
# showing the modularity of three algorithms on third word.
# adding the edge density calculated above in the data frame.
communities_list[[3]] <- communities_list[[3]]  %>%
  cbind(c(toString(life_edge_density1),
          toString(life_edge_density2),
          toString(life_edge_density3)))

# naming the columns of data frame.
names(communities_list[[3]]) <- c("Algorithm name",
                       "Number of communities",
                       "Modularity", "Edge Density")
```

```r
# printing the table for LIFE community.
communities_list[[3]] %>%
  kbl(align = "lccl",
      caption = "Community Detection on subgraph of target word 'LIFE'") %>%
  kable_styling(latex_options = c("scale_down", "HOLD_position")) %>%
  column_spec(4, width = "30em")
```

Table 5: Community Detection on subgraph of target word 'LIFE'

Algorithm name	Number of communities	Modularity	Edge Density
Girvan-Newman Algorithm	7	0.66	0.1255, 0.1619, 0.081, 0.1542, 0.5, 0.1667, 0.8333
Fast greedy modularity maximisation	14	0.83	0.1648, 0.4, 0.2121, 0.2182, 0.2051, 0.5, 0.2444, 0.4, 0.6667, 0.25, 0.3485, 0.3571, 0.5333, 0.8333
Louvain Algorithm	14	0.83	0.2121, 0.3571, 0.2051, 0.2444, 0.3485, 0.5, 0.2182, 0.4, 0.6667, 0.1648, 0.5333, 0.25, 0.8333, 0.4

For the word LIFE, I chose Louvain Algorithm because of the following reasons:

1. We can see from the table that the modularity of Louvain and fast greedy is better than Girvan Newman and hence it will partition the graph well. Thus, I first chose them over Girvan Newman.
2. We can see that edge density of Louvain is slightly better than fast greedy. We can see where fast greedy had edge density of 0.2051, louvain had an edge density of 0.3485. A greater edge density means that the communities are much dense and hence better.
3. I also calculated the size of each community for each algorithm. I found that Girvan Newman had a community of size 22 which can be semantically heterogeneous. Thus I chose Louvain as it didn't have any community of size greater than 14.

```r
print(life_comm_len1)
```

```
## [1] 22 15 36 23  4 12  4
```

```r
print(life_comm_len2)
```

```
##  [1] 14  5 12 11 13  4 10  6  3  8 12  8  6  4
```

```r
print(life_comm_len3)
```

```
##  [1] 12  8 13 10 12  4 11  5  3 14  6  8  4  6
```

4. Fast greedy algorithm are biased to merge small communities together. However, louvain first finds small communities and then recursively increase the community size. Thus, the communities are not merged rather a hierarchical structure is created. This difference is more clear on a larger network than in the current graph used.
5. The complexity of fast greedy algorithm is O(n^2) while that of louvain is O(L) which is less than fast greedy and hence louvain is preferred for larger networks.

I also tried using walktrap but gave very large communities of around 25 words which is too big and hence may have heterogeneous meaning. Thus, i discarded using walktrap for our use-case as it is only preferred for a large network.

```r
# creating interpretations
movie_inter <- c("related to physical looks of actors in a movie",
                 "related to refreshments required during a movie",
                 "related to types of movie like an action/horror film.
                 It is also related to the background set of a movie like a theatre",
                 "related to creative aspects that can be used in a movie",
                 "related to any fighting sequence of a movie",
                 "related to any article published about a movie in a magazine or any other place",
                 "related to if the movie won any awards during a show",
                 "related to any review or critics of a movie",
                 "related to fictional movies like superman",
                 "related to deadly creatures like vampire",
                 "related to movie made on brave people like Mahatma Gandhi",
                 "related to movies made for texas side",
                 "related to the attire required by actors during a movie",
                 "related to any dark stain made on paper")

# adding interpretations to the data frame
community_names[[1]] <- cbind(community_names[[1]], movie_inter)
names(community_names[[1]]) <- c("number", "cluster", "size", "interpretation")

