32439180 Looi Teck San FIT3152 Assignment 3

Question 1

I had decided to collect 15 documents based on 5 different topics which included AI, car, dog, scuba diving and also movie

Question 2

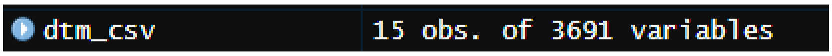
First, I copy and paste the text into words. After that, I save each file into txt format and name each document including its topic, for example, the text talking about Ferrari, I would name the file as car_ferrari.txt, text talking about Chatgpt, I would name the file as ai_chatgpt.txt etc, where car and ai are their respective topic, so that it is easier for me to recognize each document later on.

After that I create the corpus using the methods covered in lectures and tutorials.

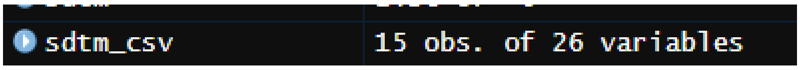
```
cname = file.path(".","CorpusAbstracts","txt")

cname

dir(cname)

# create corpus
docs = Corpus(DirSource((cname)))
summary(docs)
```

Question 3

For my approach, before removing the sparse terms, I started some pre-process like remove all the numbers, remove all the punctuations, transform each term in each document to lower case, remove stop words, remove white space and finally stemming all the terms. After doing all these process, I created the Document Term Matrix(DTM) and I ended up with more than 3000 terms.



Therefore, I decided to remove 45% empty terms, which means any term that appears in less than 45% of the corpus will be removed. After removing the sparse terms, I ended up with 26 terms to be analyse. Also, based on the result with 26 terms, I decided not to preserve any terms as 26 terms is enough for my further analysis.

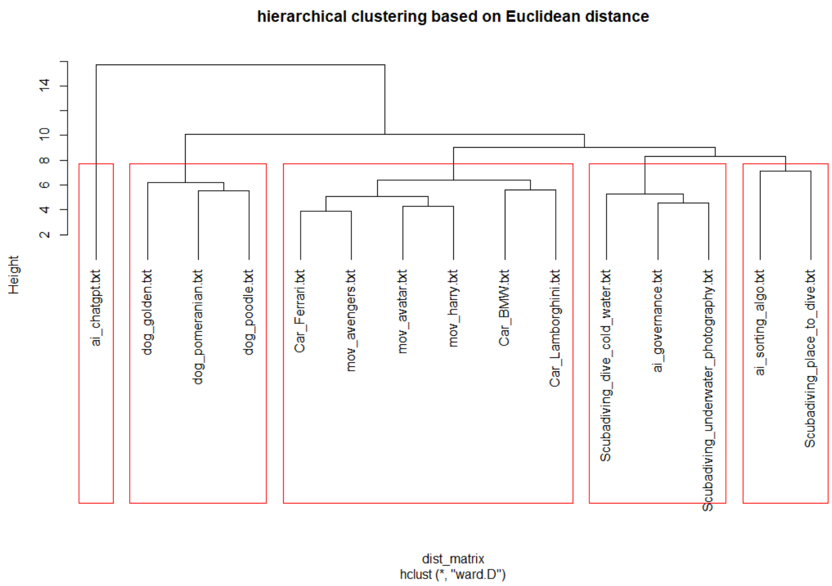


I did not include the original DTM (before removing sparse terms) in the appendix because it contains too much of terms (3000+ terms).

Question 4

For this question, I used 2 approaches, the conventional approach (Euclidean Distance) and also Cosine Distance. Also, I decided to cluster them into 5 clusters is because I had collected 5 different topics document.

Conventional approach:



Based on the hierarchical clustering graph, the conventional approach doesn't really reflect the variety of topics I had identified when I collected the documents except for dog related documents because it only clustered correctly for dog related documents, whereas the others had been misclustered. This might due to I had removed a lot of sparse terms and did not preserve any key words, hence it is not performing very well when clustering these documents and hence after removing all the sparse terms, the remaining terms might be similar between those misclustered documents.

Acuuracy:

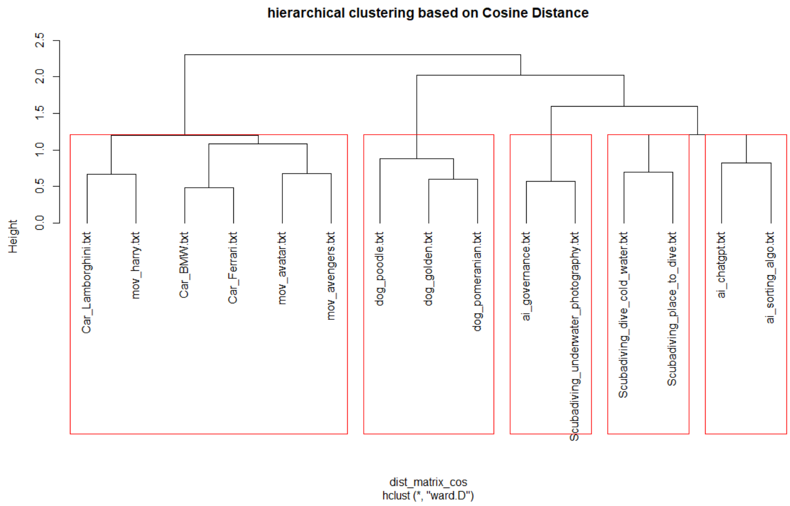
```
> table(GroupNames = topics, Clusters = groups)
             Clusters
GroupNames    1 2 3 4 5
  ai          1 1 1 0 0
  car         0 0 0 3 0
  dog         0 0 0 0 3
  mov         0 0 0 3 0
  scubadiving 0 2 1 0 0
```

(3+3+2+1)/15 = 0.6

Although based on the graph it is not performing very well but based on the accuracy score it is acceptable as the accuracy is 0.6 which is better than random guessing.

