Text Mining and Graph Application on SFR Tweets

April 2019

Khanh TRUONG - Mickaël OLIFANT - Clarisse BOINAY

Graph theory is the field of mathematics in which the object studies are graphs. A graph is defined by a set of nodes (vertices) and a set of respective edges (links). In some problems, nodes attributes and edges attributes are also included. Graph theory and text mining can be combined to become a truly powerful tools to analyze data of social network. The use of this approach is increasingly important especially in recent years with the rapid growth of Facebook and Tweeter.

In this project, tweets about SFR (a French telecommunications company) are going to be employed and analyzed to discover what the topics of users are when they mention SFR.

Following are content of this project:

1. Filter data
2. Clean text of tweets
3. Document term matrix
4. K-means clustering
5. Graph approach
6. Conclusion

1. Filter data

Only SFR French tweets are selected for this analysis. Moreover, the re-tweeted tweets are removed from the dataset since re-tweeted tweets make the data imbalanced (there are many identical content tweets) and therefore may lead to bias in the analysis. The final data set includes 6,257 rows, representing 6,257 tweets.

2. Clean text of tweets

Obviously, the raw retrieved tweets are not ready not for statistical models. It is necessary to perform several preprocessing steps order to clean the text. Accordingly, both regular expression and package "tm" are used.

2.1 Regular expression

An example raw original tweet is presented below to demonstrate the effect of every cleaning approach.

"Bon démarrage pour le #SNLGAD sur @M6 https://t.co/fkhNM5x8uN via @SFRNews https://t.co/ObBmqlXwWq"

First of all, although hash-tags (words start by "#") are usually the key topic of tweets, we remove those hash-tag words. It is because since those key words typically appear in the main text of the tweets anyway. As a result, if we counted the hash-tag, those words would be counted twice. Moreover, many tweets do not have hash-tag. Hence, it would be fairer to remove hash-tag in this investigation.

"Bon démarrage pour le sur @M6 https://t.co/fkhNM5x8uN via @SFRNews https://t.co/ObBmqlXwWq"

Secondly, we remove mentioned entities in the tweets, which are string of words start by "@". The mentioned entities are user names most of the time; therefore, those entities words should be excluded from our analysis.

"Bon démarrage pour le sur https://t.co/fkhNM5x8uN via https://t.co/ObBmqlXwWq"

Thirdly, url, which are strings of words start by "*http*", are removed from the tweets since they do not reveal the content of the tweets.

"Bon démarrage pour le sur via "

2.2 2.2. Corpus transformations with "tm" package

Various types of text cleaning are going to be utilized including but not limited to punctuation removal, stop-words removal, stemming. Similar to using regular expression, an example of raw tweet is presented to see the outcome of every text preprocessing step.

"A LA TV - Match Suisse / France sur SFR Sport 2 à 10h30"

First, we ignore all special characters using punctuation removing method.

"A LA TV Match Suisse France sur SFR Sport 2 à 10h30"

Besides, numbers would be taken out.

"A LA TV Match Suisse France sur SFR Sport à h"

Furthermore, we assume that the content of text is not case sensitive, i.e. lowercase words have similar meaning as they are in uppercase form.

"a la tv match suisse france sur sfr sport à h"

Next, we withdraw common words which do not reveal much information such as "*la*" or "*à*" as the above example, also called "stop words". Though package "tm" provides a list of French stop-words, we figure out that there are many other meaningless words in our tweet data such as "*sfr*", "*chez*" and "*cest*" not included in the base list of package "tm". Thus, many additional stop-words are attached to better clean the text (more detail about list of additional stop-words is available in the script code at Appendix).

"a tv match suisse france sport h"

Finally, we perform stemming process. This step is important since it eliminates the form variation of a word. For instance, "*datas*" and "*data*" would be stemmed into similar word "*data*". Consequently, the analysis does not consider "*datas*" and "*data*" as different words.

"a tv match suiss franc sport h"

3. Document Term Matrix

3.1 Term frequency

The corpus text data is going to be transformed in to Document Term Matrix using term frequency. We are aware that there are still many strings that have not cleaned out by the regular express and corpus transformations such as *"<U+0001F62E>*" and "*€*".

Therefore, we filter out every string that appears in less than 1% of the data (~ 62 tweets). If the threshold was set to be too small, some weird strings would still remain in the data. On the other hand,

if the limit was too large, some important key words (which may indicate a special topic) would be omitted. The threshold 1% used in this analysis is already carefully considered and tested.