# showing the words and size of each community for first target word.
community_names[[1]]  %>%
  kbl(align = "llcl",
      caption = "Results of Community Detection on subgraph of target word 'MOVIE'") %>%
  kable_styling(latex_options = c("scale_down", "HOLD_position")) %>%
  column_spec(2, width = "20em") %>%
  column_spec(4, width = "20em")
```

Table 6: Results of Community Detection on subgraph of target word ’MOVIE’

number	cluster	size	interpretation
1	BODY, CUTE, ANATOMY, ASS, BUTT	5	related to physical looks of actors in a movie
2	CAN, BOTTLE, CONTAINER, COKE, DIET, MIGHT	6	related to refreshments required during a movie
3	CLASS, SHOW, ACT, THEATER, TOUGH, ACTION, MOVIE, SCARY, PERFORMANCE, SUICIDE, MURDER, CHARACTER, BORED, DEADLY, ENTERTAINMENT, JAW	16	related to types of movie like an action/horror film. It is also related to the background set of a movie like a theatre
4	PAINTING, ALGAE, POND, MOLD, SCULPTURE, ARTS	6	related to creative aspects that can be used in a movie
5	HIT, BEAT, FIGHT, CRASH, IMPACT, COLLISION, LASH	7	related to any fighting sequence of a movie
6	BOOK, PAPER, MAGAZINE, TEAR, STICKER, FRAY, NEWSSTAND	7	related to any article published about a movie in a magazine or any other place
7	TRIAL, WINNER, FIRST, CURVE, TURN, CIRCLE, TROPHY, CASE, CYCLE	9	related to if the movie won any awards during a show
8	TEST, LOOK, ANALYSIS, EXAM, REVIEW	5	related to any review or critics of a movie
9	FLY, AIRPLANE, FLIGHT, EAGLE, SUPERMAN, SUPERWOMAN, CLARK KENT, SPIDERMAN	8	related to fictional movies like superman
10	TALK, BLOOD, INTERVIEW, ASSOCIATE, VAMPIRE, DRACULA	6	related to deadly creatures like vampire
11	COURAGEOUS, FEARLESS	2	related to movie made on brave people like Mahatma Gandhi
12	COW, COWBOY, RANCH, COWGIRL, TEXAS	5	related to movies made for texas side
13	CLOTHES, SET, LUGGAGE, BAGGAGE, SUITCASE, BROKEN, TORN, TRUNK, TREND, WORN	10	related to the attire required by actors during a movie
14	SMEAR, BLOT, INK, PORCUPINE, SPIKE, QUILL	6	related to any dark stain made on paper

```
home_inter <- c("related to teenage years of people",
                "related to stay at a place and interacting with people",
                "related to people who are strong mentally",
                "related to schools/universities and it's accessories",
                "related to emotions like a kiss in a home",
                "related to flights and baggage which people use while coming home",
                "related to types of cities where people live like a county ",
                "related to backyard and it's accessories at a home",
                "related to different types of rooms/place in a home like for fun activity/to study etc
                "related to any forgotten aspect that happened in a home like fire",
                "related to time aspect of a day like afternoon",
                "related to the pricing of home",
                "related to any fun day in a home involving food and drinks",
                "related to things that happen in varied amount like sometimes/frequently",
                "related to lady of the house and dry fruits",
                "related to people who do not leave the house",
                "an area belonging to a particular person",
                "types of home like a cottage in a village, an igloo in a cold place")
```

```r
# adding interpretations to the data frame
home_comm_df <- cbind(home_comm_df, home_inter)
names(home_comm_df) <- c("number", "cluster", "size", "interpretation")

# showing the words and size of each community for second target word.
home_comm_df  %>%
  kbl(align = "llcl",
      caption = "Results of Community Detection on subgraph of target word 'HOME'") %>%
  kable_styling(latex_options = c("scale_down", "HOLD_position")) %>%
  column_spec(2, width = "20em") %>%
  column_spec(4, width = "20em")
```

Table 7: Results of Community Detection on subgraph of target word 'HOME'

number	cluster	size	interpretation
1	SEX, GROW, PUBERTY, BECOME, PROM, LUBRICATE	6	related to teenage years of people
2	TAKE, AWAY, GO, ENTER, PARTICIPATE, RETURN, ENGAGE, INTERACT	8	related to stay at a place and interacting with people
3	STRONG, WILL, AMBITION, OVERPOWER, MIGHT	5	related to people who are strong mentally
4	SCHOOL, COLLEGE, BUSINESS, ADMISSION, HARD, BACKPACK, BOOKBAG, CAMPUS, LABOR, MANAGEMENT	10	related to schools/universities and it's accessories
5	FAST, TALK, KISS, QUICK, LIP, MUCK, PUCK, REAL WORLD	8	related to emotions like a kiss in a home
6	CRASH, MOVING, PLANE, ARRIVE, CARRY, LUGGAGE	6	related to flights and baggage which people use while coming home
7	POLICE, BASE, LINE, BOTTOM, COUNTY, RURAL	6	related to types of cities where people live like a county
8	GREEN, GRASS, ENVIRONMENT, SHED, MOWER, BACK YARD, MOLT	7	related to backyard and it's accessories at a home
9	HOME, LECTURE, CORRIDOR, HALL, PLEASURE, ROOM, PALACE	7	related to different types of rooms/place in a home like for fun activity/to study etc.
10	DISMISS, FIRE, BAND, FORGET, MEMORY, DRILL	6	related to any forgotten aspect that happened in a home like fire
11	QUIT, NOW, AFTERNOON, NAP, TIME, SHARP, SACK, GLASS, HOUR	9	related to time aspect of a day like afternoon
12	IN, REAL ESTATE, EXPENSIVE, FAD	4	related to the pricing of home
13	PARTY, BARBECUE, PATIO, COOKOUT, SOCIAL, COCKTAIL, HERMIT	7	related to any fun day in a home involving food and drinks
14	NEVER, SOMETIMES, HAPPINESS, LUCK, OFTEN, LEPRECHAUN, FREQUENT	7	related to things that happen in varied amount like sometimes/frequently
15	ALMOND, KISSES, CASHEW, MISSES, STRIKES	5	related to lady of the house and dry fruits
16	STAY, REMAIN, HAUNT, VISIT, LINGER, RETAIN	6	related to people who do not leave the house
17	TERRITORY, PREMISES	2	an area belonging to a particular person
18	COTTAGE, HUT, THATCHED, IGLOO, ESKIMO	5	types of home like a cottage in a village, an igloo in a cold place