Cosine Distance approach:

Method to find Cosine distance learned from: [R: Calculate cosine distance from a term-document matrix with tm and proxy - Stack Overflow]()



From the graph we can see that Cosine Distance approach clustering does reflect the variety of topics I had identified when I collected the documents, First of all, all dog related documents had been clustered into the same cluster. Also, even though one of ai and one of scuba diving related document had been misclustered into the same cluster, but based on the graph, the other two document related to ai and scuba diving had been clustered correctly. Lastly, All car and movie related documents had been cluster into the same cluster, this might due to I had removed a lot of sparse terms and did not preserve any keywords, hence it is not performing very well when clustering these documents and hence after removing all the sparse terms, the remaining terms might be similar between car and movie related documents.

Accuracy:

```
> table(GroupNames = topics_cos, Clusters = groups_cos)
            Clusters
GroupNames    1 2 3 4 5
  ai          2 1 0 0 0
  car         0 0 3 0 0
  dog         0 0 0 3 0
  mov         0 0 3 0 0
  scubadiving 0 1 0 0 2
```

(2+3+3+2)/15 = 0.667

Based on the accuracy score it is acceptable as the accuracy is 0.67 which is better than random guessing.

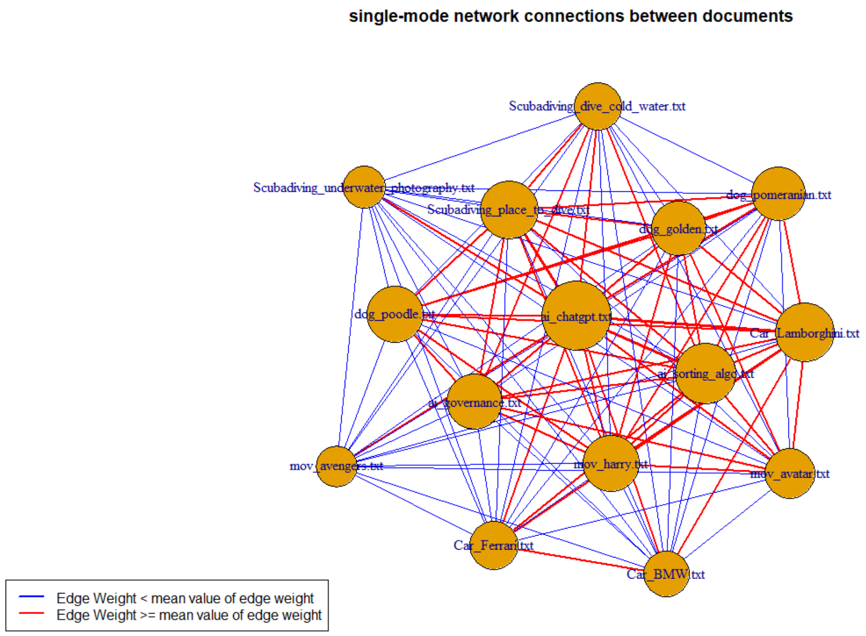
So based on the accuracy Cosine Distance approach slightly outperformed the conventional approach on clustering the documents. Also, if we look at the graphs for the hierarchical clustering we will notice that for Cosine Distance approach, both Scubadiving_dive_cold_water.txt and Scubadiving_place_to_dive.txt also ai_chatgpt.txt and ai_sorting_algo.txt had been clustered together while for the conventional approach it is not. Hence Cosine Distance approach is doing a better job.

Question 5

Table for degree, betweenness, closeness and eigen value

```
> print(tab_res[order(-eig),], digit = 3)
                                   degree between   close   eig
ai_chatgpt.txt                         14       0 0.00386 1.000
ai_sorting_algo.txt                    14       0 0.00452 0.869
Car_Lamborghini.txt                    14       0 0.00465 0.846
Scubadiving_place_to_dive.txt          14       0 0.00472 0.835
dog_poodle.txt                         14       0 0.00485 0.815
mov_harry.txt                          14       0 0.00488 0.813
ai_governance.txt                      14       0 0.00498 0.797
dog_golden.txt                         14       0 0.00508 0.785
dog_pomeranian.txt                     14       0 0.00515 0.774
mov_avatar.txt                         14       0 0.00549 0.727
Car_Ferrari.txt                        14       0 0.00571 0.698
Scubadiving_dive_cold_water.txt        14       0 0.00588 0.683
Car_BMW.txt                            14       0 0.00599 0.668
Scubadiving_underwater_photography.txt 14       0 0.00662 0.608
mov_avengers.txt                       14       0 0.00671 0.601
```

Based on the result we can see that all the documents are having same degree 14 as if we were to look at the Corpus Matrix(Table 2: Corpus Matrix) I created from Document Term Matrix(DTM) we will notice that except the diagonal value, all other value are non-zero value, this indicates that all document does share some similar terms hence the degree is the same for all documents. Based on the degree, the betweenness is 0 because all vertex already has a direct path to all other vertices hence the betweenness is 0 for all documents. Since the highest closeness value is not having the highest eigen value and based on the assignment specification, the most important (central) documents in the network would be ai_chatgpt.txt because it has the highest eigen value. Because eigen value is used to measure the importance or centrality of a node in a network.



This graph tells me that the documents are similar to each other because they are fully connected to each other, in simple terms this graph is a complete graph, but some connections are weak, and some are strong. Also, there aren't any clear groups in the data, because based on the corpus matrix (Table 2: Corpus Matrix), we can see that all documents share each and every 26 terms, there isn't any group or topic of documents share a particular set of terms, hence there isn't any clear group based on my approach. Furthermore, based on the Eigen Value and the graph, the most important (central) document will be **ai_chatgpt.txt**.