As a result, some tweets are cleaned off. The final Document Term Matrix contains 5,268 rows representing 5,268 tweets and 112 columns representing 112 key words. The following is the table of most frequent words in the Document Term Matrix.

Words	Frequency
box	419
client	386
fait	358
tout	349
depui	332
quand	331
servic	297
fibr	274
fair	271
réseau	247

As depicted, "*box*", "*client*" and "*tout*" are the most common words. This list displays some indications of topic discussed in Tweeter about the company SFR.

3.2 Correlation of words

Based on the Document Term Matrix, it is possible to retrieve the most correlated couple of words. For instance, we would like to know what people talk when they mention "*réseau*". The below table shows three words that go with "*réseau*" the most. Despite the fact that the correlation statistic is not immensely significant, it seems SFR receives fairly many negative feedbacks for their connection service.

Words	Correlation with "réseau"
aucun	0.06
merd	0.05
mobil	0.04

Moreover, we would like to know what people say when they mention "*servic*". Once more, there are some negative adjectives that usually go along with "*servic*" such as "*aucun*" and "*pire*", which may be an alert to SFR that they have some problems in their customer services.

Words	Correlation with "servic"
client	0.22
aucun	0.09
téléphone	0.06
depui	0.05
pire	0.05

Nevertheless, it is understandable that people do not habitually tell good feedback about daily services such as mobile or internet connection. For example, it's uncommon to say "*My internet connection is good*" on Facebook or Tweeter. Users typically only report on public online network when they have some problem. Therefore, it is not surprised that negative feedbacks dominance over positive ones.

The following table provide a complete overview of users' reaction when they mention products of SFR on Tweeter.

Key word	box	fibr	téléphone	mobil	client
Most correlated words	nouvell	abonn	servic	red	servic
	march	box	wifi	forfait	red
	décodeur	peux	changer	lign	bjr
	bjr	bjr	internet	opérateur	nouveau
	internet	free	nouveau	toujour	pire

3.3 Term frequency – Inverse document frequency (IF-IDF)

Before applying machine learning model such as k-means and Louvain method, we transform the term frequency matrix (tf matrix) into term frequency–inverse document frequency (tf-idf matrix). If-idf is a method that transforms the term frequency into tf-idf score which reflects importance of words in a document. Basically, tf-idf score of a word in a tweet will increase if that word appears several times in the tweet. Reversely, the score will decease if the word appear in many other tweets.

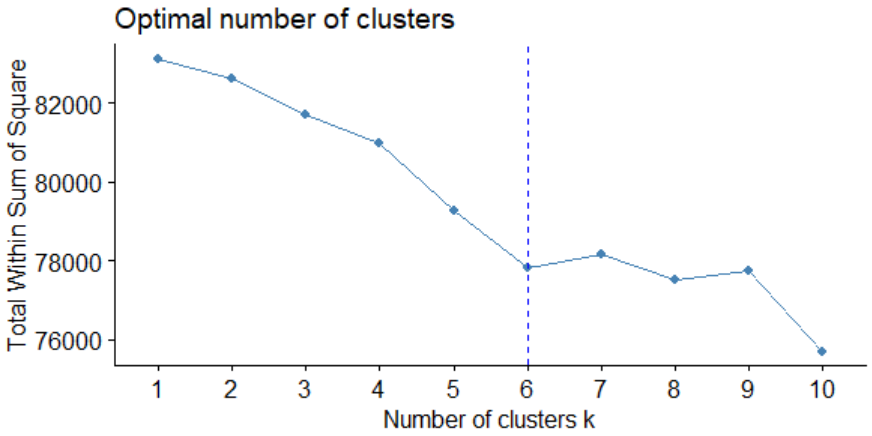
By leveraging the importance of words, it would be more efficient to discover topics in the tweets by clustering models.

4. K-means clustering

K-means clustering is one of the most unsupervised learning technique used because of its simplicity and ease to use. The main disadvantage of the method is the number of cluster k needs to be specified beforehand.