```r
# showing the words and size of each community for third target word.
life_inter <- c("related to aspects in a teenagers life like a party, study, any recreational activity"
                "related to financial aspect of life involving money",
                "related to environment specifically aquatic one",
                "related to different types of emotions and nature of people",
                "related to aspects involving a speech or verbal communication",
                "related to squuezing something like a can till it becomes flat",
                "related to conditions occuring due to presence or absence of air like breathing",
                "related to a muddy area requiring support for people to walk",
                "related to people who lack energy to do something exciting",
                "related to aspects that show that people are alive",
                "related to different languages spoken by people",
                "related to different things that are people's hobby",
                "using two wheeler for transport",
                "related to food and eating aspects of people")

# adding interpretations to the data frame
community_names[[3]] <- cbind(community_names[[3]], life_inter)
names(community_names[[3]]) <- c("number", "cluster", "size", "interpretation")

community_names[[3]]  %>%
  kbl(align = "llcl",
      caption = "Results of Community Detection on subgraph of target word 'LIFE'") %>%
  kable_styling(latex_options = c("scale_down", "HOLD_position")) %>%
  column_spec(2, width = "20em") %>%
  column_spec(4, width = "20em")
```

Table 8: Results of Community Detection on subgraph of target word 'LIFE'

number	cluster	size	interpretation
1	NONSENSE, ACTIVITY, FUN, TEENAGER, FOE, PARTY, SOCIAL, BLAST, DOE, RECREATION, STUDIES, STAG	12	related to aspects in a teenagers life like a party, study, any recreational activity
2	MONEY, SUCCEED, SUCCESS, DOLLAR, COST, WORTH, COIN, POSTAGE	8	related to financial aspect of life involving money
3	WATER, GROW, PLANT, MOSS, OCEAN, BEACH, BIOLOGIST, STREAM, CANAL, ROOT, CRUISE, PLANTS, FERN	13	related to environment specifically aquatic one
4	PROBLEM, FEELING, NICE, SINCERE, ATTITUDE, PERSONALITY, FRIENDLY, SENSITIVE, IMPRESSION, MEANINGFUL	10	related to different types of emotions and nature of people
5	TELL, SPEAK, SAY, OPINION, COMMENT, SUGGEST, NOTIFY, QUESTION, ANNOUNCE, ENGAGEMENT, STATEMENT, RENOUNCE	12	related to aspects involving a speech or verbal communication
6	ACORN, SQUASH, SQUEEZE, FLATTEN	4	related to squuezing something like a can till it becomes flat
7	AIR, HEALTH, BREATHE, CONDITION, RESPIRATION, BREEZEWAY, STRANGLE, SUFFOCATE, CPR, PERSPIRATION, ASPHYXIATE	11	related to conditions occuring due to presence or absence of air like breathing
8	OUT, WAY, DOWN, BOG, RUNG	5	related to a muddy area requiring support for people to walk
9	LAZY, BORING, DULL	3	related to people who lack energy to do something exciting
10	ALIVE, LIVE, LIFE, AIRPORT, LIVING, FACTS, HEART, ARRIVE, TAMPA, SOUL, DELIVER, MAILMAN, ESSENCE, PRECIOUS	14	related to aspects that show that people are alive
11	ENGLISH, SPANISH, LANGUAGE, LITERATURE, LATIN, POEM	6	related to different languages spoken by people
12	GROUP, BAND, DATA, COMPUTER, PLAYER, CLUB, TRUMPET, DISC	8	related to different things that are people's hobby
13	RIDE, BICYCLE, MOTORCYCLE, CYCLE	4	using two wheeler for transport
14	ANNUAL, PICNIC, EAT, LOCK, MOUTH, JAW	6	related to food and eating aspects of people

Part 3: Visualizing Community

For visualizing, I have used custom pallete from DV assignment and added 7 more colors to it and reduced their alpha.