Improvement for graph

Strength of Connection

In order to show the strength of each edges, I had switch the colour of each edges to red and blue. Besides that, their width is also different. First, I look at the mean value of each edge's weight, if it is lower than the mean value it will be blue, indicating that the strength of connections between vertex might be weaker and each document connected by a blue edge will be not that similar compared to red edges. Furthermore, for edge's weight larger than their mean value, the colour will be red, indicating that the strength of connections between vertex might be stronger and each document connected by a red edge will be more similar. Also, I had also adjusted the width for each edges, the stronger the connection is, the thicker the width of the edge is. From the graph above we can observe that for example edges between dog_poodle.txt and dog_golden.txt are quite important as the colour of the edge is red and it is thick as compared to other edges, meaning to say these 2 documents might be related and similar. Another example is that, we can also observe that connection between mov_avengers.txt and car_ferrari.txt may be not that important because it has a blue edge and the width of the edge is quite slim, in simple terms these documents might not be as related and similar.

Relative Importance of Nodes

In order to show the importance of each vertex, I decided to show each vertex with different size based on their eigen value, therefore we can see the most important document will be ai_chatgpt.txt as it is having the largest size.

Communities

In order to find communities in the network, I had decided to use cluster_louvain method comes with igraph library.

Cluster_louvain method learned from: igraph R manual pages

```
> communities <- cluster_louvain(G_sdtm)
> communities
IGRAPH clustering multi level, groups: 1, mod: 0
+ groups:
  $`1`
   [1] "ai_chatgpt.txt"                  "ai_governance.txt"              "ai_sorting_algo.txt"
   [4] "Car_BMW.txt"                     "Car_Ferrari.txt"                "Car_Lamborghini.txt"
   [7] "dog_golden.txt"                  "dog_pomeranian.txt"             "dog_poodle.txt"
  [10] "mov_avatar.txt"                  "mov_avengers.txt"               "mov_harry.txt"
  [13] "Scubadiving_dive_cold_water.txt" "Scubadiving_place_to_dive.txt"  "Scubadiving_underwater_photography.txt"
```

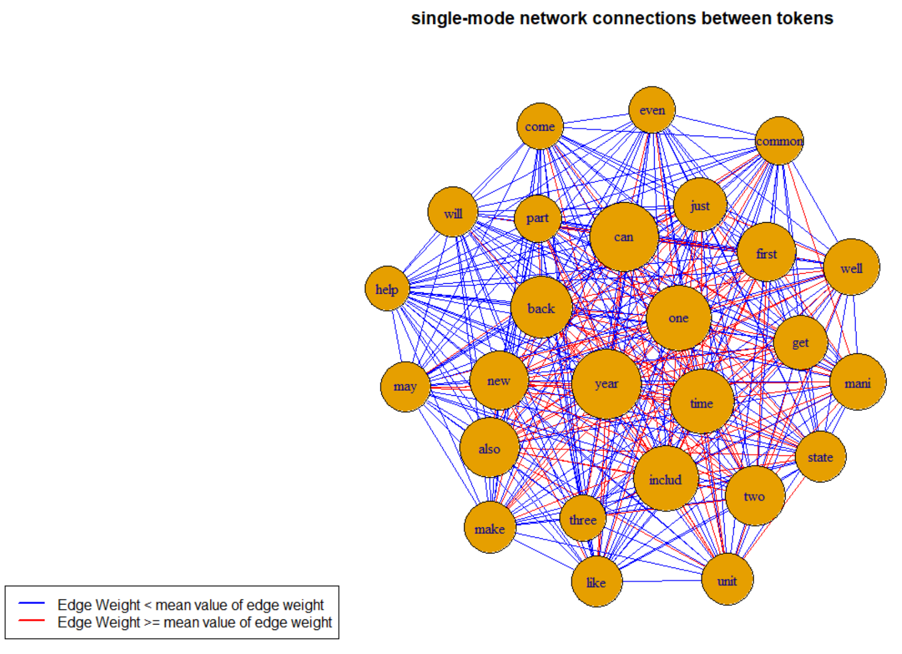
Based on the result, I only have 1 community for my graph, this might due to because my graph is highly connected and also, if we look at the Corpus Matrix (Table 2: Corpus Matrix), the terms are similar between all documents, therefore this would create me a complete graph, therefore the method return me with 1 community and hence I show the community in my network using colour yellow for the vertex.

Question 6

Table for degree, betweenness, closeness and eigen value

```
> print(t_tab_res[order(-t_eig),], digit = 3)
       t_degree t_between t_close t_eig
year         25         0 0.00391 1.000
can          25         0 0.00395 0.988
includ       25         0 0.00417 0.942
one          25         0 0.00418 0.936
time         25         0 0.00422 0.931
back         25         0 0.00444 0.885
two          25         0 0.00452 0.870
also         25         0 0.00459 0.861
first        25         0 0.00459 0.857
new          25         0 0.00461 0.855
mani         25         0 0.00485 0.811
well         25         0 0.00488 0.811
get          25         0 0.00508 0.780
just         25         0 0.00510 0.776
make         25         0 0.00529 0.746
unit         25         0 0.00538 0.741
state        25         0 0.00538 0.740
like         25         0 0.00538 0.734
will         25         0 0.00543 0.727
may          25         0 0.00546 0.725
common       25         0 0.00571 0.695
part         25         0 0.00588 0.678
three        25         0 0.00588 0.676
come         25         0 0.00588 0.676
even         25         0 0.00588 0.676
help         25         0 0.00621 0.639
```

Based on the result we can see that all the terms are having same degree 25 as if we were to look at the Token Matrix(Table 3: Token Matrix), we will notice that except the diagonal value, all other value are non-zero value, this indicates that all terms do co-occur within a document. Based on the degree, the betweenness is 0 because all vertex already has a direct path to all other vertices hence the betweenness is 0 for all terms. Since the highest closeness value is not having the highest eigen value and based on the assignment specification, the most important (central) term in the network would be year because it has the highest eigen value. Because eigen value is used to measure the importance or centrality of a node in a network.