4.1 Select optimal k

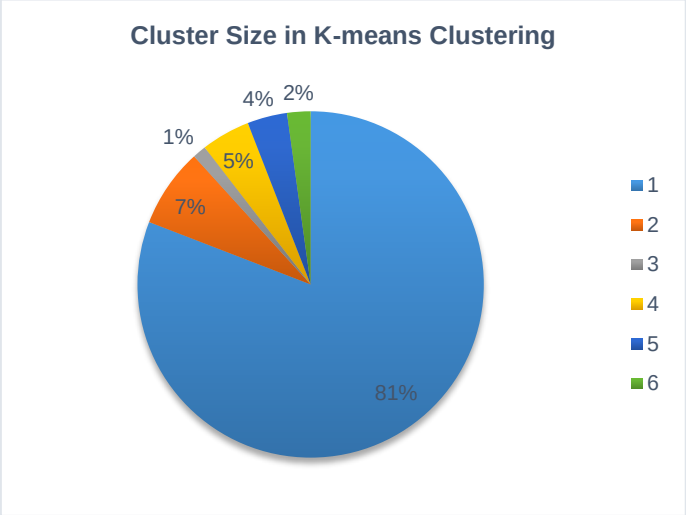
We choose k^* at which the total within sum of square (WSS) levelled, i.e. for $k > k^*$, there is not significant drop in WSS if k increases.



As depicted in the figure, for $k > 6$, the WSS starts to be flat. Therefore, $k = 6$ is chosen for our k-means clustering.

4.2 K-means clustering

The distribution of tweets according to clusters is presented in a PCA plane. There are many overlap regions in the plane and the tweets seems to be highly centered. Nevertheless, the above PCA plane should not be taken seriously into account because the plane explains only 5% variation of the data (more detail in appendix). Moreover, our main concern is the topic of every tweet cluster, rather than how the tweets are distributed.



4.3 Topic of every cluster

	cluster1	cluster2	cluster3	cluster4	cluster5	cluster6
Most important words	tout	rien	video	moi	servic	aucun
	quand	alor	jour	nouvell	client	just
	sport	faut	quand	bonsoir	depui	problèm
	fait	dire	semain	box	aucun	réseau
	box	contact	coup	depui	fair	client

For every cluster, we extract the words that have highest important score (tf-idf score). It is shown in the above table that topics of the tweets in different clusters are fairly distinguished. While tweets in cluster1 usually mention "*sport*" in their content, tweets in cluster2 seems to speak about "*contract*"

things. Moreover, content of tweets in cluster5 refer customer services with key words "*servic*" and "*client*". Whereas topic in cluster6 is connection problems.

5. Graph approach

5.1 Graph creation

From tf-idf Document Term Matrix, a graph is created in which each:

- Nodes: one node represents one tweet,
- Edges: there is an edge between two tweets if they have at least one common word.
- Edges weight: the common tf-idf score.

Due to computation constraints (more detail in script code at Appendix), only 1,000 random tweets are taken from the tf-idf matrix, which represent approximately 20% of the data. As a result, the final graph has 1,000 nodes and 79,666 edges.

5.2 Graph description

Before doing a clustering, we study some basics statistics about this graph.

Centrality measure	Value
density	0.15949
transitivity	0.42874
diameter	5.36756
radius	3
girth	3

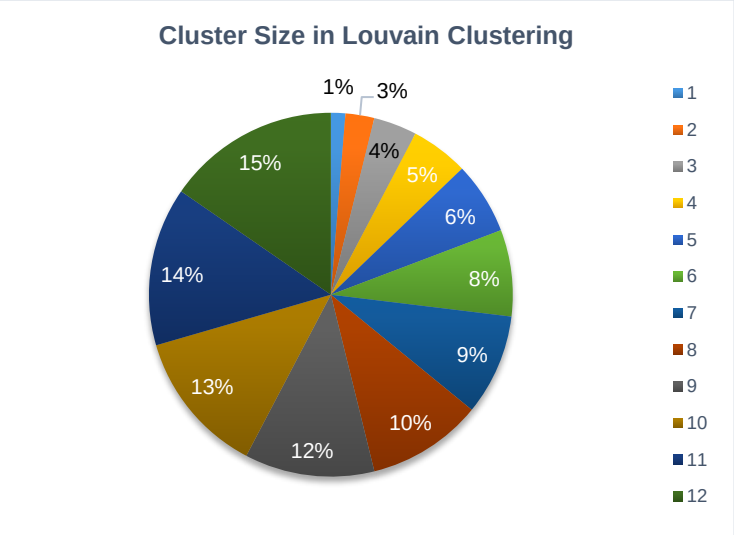
- The **density** of a graph is the number of edges of the graph divided by the number of possible edges. Here, the density is 0.15949. This means that there are few edges compared to the potential number.
- The **transitivity** of a graph is based on the relative number of triangles in the graph, compared to total number of connected triples of nodes. Here, the transitivity is equal to 0.42874.
- The **diameter** of a graph is the maximum eccentricity of any vertex in the graph. That is, it is the greatest distance between any pair of vertices. To find the diameter of a graph, first find the shortest path between each pair of vertices. The greatest length of any of these paths is the diameter of the graph. The weighted version is used is this case. The diameter is here 5.36756.
- The **radius** of a graph is the minimum graph eccentricity of any graph vertex in a graph. The radius is here 3.
- The **girth** of a graph is the length of a shortest cycle contained in the graph. The girth here is equal to 3, as in a triangular mesh.