```
library(RColorBrewer)
library(colorblindr)
# using the custom color palette created in DV assignment.
colours <- c('#CF4994', '#FA0A00', '#FFE924','#45403D' , '#FA114D','#009E73' ,
             '#6FDB0F','#008F12', '#00DBCD', '#DB9319')
# adding more colors to the custom pallete.
colours <- c(colours, brewer.pal(10, "Paired"))
```

```r
# function to reduce alpha values of colors.
# this has been taken from notes.
add.alpha <- function(cols, alpha) rgb(t(col2rgb(cols)/255), alpha = alpha)
# calling the function to reduce the alpha value for each color.
colours<-add.alpha(colours, 0.7)
```

To plot the graph, I have kept the size of nodes based on their degree and size of label as well based on their degree. To color nodes according to community, I have set colour parameter in geom_node_point() as a factor of communities. To set manual colors to the graph, I have used scale_color_manual() and scale_edge_color_manual().

To find the most central node in the community, I found the node in each community with highest degree. Any centrality measure could have been used.

To color edges, I assigned community to edges as well. The logic is if both the vertices of edge belong to same community then edge will be assigned the same community. However, if vertices belong to different community, then they will not be assigned any community (for this case a value 9999) and will be colored grey.

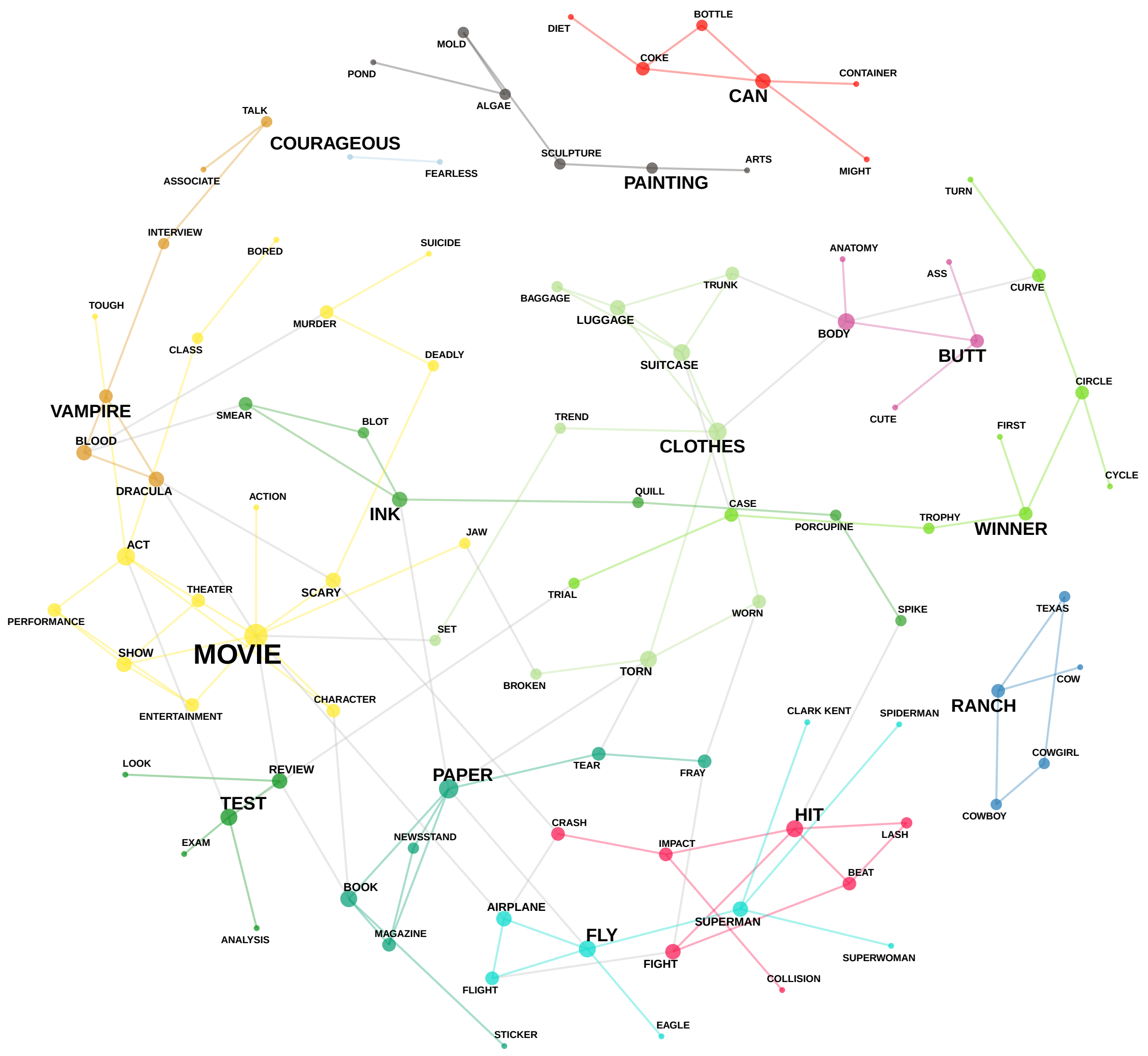
Movie Subgraph