This graph tells me that the terms co-existed in each and every document because they are fully connected to each other, in simple terms this graph is a complete graph, but some connections are weak, and some are strong. There isn't any clear groups in the data, because based on the Token Matrix(Table 3: Token Matrix), we can see that except the diagonal value, all other value are non-zero value, this indicates that all terms do co-occur within a document, there isn't any terms that does not co-exist within a document, hence there isn't any clear group based on my approach. Furthermore, based on the Eigen Value and the graph, the most important (central) document will be **year**.

Improvement for graph

Strength of Connection

In order to show the strength of each edges, I had switched the colour of each edges to red and blue. Besides that, their width is also different. First, I look at the mean value of each edge's weight, if it is lower than the mean value it will be blue, indicating that the strength of connections between vertex might be weaker. Furthermore, for edge's weight larger than their mean value, the colour will be red, indicating that the strength of connections between vertex might be stronger. Also, I had also adjusted the width for each edges, the stronger the connection is, the thicker the width of the edge is. From the graph above, although it is quite complicated and messy to interpret, but we still can observe that for example edges between **unit** and **state** are quite strong as the colour of the edge is red. Another example is that, we can also observe that connection between **state** and **mani** may be not that strong because it has a blue edge.

Relative Importance of Nodes

In order to show the importance of each vertex, I decided to show each vertex with different size based on their eigen value, therefore we can see the most important terms will be **Year** as it has the largest size.

Communities

In order to find communities in the network, I had decided to use cluster_louvain method comes with igraph library.

Cluster_louvain method learned from: igraph R manual pages

```
> communities_t
IGRAPH clustering multi level, groups: 1, mod: 0
+ groups:
  $`1`
   [1] "also"   "back"   "can"    "come"   "common" "even"   "first"  "get"    "help"
  [10] "includ" "just"   "like"   "make"   "mani"   "may"    "new"    "one"    "part"
  [19] "state"  "three"  "time"   "two"    "unit"   "well"   "will"   "year"
```

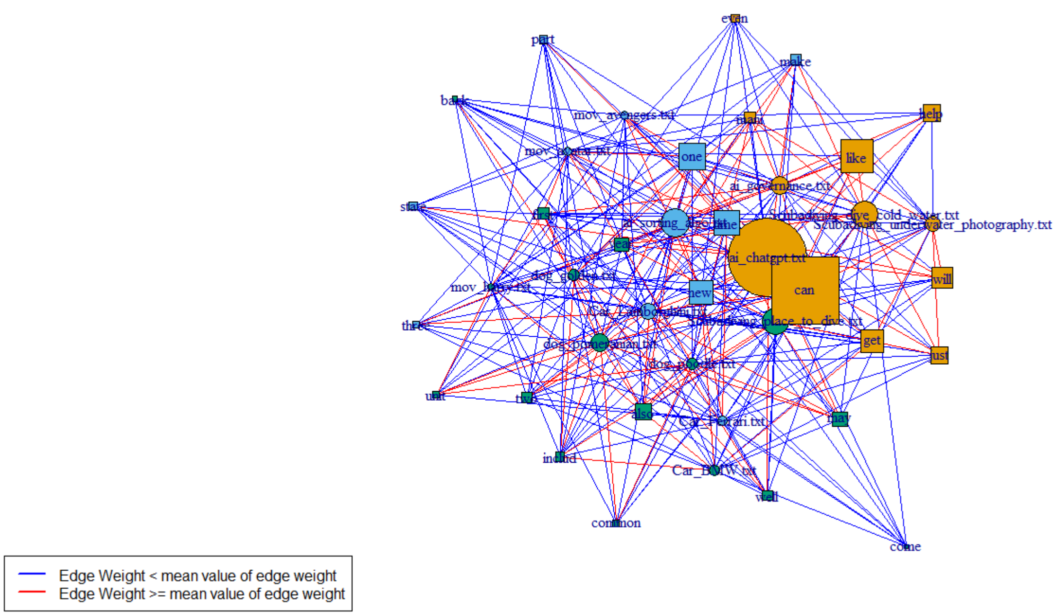
Based on the result, I only have 1 community for my graph, this might due to because my graph is highly connected and also, if we look at the Corpus Matrix Token Matrix(Table 3: Token Matrix), the terms co-exist in all of the documents, therefore this would create me a complete graph, therefore the method return me with 1 community and hence I show the community in my network using colour yellow for the vertex.

Question 7

```
> print(b_tab_res[order(-eig),], digit = 3)
                                    b_degree b_betweeness b_closeness b_eigen_value
ai_chatgpt.txt                            26         8.55      0.0103        1.0000
ai_sorting_algo.txt                       22        21.58      0.0114        0.3727
Car_Lamborghini.txt                       21        56.15      0.0120        0.2136
Scubadiving_place_to_dive.txt             21        13.07      0.0112        0.3406
dog_poodle.txt                            20        24.73      0.0115        0.1503
mov_harry.txt                             20        95.47      0.0132        0.0925
ai_governance.txt                         20        42.17      0.0119        0.2395
dog_golden.txt                            19        22.04      0.0105        0.1580
dog_pomeranian.txt                        19        18.34      0.0109        0.2363
mov_avatar.txt                            18        49.71      0.0118        0.1204
Car_Ferrari.txt                           17        52.04      0.0120        0.1292
Scubadiving_dive_cold_water.txt           17        32.84      0.0114        0.3456
Car_BMW.txt                               16        20.62      0.0110        0.1418
Scubadiving_underwater_photography.txt    15        34.09      0.0105        0.2083
mov_avengers.txt                          14        49.41      0.0114        0.1192
>
```

Based on the result we can see that all the terms and documents are having different degree, this indicates that not all vertices are connected to each other. Since the highest closeness value is not having the highest eigen value and based on the assignment specification, the most important (central) **document** in the network would be **ai_chatgpt.txt** because it has the highest eigen value. Because eigen value is used to measure the importance or centrality of a node in a network.