5.3 Louvain algorithm

Community detection is very important to understand deeply the relation between individuals, especially in social network. The graph used in this case is constructed regarding the inner characteristics of each tweet, which is here the tf-idf score of the main words of the tweets. The expectation here is with the community detection, we could be able to identify the subject of tweets concerning SFR. Remind that the studied graph is based on a sample of the whole database due to computation constraints. Since the graph is complete, this may potentially influence the results.

The Louvain algorithm works as follow : it maximizes a modularity score for each community, where the modularity quantifies the quality of an assignment of nodes to communities by evaluating how much more densely connected the nodes within a community are, compared to how connected they would be in a random network.

The method consists of repeated application of two steps. The first step is a "greedy" assignment of nodes to communities, favoring local optimizations of modularity. The second step is the definition of a new coarse-grained network, based on the communities found in the first step. These two steps are repeated until no further modularity-increasing reassignments of communities are possible.



Twelve clusters (also called “communities”) are resulted from Louvain algorithm. The distribution of cluster is far more balanced compared to k-means method. Remind that the goal of the analysis is to reveal the topic of tweets; therefore distribution of tweets to clusters should not be an important metric. The obtained modularity for this clustering is 0.39068.

5.4 Topic of every cluster

	cluster1	**cluster2**	**cluster3**	**cluster4**	**cluster5**	**cluster6**
Most important words	sport	dit	video	fait	depui	tout
	match	connexion	bonn	erreur	compt	comm
	chaîn	svp	personn	déjà	internet	autr
	march	internet	bonsoir	téléphone	factur	quoi
	fair	pass	quoi	connexion	toujour	temp

	cluster7	**cluster8**	**cluster9**	**cluster10**	**cluster11**	**cluster12**
Most important words	fibr	quand	servic	bien	box	bon
	oui	faut	client	non	décodeur	problèm
	merd	souci	pire	vraiment	forfait	réseau
	pourquoi	nest	opérateur	foi	imposs	avoir
	orang	quil	donc	oui	moi	mobil

The above table depicts the topic of tweets clusters resulting from Louvain algorithm. The cluster1 is quite similar to cluster1 of k-means, where both mention about “*sport*”. Similarly, cluster12 in Louvain algorithm is comparable to cluster6 of k-means, where both speak about “*réseau*” and “*problèm*”.

On the other hand, Louvain model is able to show some topics that are not included in k-mean such as cluster2 and cluster4 in which they mention about internet connection and telephone error, respectively.