```r
# finding the central node in each community and increasing their font-size.
# iterating over each community of first word.
for (comm in communities(louvain_community_object[[1]])){
  # creating a subgraph of each community.
  comm_subg_movie <- induced.subgraph(movie_subgraph, comm)
  # finding the node with maximum degree in the community.
  # a node with maximum degree in a community will be the most central node
  # as it will be connected to many other nodes in the community.
  max_degree_movie <- V(comm_subg_movie)[which(degree(comm_subg_movie) == max(degree(comm_subg_movie)))]
  # setting the label size 4 of central nodes in the community.
  cex_size_movie[which(names(cex_size_movie) == max_degree_movie[1]$name)] = 4
  # setting the label size 6 of the target node.
  cex_size_movie[which(names(cex_size_movie) == target_words[1])] = 6
}

# to color edges, assigning each edge to community.
# if both the vertices of edge is in the same community then the edge will have the same community.
# if the vertices are in different community then they do not belong to any community.
V(movie_subgraph)$color <- louvain_community_object[[1]]$membership
# getting all the edges from the subgraph.
edges <- as.data.frame(get.edgelist(movie_subgraph))
# for each edge, check if the community of vertex 1 is equal to vertex 2.
# if they are equal assign the first vertex community to the edge.
E(movie_subgraph)$color <- apply(edges, 1,
  function(x)
    ifelse(V(movie_subgraph)$color[
      V(movie_subgraph)[x[1]]] == V(movie_subgraph)$color[V(movie_subgraph)[x[2]]],
  # if the vertex are not in same community assign 9999 to the community of edge.
  V(movie_subgraph)$color[V(movie_subgraph)[x[1]]], 9999))