Bipartite (two-mode) network graph



This graph tells me that, the relationship between documents and terms is that there are some documents which are very similar because they have similar terms. There is 3 different groups in my graph which are represented as yellow, green and blue colour. Furthermore, based on the Eigen Value and the graph, the most important (central) document will be **ai_chatgpt.txt** and the most important (central) terms will be **can**.

Improvement for graph

Strength of Connection

In order to show the strength of each edges, I had switched the colour of each edges to red and blue. Besides that, since the graph is too complex, hence I decided not to add in width like I did for the previous question. First, I look at the mean value of each edge's weight, if it is lower than the mean value it will be blue, indicating that the strength of connections between vertex might be weaker. Furthermore, for edge's weight larger than their mean value, the colour will be red, indicating that the strength of connections between vertex might be stronger. Therefore by looking at the colours, we can tell the connection between document **scubadiving_underwater_photography.txt** and the term **will** is quite strong.

Relative Importance of Nodes

In order to show the importance of each vertex, I decided to show each vertex with different size based on their eigen value, therefore we can see the most important terms will be **can** as it has the largest size and the most important document will be **ai_chatgpt.txt** as it also has the largest size.

Communities

In order to find communities in the network, I had decided to use cluster_louvain method comes with igraph library.

Cluster_louvain method learned from: [igraph R manual pages]

```
> communities_b
IGRAPH clustering multi level, groups: 3, mod: 0.2
+ groups:
  $`1`
   [1] "ai_chatgpt.txt"                         "ai_governance.txt"                      "Scubadiving_dive_cold_water.txt"
   [4] "Scubadiving_underwater_photography.txt" "can"                                    "even"
   [7] "get"                                    "help"                                   "just"
  [10] "like"                                   "mani"                                   "will"

  $`2`
   [1] "ai_sorting_algo.txt" "Car_Ferrari.txt"     "mov_avatar.txt"      "mov_avengers.txt"    "new"
   [6] "one"                 "part"                "state"               "three"               "time"

  + ... omitted several groups/vertices
```

Based on the result, I will have 3 community for my graph, hence I decided to show it in my graph with different colours(yellow, green and blue)based on each group from group 1 to group 3.

Appendix

	also	back	can	come	common	even	first	get	help	includ	just	like	make	mani	may	new	one	part	state	three	time	two	unit	well	will	year
ai_chatgpt.txt	4	1	32	1	1	4	2	13	9	2	8	17	3	2	6	8	10	3	3	2	8	4	1	4	9	4
ai_governance.txt	0	1	4	0	1	1	3	2	2	1	1	8	1	3	0	2	3	2	0	1	0	1	0	1	8	3
ai_sorting_algo.txt	3	1	8	1	1	0	2	0	3	1	0	4	4	3	1	11	7	2	2	3	9	1	2	0	3	1
Car_BMW.txt	3	0	2	2	0	0	2	4	0	5	1	1	0	0	1	4	0	0	0	1	2	2	0	2	1	2
Car_Ferrari.txt	2	0	2	1	0	0	1	2	1	1	2	0	0	0	0	5	0	0	1	2	4	1	2	1	1	1
Car_Lamborghini.txt	5	1	6	1	0	1	2	1	0	1	2	1	3	0	0	5	1	1	0	4	1	3	3	3	1	2
dog_golden.txt	3	3	2	0	1	1	5	0	0	3	0	0	1	4	3	1	6	2	2	0	1	2	2	4	0	2
dog_pomeranian.txt	5	1	8	1	5	0	6	0	0	4	0	0	0	2	3	0	3	2	2	1	1	3	4	1	2	1
dog_poodle.txt	3	1	2	1	5	2	0	1	0	2	1	2	0	3	5	2	1	0	1	0	2	2	3	1	0	3
mov_avatar.txt	0	1	0	0	0	0	4	1	1	1	0	2	2	1	1	4	3	1	1	2	3	2	1	0	0	3
mov_avengers.txt	1	1	1	0	0	0	0	2	1	1	1	0	0	0	0	5	3	2	2	0	5	0	1	0	0	1
mov_harry.txt	3	1	1	0	1	1	3	1	0	1	1	0	1	1	0	0	1	1	2	2	1	2	6	1	0	2
Scubadiving_dive_cold_water.txt	0	1	15	1	1	1	1	0	2	0	3	4	3	4	1	0	2	0	0	0	5	0	0	1	1	3
Scubadiving_place_to_dive.txt	6	1	8	2	2	1	0	2	2	3	5	3	2	2	4	2	6	0	1	0	8	0	0	1	1	5
Scubadiving_underwater_photography.txt	2	0	5	0	0	1	2	6	1	0	3	1	1	3	2	1	2	0	0	0	0	1	0	0	8	0