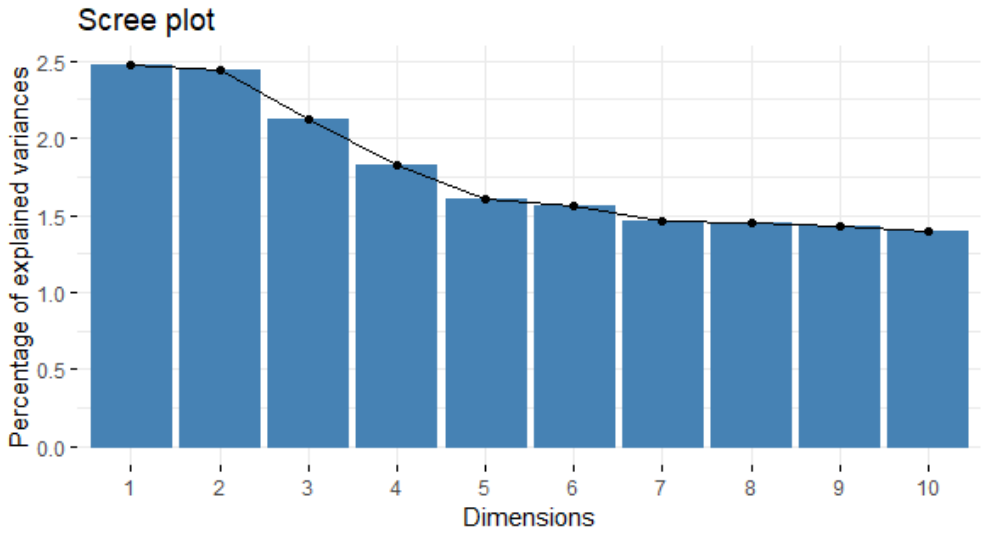
6. Conclusion

This project performs text mining and graph analysis on tweets of SFR, a large French corporation. Two main employed approaches are k-means and Louvain method to discover the topic of those tweets.

It is not surprised that negative tweets about SFR is more common than positive ones because people tend to report on social network when they have a problem. On one hand, Louvain method generates more clusters of tweets and, hence, is able to reveal more possible topics than k-means. However, the difference of those clusters are not confirmed. Whereas, cluster resulting from k-means seem to be more distinct. On the other hand, both approaches provide some alike clusters.