# plotting the graph with communities colored.
ggraph(movie_subgraph, layout = "fr") +
  # if the edges are assigned a community then they will be colored community wise.
  geom_edge_link0(aes(filter=color!=9999 ,color=as.factor(color)), width=0.6, alpha=0.35) +
  # if edges do not belong to any community then they will be colored grey.
  geom_edge_link0(aes(filter=color==9999), color='grey', width=0.6, alpha=0.35) +
  # nodes will be colored based on community.
  geom_node_point(aes(size = vertex_size_movie,
                      colour=factor(louvain_community_object[[1]]$membership))) +
  # setting the size of label.
  geom_node_text(aes(label = name),
                 size = cex_size_movie,
                 fontface = "bold",
                 repel = TRUE) +
  # using custom pallete to color vertices and edge of graph.
  scale_color_manual(values = colours) +
  scale_edge_colour_manual(values = colours) +
  theme(legend.position = "none",
        panel.background = element_rect(fill = "white")) +
  ggtitle("Figure 4: A subgraph around the word 'MOVIE' with the communities indicated by color")
```

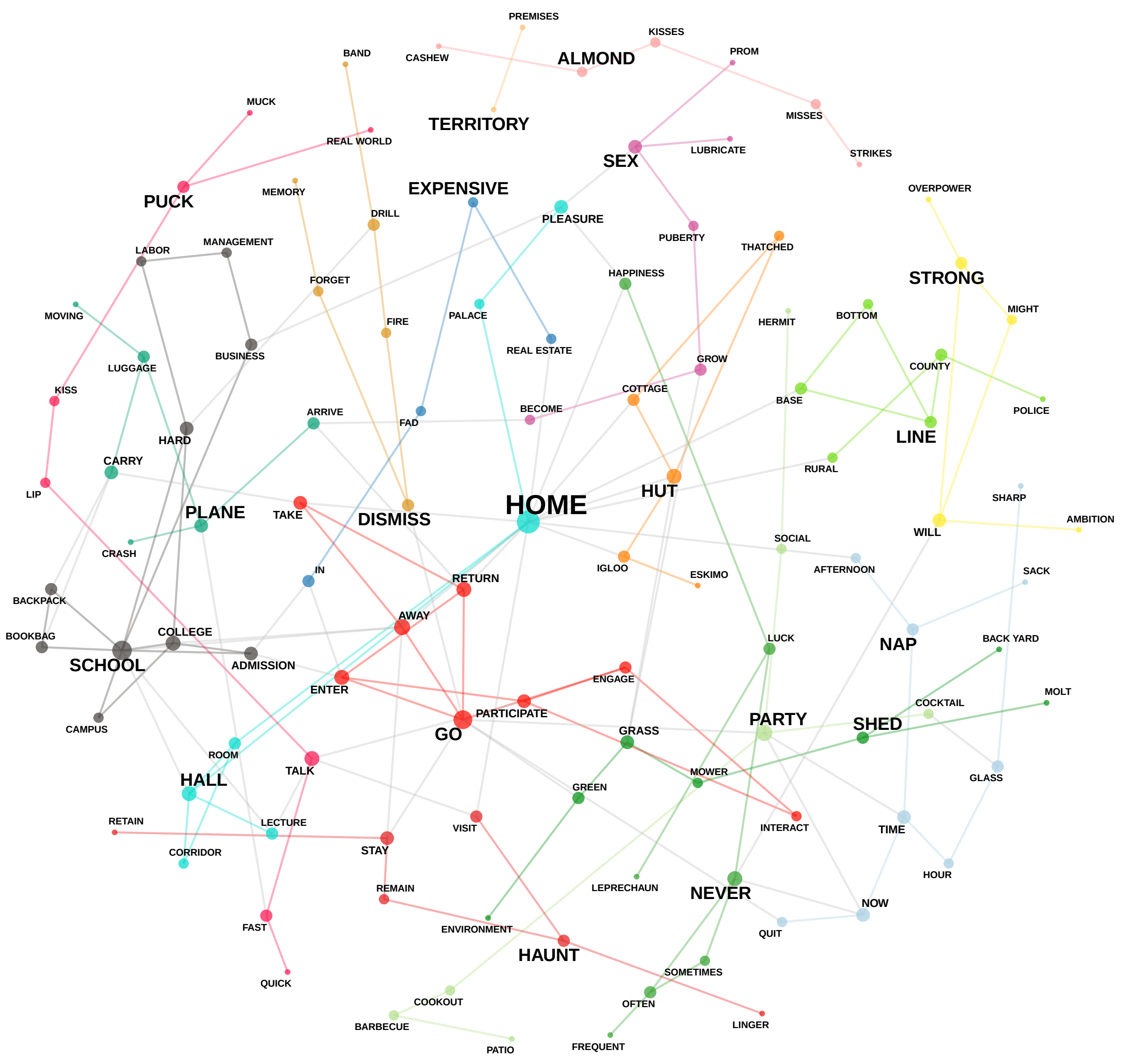
Figure 4: A subgraph around the word 'MOVIE' with the communities indicated by color



```r
# finding the central node in each community and increasing their font-size.
# iterating over each community of second word.
for (comm in communities(edge_bet_community_object[[2]])){
  # creating a subgraph of each community.
  comm_subg_home <- induced.subgraph(home_subgraph, comm)
  # finding the node with maximum degree in the community.
  # a node with maximum degree in a community will be the most central node
  # as it will be connected to many other nodes in the community.
  max_degree_home <- V(comm_subg_home)[which(degree(comm_subg_home) == max(degree(comm_subg_home)))]
  cex_size_home[which(names(cex_size_home) == max_degree_home[1]$name)] = 4
  cex_size_home[which(names(cex_size_home) == target_words[2])] = 6
}