Table 1: DTM after removing all the sparse terms

	ai_chatgpt.txt	ai_governance.txt	ai_sorting_algo.txt	Car_BMW.txt	Car_Ferrari.txt	Car_Lamborghini.txt	dog_golden.txt	dog_pomeranian.txt	dog_poodle.txt	mov_avatar.txt	mov_avengers.txt	mov_harry.txt	Scubadiving_dive_cold_water.txt	Scubadiving_place_to_dive.txt	Scubadiving_underwater_photography.txt
ai_chatgpt.txt	0	20	22	16	17	21	19	19	20	18	14	20	17	21	15
ai_governance.txt	20	0	16	12	12	17	14	13	14	14	10	16	14	16	13
ai_sorting_algo.txt	22	16	0	13	14	17	17	18	16	17	12	16	14	17	12
Car_BMW.txt	16	12	13	0	14	15	10	12	13	10	8	11	10	13	10
Car_Ferrari.txt	17	12	14	14	0	15	11	13	13	11	11	13	9	13	9
Car_Lamborghini.txt	21	17	17	15	15	0	15	15	16	14	12	17	13	16	12
dog_golden.txt	19	14	17	10	11	15	0	16	16	14	11	17	12	15	10
dog_pomeranian.txt	19	13	18	12	13	15	16	0	15	13	10	16	12	14	8
dog_poodle.txt	20	14	16	13	13	16	16	15	0	13	12	16	13	18	11
mov_avatar.txt	18	14	17	10	11	14	14	13	13	0	11	14	10	13	10
mov_avengers.txt	14	10	12	8	11	12	11	10	12	11	0	12	7	12	7
mov_harry.txt	20	16	16	11	13	17	17	16	16	14	12	0	12	15	10
Scubadiving_dive_cold_water.txt	17	14	14	10	9	13	12	12	13	10	7	12	0	16	11
Scubadiving_place_to_dive.txt	21	16	17	13	13	16	15	14	18	13	12	15	16	0	13
Scubadiving_underwater_photography.txt	15	13	12	10	9	12	10	8	11	10	7	10	11	13	0

Table 2: Corpus Matrix

	also	back	can	come	common	even	first	get	help	includ	just	like	make	mani	may	new	one	part	state	three	time	two	unit	well	will	year
also	0	9	12	8	7	7	9	9	6	11	9	7	7	8	8	10	10	7	9	7	11	10	9	9	8	11
back	9	0	11	7	9	8	9	8	7	11	8	8	9	10	8	9	12	9	9	7	11	9	9	9	7	12
can	12	11	0	9	9	9	11	10	8	12	11	9	9	10	9	11	12	8	9	8	12	11	9	11	10	13
come	8	7	9	0	6	5	7	6	5	8	7	7	5	6	7	7	7	4	6	6	9	7	6	8	8	9
common	7	9	9	6	0	7	7	5	5	8	6	6	7	9	7	6	9	6	7	5	8	7	6	8	6	9
even	7	8	9	5	7	0	7	7	5	7	8	7	8	8	6	7	9	5	5	4	7	7	5	8	6	8
first	9	9	11	7	7	7	0	8	7	10	8	8	9	9	8	9	10	8	7	9	10	11	8	9	9	11
get	9	8	10	6	5	7	8	0	7	10	10	8	7	7	6	10	9	6	7	7	9	9	7	8	7	10
help	6	7	8	5	5	5	7	7	0	7	7	7	7	7	6	8	8	5	6	5	7	6	5	5	7	8
includ	11	11	12	8	8	7	10	10	7	0	9	8	8	9	8	11	11	9	10	9	12	11	10	10	8	13
just	9	8	11	7	6	8	8	10	7	9	0	8	7	7	6	9	9	5	6	6	9	8	6	9	8	10
like	7	8	9	7	6	7	8	8	7	8	8	0	8	8	8	9	9	5	5	6	8	8	5	7	8	9
make	7	9	9	5	7	8	9	7	7	8	7	8	0	9	7	8	10	7	6	6	8	8	6	7	7	9
mani	8	10	10	6	9	8	9	7	7	9	7	8	9	0	9	8	11	7	8	6	9	9	7	8	7	10
may	8	8	9	7	7	6	8	6	6	8	6	8	7	9	0	8	9	5	7	5	9	8	6	7	7	9
new	10	9	11	7	6	7	9	10	8	11	9	9	8	8	8	0	10	7	8	7	10	10	8	8	8	11
one	10	12	12	7	9	9	10	9	8	11	9	9	10	11	9	10	0	9	9	7	11	10	9	9	8	12
part	7	9	8	4	6	5	8	6	5	9	5	5	7	7	5	7	9	0	7	7	8	8	8	6	5	9
state	9	9	9	6	7	5	7	7	6	10	6	5	6	8	7	8	9	7	0	6	10	8	9	7	5	10
three	7	7	8	6	5	4	9	7	5	9	6	6	6	6	5	7	7	7	6	0	8	9	7	7	7	9
time	11	11	12	9	8	7	10	9	7	12	9	8	8	9	9	10	11	8	10	8	0	10	10	10	8	13
two	10	9	11	7	7	7	11	9	6	11	8	8	8	9	8	10	10	8	8	9	10	0	9	9	8	11
unit	9	9	9	6	6	5	8	7	5	10	6	5	6	7	6	8	9	8	9	7	10	9	0	7	5	10
well	9	9	11	8	8	8	9	8	5	10	9	7	7	8	7	8	9	6	7	7	10	9	7	0	8	11
will	8	7	10	8	6	6	9	7	7	8	8	8	7	7	7	8	8	5	5	7	8	8	5	8	0	9
year	11	12	13	9	9	8	11	10	8	13	10	9	9	10	9	11	12	9	10	9	13	11	10	11	9	0

Table 3: Token Matrix

Source

car

BMW: https://paultan.org/2023/01/10/2023-bmw-3-series-facelift-launched-in-malaysia-ckd-g20-lci-320i-for-rm264k-330e-rm279k-330i-rm298k/

Ferrari: https://paultan.org/2015/06/16/ferrari-488-gtb-debuts-in-malaysia-from-rm1-07-mil/

lambo: https://paultan.org/2023/03/30/lamborghini-revuelto-debuts-6-5-litre-na-v12-phev-with-1015-ps-gets-new-8dct-three-e-motors-adas/

Scuba Diving

Scuba Diving underwater photraphy:https://blog.padi.com/five-tips-for-getting-started-with-underwater-photography/

Scuba Diving dive cold water:https://blog.padi.com/learning-to-dive-in-cold-water/

Scubadiving_place_to_dive: https://blog.padi.com/best-places-to-scuba-dive-year-round/