Appendix

Principal component analysis



```
###                 Text Mining and Graph Application on SFR Tweets
###                                 April 2019

###               Khanh TRUONG - Mickaël OLIFANT - Clarisse BOINAY

# Import packages ---------------------------------------------------------------
library(knitr)        # used to make kable tables
library(tm)           # text mining package
library(magrittr)     # allows pipe operator
library(tidytext)     # tidy text for plots
library(ggplot2)      # used for plots
library(gridExtra)    # arrange plots
library(factoextra)   # used for plots PCA
library(NbClust)      # optimal k in kmeans
library(dplyr)        # Manipulate data frames
library(usedist)      # calcualate pairwise similarity
library(reshape2)     # for reshape similarity matrix

theme_update(plot.title = element_text(hjust = 0.5)) #adjust the tittle of plots


#fFr the graph part :
library(RColorBrewer)
library(microbenchmark)
library(igraph)


# Import files ------------------------------------------------------------------
temp = list.files("Parsing") # all files in folder Parsing
temp = paste0('Parsing/', temp) # add Parsing/ before the names' files
tweet_list = lapply(temp, read.delim, encoding='UTF-8') # import all the files
summary(tweet_list)
rm(temp)

# Merge data
tweet = data.frame()
for (i in 1:length(tweet_list)){

  tweet_i = data.frame(tweet_list[i])[c('text', 'lang',
                                        'entities_user_mentions_name')]
  tweet = rbind(tweet, tweet_i)

  rm(i, tweet_i)
}

# Take only sfr French tweets
tweet_fr = tweet[tweet$lang=='fr', ]
sfr = tweet_fr[grep("sfr",
                    tweet_fr$entities_user_mentions_name,
                    ignore.case=TRUE), ]
dim(sfr)


# Clean text data ---------------------------------------------------------------
str(sfr$text) # data type of column 'text' is factor
sfr$text = as.character(sfr$text) # change to character
str(sfr$text) # check again

# Use regular expression to clean text

sfr = sfr[!grepl("^RT", sfr$text, ignore.case = TRUE),] #remove retweeted tweets
rownames(sfr) <- NULL # reset index

old = sfr[17, ]$text # take an example tweet

sfr$text <- gsub("#\\w+ *", "", sfr$text) # remove hash-tag (start by "#")
old; sfr[17, ]$text
```

```r
sfr$text <- gsub("@\\w+ *", "", sfr$text) # remove entities (start by "@")
old; sfr[17, ]$text

sfr$text <- gsub('http\\S+\\s*', '', sfr$text) # remove link (start by "http")
old; sfr[17, ]$text

rm(old)


# Corpus Transformations ------------------------------------------------------
sfr_corpus <- VCorpus(VectorSource(sfr$text)) # transform in to corpus
sfr_corpus

getTransformations() # all transformations that tm supports

# take an example tweet to see effects of transformation
old = strwrap(sfr_corpus[[8]])

# remove punctuation
sfr_corpus <- tm_map(sfr_corpus, removePunctuation)
old; strwrap(sfr_corpus[[8]])

# remove number
sfr_corpus <- tm_map(sfr_corpus, removeNumbers)
old; strwrap(sfr_corpus[[8]])

# lowercase
sfr_corpus <- tm_map(sfr_corpus, content_transformer(tolower))
old; strwrap(sfr_corpus[[8]])

# remove stopwords
stop_word_add = c('sfr', 'chez', 'cest', 'plus', 'via', 'jai', 'merci',
                  'bonjour', 'alor', 'aussi', 'avant', 'aprè', 'tous' ,'fai')
sfr_corpus <- tm_map(sfr_corpus, removeWords,
                     c(stopwords("french"), stop_word_add))
old; strwrap(sfr_corpus[[8]])

# stemming
sfr_corpus <- tm_map(sfr_corpus, stemDocument, language = 'french')
old; strwrap(sfr_corpus[[8]])


# strip extra whitespace
sfr_corpus <- tm_map(sfr_corpus, stripWhitespace)
old; strwrap(sfr_corpus[[8]])

rm(old)


# Document Term Matrix --------------------------------------------------------
# Coerces into a Document Term Matrix
# At the same time, remove words that appear in fewer than 1% of the data
# Those words may be special characters that
# haven't cleaned yet in corpus transformation
sfr_dtm <- DocumentTermMatrix(
  sfr_corpus,
  control=list(bounds = list(global = c(0.01*nrow(sfr), Inf))))

rowTotals <- apply(sfr_dtm, 1, sum) # sum of remaining words in each tweet
sfr_dtm <- sfr_dtm[rowTotals > 0, ] # remove all tweets without words
rm(rowTotals)

dim(sfr_dtm)
inspect(sfr_dtm[10:15, 100:105]) # inspects
```

```r
# Word Frequency ------------------------------------------------------------
# Sum all columns(words) to get frequency
words_frequency <- colSums(as.matrix(sfr_dtm))
ord <- order(words_frequency, decreasing=TRUE)

# get the top 10 words by frequency of appearance
words_frequency[head(ord, 10)] %>%
  kable()

# Find words that are most correlated with "client"
findAssocs(sfr_dtm, "client", .03) %>%
  kable()

# Find words that are most correlated with "servic"
findAssocs(sfr_dtm, "servic", .04) %>%
  kable()

# Find words that are most correlated with "réseau"
findAssocs(sfr_dtm, "réseau", 0.04) %>%
  kable()

# Find words that are most correlated with "box"
findAssocs(sfr_dtm, "box", 0.05) %>%
  kable()

# Find words that are most correlated with "fibr"
findAssocs(sfr_dtm, "fibr", 0.03) %>%
  kable()

# Find words that are most correlated with "téléphone"
findAssocs(sfr_dtm, "téléphone", 0.03) %>%
  kable()

# Find words that are most correlated with "mobil"
findAssocs(sfr_dtm, "mobil", 0.05) %>%