# to color edges, assigning each edge to community.
# if both the vertices of edge is in the same community then the edge will have the same community.
# if the vertices are in different community then they do not belong to any community.
V(home_subgraph)$color <- edge_bet_community_object[[2]]$membership
# getting all the edges from the subgraph.
edges <- as.data.frame(get.edgelist(home_subgraph))
# for each edge, check if the community of vertex 1 is equal to vertex 2.
# if they are equal assign the first vertex community to the edge.
E(home_subgraph)$color <- apply(edges, 1,
  function(x)
    ifelse(V(home_subgraph)$color[
      V(home_subgraph)[x[1]]] == V(home_subgraph)$color[V(home_subgraph)[x[2]]],
  # if the vertex are not in same community assign 9999 to the community of edge.
  V(home_subgraph)$color[V(home_subgraph)[x[1]]], 9999))
# plotting the graph with communities colored.
ggraph(home_subgraph, layout = "fr") +
  # if the edges are assigned a community then they will be colored community wise.
  geom_edge_link0(aes(filter=color!=9999 ,color=as.factor(color)), width=0.6, alpha=0.35) +
  # if edges do not belong to any community then they will be colored grey.
  geom_edge_link0(aes(filter=color==9999), color='grey', width=0.6, alpha=0.35) +
  # nodes will be colored based on community.
  geom_node_point(aes(size = vertex_size_home,
                      colour=factor(edge_bet_community_object[[2]]$membership))) +
  # setting the size of label.
  geom_node_text(aes(label = name),
                 size = cex_size_home,
                 fontface = "bold",
                 repel = TRUE) +
  # using custom pallete to color vertices and edge of graph.
  scale_color_manual(values = colours) +
  scale_edge_colour_manual(values = colours) +
  theme(legend.position = "none",
        panel.background = element_rect(fill = "white")) +
  ggtitle("Figure 5: A subgraph around the word 'HOME' with the communities indicated by color")
```

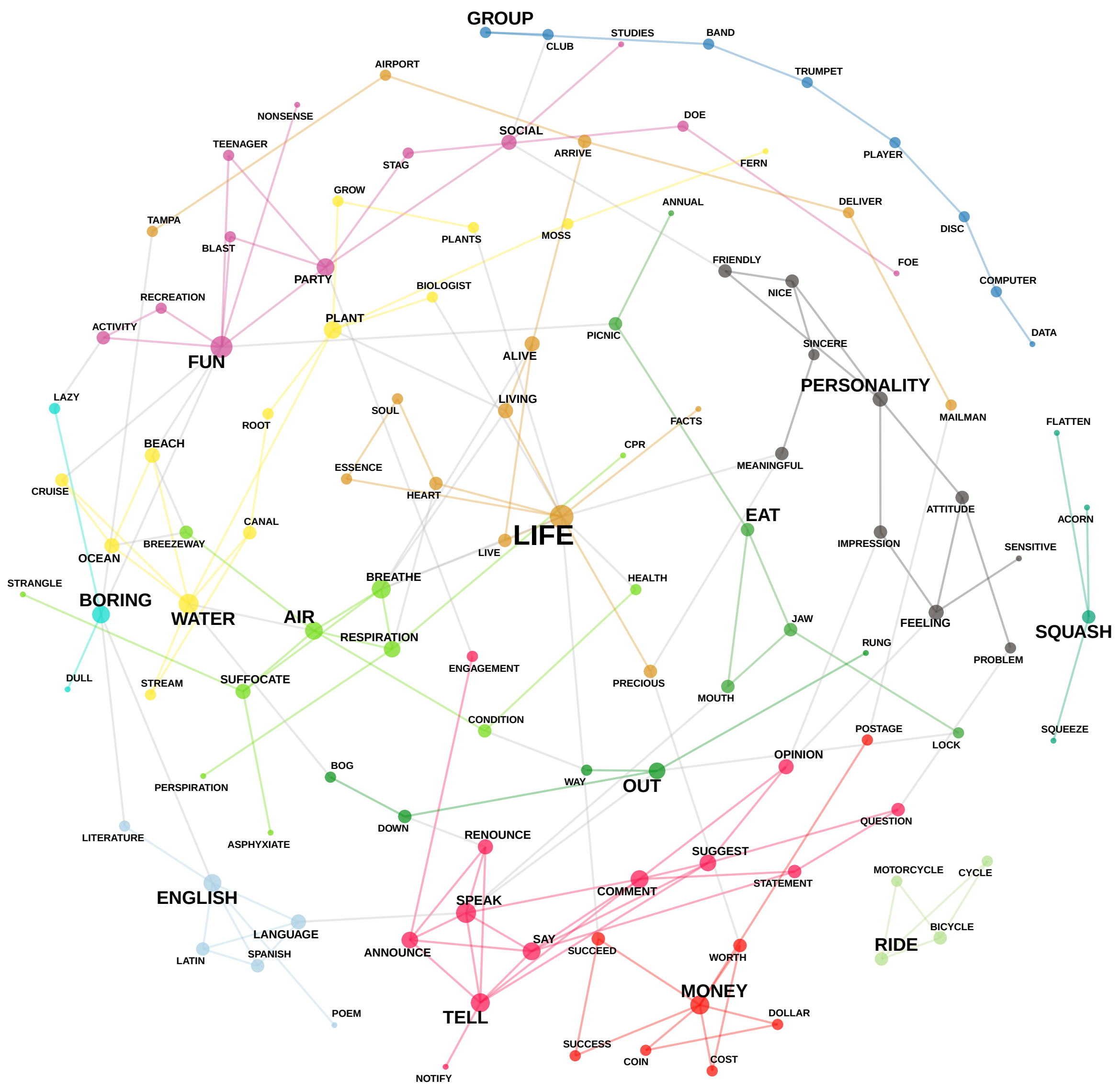
Figure 5: A subgraph around the word 'HOME' with the communities indicated by color