AI

Ai sorting algo: https://www.deepmind.com/blog/alphadev-discovers-faster-sorting-algorithms

ai governance: https://openai.com/blog/governance-of-superintelligence

chatgpt: https://www.cnet.com/tech/services-and-software/google-launches-new-ai-search-engine-how-to-sign-up/

dog

golden: https://en.wikipedia.org/wiki/Golden_Retriever

pomeranian: https://en.wikipedia.org/wiki/Pomeranian_dog

poodle: https://en.wikipedia.org/wiki/Poodle

Movie

mov_harry :
https://en.wikipedia.org/wiki/Harry_Potter_and_the_Philosopher%27s_Stone_(film)

mov_avenger: https://en.wikipedia.org/wiki/Avengers:_Endgame

mov_avatar: https://en.wikipedia.org/wiki/Avatar:_The_Way_of_Water

R programming code:

```
# remove enviroment first
rm(list=ls())
#install.packages("tm")
#install.packages("slam")
#install.packages("SnowballC")
#install.packages("textmineR")
#install.packages("igraphdata")

library(slam)
library(tm)
library(SnowballC) # for stemming
library(textmineR)
library(igraph)
library(igraphdata)
library(gridExtra)

#question 2

# create corpus
cname = file.path(".","CorpusAbstracts","txt")

cname

dir(cname)

# create corpus
docs = Corpus(DirSource((cname)))
summary(docs)
```

```
# question 3

# Tokenisation
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, removePunctuation)

docs <- tm_map(docs, content_transformer(tolower))

# Filter words

# Remove stop words
docs <- tm_map(docs, removeWords, stopwords("english"))

# remove white space
docs <- tm_map(docs, stripWhitespace)

# Stemming
docs <- tm_map(docs, stemDocument, language = "english")

# create DTM
dtm <- DocumentTermMatrix(docs)
dtm_csv = as.data.frame(as.matrix(dtm))
write.csv(dtm_csv, "dtm.csv")

inspect(dtm)


# remove sparse term down to 20++

#ori dtm
dim(dtm)
```

```
# 45% empty
sdtm <- removeSparseTerms(dtm,0.45)
dim(sdtm)

inspect(sdtm)

# output it as table so that it can be saved to jpeg and insert into my report
rep_sdtm = as.table(sdtm)

#method i learned online to output the correlation as table so i can paste it in words
jpeg("rep_sdtm.jpg", height=1000, width=2000, units = "px")
grid.table(rep_sdtm)
dev.off()

# write to csv
sdtm_csv = as.data.frame(as.matrix(sdtm))
write.csv(sdtm_csv, "sdtm.csv")




# question 4

#  hierarchical clustering
sdtm_matrix = as.matrix(sdtm)
dim(sdtm_matrix)
class(sdtm_matrix)

# plot using normal value
dist_matrix <- dist(scale(sdtm_matrix))
fit = hclust(dist_matrix, method = "ward.D")
```

```
plot(fit, main="hierarchical clustering based on Euclidean distance")

plot(fit, hang=-1, main="hierarchical clustering based on Euclidean distance")

rect.hclust(fit, k=5, border = "red")



topics =
c("ai","ai","ai","car","car","car","dog","dog","dog","mov","mov","mov","scubadiving","scuba
diving","scubadiving")

groups = cutree(fit,k=5)


table(GroupNames = topics, Clusters = groups)



# plot using cosine

# count cosine value

require(proxy)


dist_matrix_cos <- dist(scale(sdtm_matrix), method="cosine")


fit_cos = hclust(dist_matrix_cos, method = "ward.D")

plot(fit_cos, main="hierarchical clustering based on Cosine Distance")

plot(fit_cos, hang=-1, main="hierarchical clustering based on Cosine Distance")


rect.hclust(fit_cos, k=5, border = "red")


topics_cos =
c("ai","ai","ai","car","car","car","dog","dog","dog","mov","mov","mov","scubadiving","scuba
diving","scubadiving")

groups_cos = cutree(fit_cos,k=5)


table(GroupNames = topics_cos, Clusters = groups_cos)
```

```
# question 5
# Corpus Matrix
dim(sdtm_matrix )

# create binary matrix
binary_sdtm = as.matrix((sdtm_matrix>0)+0)
binary_sdtm
# create corpus network data
#cor_matrix = binary_sdtm %*% t(binary_sdtm)
cor_matrix = binary_sdtm %*% t(binary_sdtm)

# make diagonal zero
diag(cor_matrix) = 0

cor_matrix

#output it as table
tab.cor_matrix = as.table(cor_matrix)

#method i learned online to output the correlation as table so i can paste it in words
jpeg("tab.cor_matrix.jpg", height=2000, width=3000, units = "px")
grid.table(tab.cor_matrix)
dev.off()

# Create Graph
G_sdtm = graph_from_adjacency_matrix(cor_matrix, mode = "undirected", weighted =
TRUE)

# calculate the degree, betweenness, clossness and eigen value
degree = as.table(degree(G_sdtm))
```

```
between = as.table(betweenness(G_sdtm))
close = as.table(closeness(G_sdtm))
eig = as.table(evcent(G_sdtm)$vector)

tab_res = as.data.frame(rbind(degree,between,close,eig))
tab_res = t(tab_res)

print(tab_res[order(-eig),], digit = 3)

# show the importance of edges by changing the colour of edge
mean_g_edge = mean(E(G_sdtm)$weight)
for( i in seq_len(length(E(G_sdtm)$weight))){
  if(E(G_sdtm)[i]$weight > mean_g_edge){
    E(G_sdtm)[i]$color = "red"}
  else if(E(G_sdtm)[i]$weight  < mean_g_edge){
    E(G_sdtm)[i]$color = "blue"
  }

}

# show the importance of edges by changing the width of edge
E(G_sdtm)$width = E(G_sdtm)$weight/100

# show the importance of vertices by changing the size of vertex
V(G_sdtm)$size <- eig*30

# find the communities
communities <- cluster_louvain(G_sdtm)
communities
```

```
set.seed("32439180")

plot(G_sdtm, main = "single-mode network connections between documents", vertex.color
= communities$membership)

legend("bottomleft", legend = c("Edge Weight < mean value of edge weight","Edge
Weight >= mean value of edge weight") , col = c("blue","red"), lwd = 2)




# Question 6

# based on terms



T_sdtm = sdtm_matrix


# binary matrix

T_sdtm = as.matrix((T_sdtm>0)+0)


# Token Matrix

T_matrix = t(T_sdtm) %*% T_sdtm


# set diagonal to 0

diag(T_matrix) = 0


#output it as table

rep_token_matrix = as.table(T_matrix)


#method i learned online to output the correlation as table so i can paste it in words

jpeg("rep_token_matrix.jpg", height=2000, width=3000, units = "px")

grid.table(rep_token_matrix)

dev.off()
```

```
# plot graph
Token_graph = graph_from_adjacency_matrix(T_matrix, mode="undirected", weighted =
TRUE)

# find communities
communities_t <- cluster_louvain(Token_graph )
communities_t

# calculate the degree, betweenness, clossness and eigen value
t_degree = as.table(degree(Token_graph ))
t_between = as.table(betweenness(Token_graph ))
t_close = as.table(closeness(Token_graph ))
t_eig = as.table(evcent(Token_graph )$vector)

t_tab_res = as.data.frame(rbind(t_degree,t_between,t_close,t_eig))
t_tab_res = t(t_tab_res)

print(t_tab_res[order(-t_eig),], digit = 3)



# find the the edge weight is lower or higher than the mean value
# if lower: edge colour = blue
# if higher: edge colour = red
mean_token_weight = mean(E(Token_graph)$weight)
for( i in seq_len(length(E(Token_graph)$weight))){
  if(E(Token_graph)[i]$weight >= mean_token_weight){
    E(Token_graph)[i]$color = "red"}
  else if(E(Token_graph)[i]$weight < mean_token_weight){
    E(Token_graph)[i]$color = "blue"
  }
```

```
}

# change the size of the vertex based on eigen value
V(Token_graph)$size <- t_eig*30

# change the width of edges based on the weight of each edges
E(Token_graph)$width = E(Token_graph)$weight/7



#graph_attr(Token_graph, "layout") <- layout_with_lgl(Token_graph,area =
vcount(Token_graph)^2)
set.seed("32439180")
plot(Token_graph, main = "single-mode network connections between tokens", vertex.color
= communities_t$membership)
legend("bottomleft", legend = c("Edge Weight < mean value of edge weight","Edge
Weight >= mean value of edge weight") , col = c("blue","red"), lwd = 2)

#Question 7

# make another copy of original document terms matrix
sdtm_bip = as.data.frame(as.matrix(sdtm))

# add row names
sdtm_bip$Doc = rownames(sdtm_bip)

sdtm_bip_2 = data.frame()

# put in data
for(i in 1:nrow(sdtm_bip)){
  for(j in 1:(ncol(sdtm_bip)-1)){
    touse = cbind(sdtm_bip[i,j], sdtm_bip[i,ncol(sdtm_bip)], colnames(sdtm_bip[j]))
```

```
    sdtm_bip_2 = rbind(sdtm_bip_2, touse)
  }
}

# rename the column
colnames(sdtm_bip_2) = c("weight","Doc","Terms")

# delete 0 weights
sdtm_bip_3 = sdtm_bip_2[sdtm_bip_2$weight !=0,]

# put in correct order for plotting: doc, terms, weight
sdtm_bip_3 = sdtm_bip_3[,c(2,3,1)]

#output it as table
rep_sdtm_bip_3  = as.data.frame(sdtm_bip_3 )

#method i learned online to output the correlation as table so i can paste it in words
jpeg("rep_sdtm_bip_3.jpg", height=2000, width=3000, units = "px")
grid.table(rep_sdtm_bip_3)
dev.off()

# plot bipartite network graph
g_b <- graph.data.frame(sdtm_bip_3, directed=FALSE)
bipartite.mapping(g_b)

# calulate the degree, betweenness clossness and eigen value
b_degree = as.table(degree(g_b))
b_betweeness = as.table(betweenness(g_b))
b_closeness = as.table(closeness(g_b))
b_eigen_value = as.table(evcent(g_b)$vector)
```

```
b_tab_res = as.data.frame(rbind(b_degree,b_betweeness,b_closeness,b_eigen_value))
b_tab_res = t(b_tab_res)

# print it as dataframe with highest eigen value to lowest
print(b_tab_res[order(-eig),], digit = 3)

# find the communities for this bipartite graph
communities_b <- cluster_louvain(g_b)
communities_b

V(g_b)$type <- bipartite_mapping(g_b)$type

# square for terms and circle for documents
V(g_b)$shape <- ifelse(V(g_b)$type,"square","circle")

# change the edge colour
# if edge weight < mean then blue (not important)
# if edge weight > mean then red (important)
mean_bip_weight = mean(as.numeric(E(g_b)$weight))
for( i in seq_len(length(E(g_b)$weight))){
   if(as.numeric(E(g_b)[i]$weight) >= mean_bip_weight){
   E(g_b)[i]$color <- "red"}

  else if(as.numeric(E(g_b)[i]$weight) < mean_bip_weight){
   E(g_b)[i]$color = "blue"
  }



}
```

```
# most important vertex to diamond

V(g_b)$size <- b_eigen_value*30



# change the edge width based on their respective weghit

E(g_b)$width = as.numeric(E(g_b)$weight)/7


set.seed("32439180")


plot(g_b, vertex.color = communities_b$membership, main="Bipartite (two-mode) network graph")


legend("bottomleft", legend = c("Edge Weight < mean value of edge weight","Edge Weight >= mean value of edge weight") , col = c("blue","red"), lwd = 2)
```