```r
# finding the central node in each community and increasing their font-size.
# iterating over each community of third word.
for (comm in communities(louvain_community_object[[3]])){
  # creating a subgraph of each community.
  comm_subg_life <- induced.subgraph(life_subgraph, comm)
  # finding the node with maximum degree in the community.
  # a node with maximum degree in a community will be the most central node
  # as it will be connected to many other nodes in the community.
  max_degree_life <- V(comm_subg_life)[which(degree(comm_subg_life) == max(degree(comm_subg_life)))]
  cex_size_life[which(names(cex_size_life) == max_degree_life[1]$name)] = 4
  cex_size_life[which(names(cex_size_life) == target_words[3])] = 6
}

# to color edges, assigning each edge to community.
# if both the vertices of edge is in the same community then the edge will have the same community.
# if the vertices are in different community then they do not belong to any community.
V(life_subgraph)$color <- louvain_community_object[[3]]$membership
# getting all the edges from the subgraph.
edges <- as.data.frame(get.edgelist(life_subgraph))
# for each edge, check if the community of vertex 1 is equal to vertex 2.
# if they are equal assign the first vertex community to the edge.
E(life_subgraph)$color <- apply(edges, 1,
        function(x)
          ifelse(V(life_subgraph)$color[
            V(life_subgraph)[x[1]]] == V(life_subgraph)$color[V(life_subgraph)[x[2]]],
  # if the vertex are not in same community assign 9999 to the community of edge.
  V(life_subgraph)$color[V(life_subgraph)[x[1]]], 9999))
# plotting the graph with communities colored.
ggraph(life_subgraph, layout = "fr") +
  # if the edges are assigned a community then they will be colored community wise.
  geom_edge_link0(aes(filter=color!=9999 ,color=as.factor(color)), width=0.6, alpha=0.35) +
  # if edges do not belong to any community then they will be colored grey.
  geom_edge_link0(aes(filter=color==9999), color='grey', width=0.6, alpha=0.35) +
  # nodes will be colored based on community.
  geom_node_point(aes(size = vertex_size_life,
                      colour=factor(louvain_community_object[[3]]$membership))) +
  # setting the size of label.
  geom_node_text(aes(label = name),
                 size = cex_size_life,
                 fontface = "bold",
                 repel = TRUE) +
  # using custom pallete to color vertices and edge of graph.
  scale_color_manual(values = colours) +
  scale_edge_colour_manual(values = colours) +
  theme(legend.position = "none",
        panel.background = element_rect(fill = "white")) +
  ggtitle("Figure 6: A subgraph around the word 'LIFE' with the communities indicated by color")
```

Figure 6: A subgraph around the word 'LIFE' with the communities indicated by color


GROUP
CLUB
STUDIES
BAND
TRUMPET
AIRPORT
NONSENSE
TEENAGER
SOCIAL
ARRIVE
DOE
STAG
FERN
PLAYER
GROW
DELIVER
TAMPA
ANNUAL
PLANTS
MOSS
DISC
BLAST
PARTY
FRIENDLY
FOE
BIOLOGIST
NICE
COMPUTER
RECREATION
PLANT
ACTIVITY
SINCERE
DATA
FUN
ALIVE
PICNIC
PERSONALITY
LAZY
SOUL
LIVING
MAILMAN
FACTS
FLATTEN
ROOT
BEACH
CPR
ESSENCE
MEANINGFUL
CRUISE
HEART
CANAL
EAT
ATTITUDE
ACORN
BREEZEWAY
LIFE
IMPRESSION
OCEAN
LIVE
SENSITIVE
STRANGLE
BREATHE
HEALTH
BORING
JAW
FEELING
AIR
WATER
SQUASH
RESPIRATION
RUNG
ENGAGEMENT
PROBLEM
DULL
STREAM
SUFFOCATE
PRECIOUS
MOUTH
CONDITION
POSTAGE
SQUEEZE
LOCK
OPINION
BOG
WAY
OUT
PERSPIRATION
QUESTION
LITERATURE
DOWN
RENOUNCE
ASPHYXIATE
SUGGEST
MOTORCYCLE
CYCLE
COMMENT
STATEMENT
ENGLISH
SPEAK
BICYCLE
LANGUAGE
SAY
RIDE
ANNOUNCE
SUCCEED
LATIN
SPANISH
WORTH
MONEY
POEM
TELL
DOLLAR
SUCCESS
COIN
COST
NOTIFY