  kable()


# TF:IDF --------------------------------------------------------------------
# convert our "clean" corpus into a tfidf weighted dtm
sfr_dtm_tfidf = DocumentTermMatrix(
  sfr_corpus,
  control=list(bounds = list(global = c(0.01*nrow(sfr),Inf)),
               weighting = weightTfIdf))

# sum of remaining words in each tweet
rowTotals_tfidf <- apply(sfr_dtm_tfidf, 1, sum)

# remove all tweets without words
sfr_dtm_tfidf <- sfr_dtm_tfidf[rowTotals_tfidf > 0,]
rm(rowTotals_tfidf)

dim(sfr_dtm_tfidf)
inspect(sfr_dtm_tfidf[1:5, 1:5]) # inspects


# Convert to data frame -----------------------------------------------------
sfr_tf = as.data.frame(as.matrix(sfr_dtm))
sfr_tfidf = as.data.frame(as.matrix(sfr_dtm_tfidf))


# K-mean clustering ---------------------------------------------------------
# Find optimal k
## Elbow method
set.seed(1234)
elbow = fviz_nbclust(sfr_tfidf, kmeans, method = "wss") +
  geom_vline(xintercept = 6, linetype = 2, color='blue')+
```

```
  labs(subtitle = NULL)
elbow
optimal_k = 6

# Perform clustering with optimal k
set.seed(1234)
cluster_kmean <- kmeans(sfr_tfidf, optimal_k)

cluster_kmean$size
cluster_kmean$size/nrow(sfr_tfidf)

# Visualize on PCA plane
pca = prcomp(sfr_tfidf, center=FALSE, scale = FALSE)
fviz_eig(pca)


ggplot() +
  geom_point(aes(x = pca$x[,1],
                 y = pca$x[,2],
                 color = as.factor(cluster_kmean$cluster))) +
  labs(x='First Principal Component', y='Second Principal Component',
       title ='K-Means Clustering') +
  guides(color=guide_legend(title="Cluster"))


# Topic of each cluster ------------------------------------------------------

# add cluster to sfr_tfidf
sfr_tfidf['cluster'] <- as.factor(cluster_kmean$cluster)

# for each cluster, calculate every column mean
# the mean of columns represent how 'important' and 'representing' of the word
topic = sfr_tfidf %>%
  group_by(cluster) %>%
  summarise_all(mean)
sfr_tfidf['cluster'] <- NULL # give back the original sfr_tfidf

# 6 lines of code below are for data transformation, no need to read, just run
topic = as.data.frame(t(topic))
colnames(topic) <- c('cluster1', 'cluster2', 'cluster3', 'cluster4', 'cluster5',
                     'cluster6')
topic = topic[-1, ]
topic['word'] <- rownames(topic)
topic[names(topic)!='word'] = data.frame(apply(topic[names(topic)!='word'],
                                               2,
                                               as.numeric))
rownames(topic) <- NULL # reset index

head(topic) # each row: average tf-idf score of a word in each cluster

# For each cluster, take 5 words have the highest tf-ipair_dist_list scores.
topic = data.frame(
  cluster1=topic[order(-topic$cluster1), ][1:5, 'word'],
  cluster2=topic[order(-topic$cluster2), ][1:5, 'word'],
  cluster3=topic[order(-topic$cluster3), ][1:5, 'word'],
  cluster4=topic[order(-topic$cluster4), ][1:5, 'word'],
  cluster5=topic[order(-topic$cluster5), ][1:5, 'word'],
  cluster6=topic[order(-topic$cluster6), ][1:5, 'word']
)
topic


# Louvain algorithm ----------------------------------------------------------
# create a functioin calculate the similarity score between 2 nodes
# it is equal to sum of all (min value in every column)
similar_score <- function (v1, v2) sum(mapply(min, v1, v2))
```

```
# it takes about 30 mins to execute the dist_make() function
start.time <- Sys.time()

# due to limited resources (not powerful private laptop)
# we analyze 1000 tweets (representing 1000 nodes), instead of ~5,000 tweets
n = 1000
set.seed(1234)
sfr_tfidf_sub = sample_n(sfr_tfidf, n)

pair_dist = dist_make(sfr_tfidf_sub,
                      similar_score, method = NULL)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken # time of execution
rm(time.taken)

# Transform the pairwise similarity into edges list
pair_dist_list = melt(as.matrix(pair_dist))

# remove edges that have value 0
# value 0 indicates ther is no common words between 2 tweets
pair_dist_list = pair_dist_list[pair_dist_list$value>0, ]


# Create the graph object
sfr_net = graph_from_edgelist(as.matrix(pair_dist_list[,1:2]), directed = FALSE)
E(sfr_net)$weight <- pair_dist_list[ ,'value'] # add weight to the graph
sfr_net
vcount(sfr_net)
ecount(sfr_net)
is.connected(sfr_net)

# Descriptive statistics
head(V(sfr_net))
head(E(sfr_net))
edge_density(sfr_net, loops=FALSE)
transitivity(sfr_net)
diameter(sfr_net, weights = pair_dist_list[, 'value'])
radius(sfr_net)
girth(sfr_net)

hist(pair_dist_list[, 'value'], main = "Histogram of weights", xlab = "Weights")
summary(pair_dist_list[,3])

# Louvain clustering
sfr_louvain <- cluster_louvain(sfr_net, weights = pair_dist_list[, 'value'])
modularity(sfr_louvain)
sizes(sfr_louvain)
modularity(sfr_louvain)


# Topic of each cluster in Louvain ----------------------------------------

# add cluster to sfr_tfidf
sfr_tfidf_sub['cluster'] <- as.factor(sfr_louvain$membership)

# for each cluster, calculate every column mean
# the mean of columns represent how 'important' and 'representing' of the word
topic_louvain = sfr_tfidf_sub %>%
  group_by(cluster) %>%
  summarise_all(mean)
sfr_tfidf_sub['cluster'] <- NULL # give back the original sfr_tfidf

# 6 lines of code below are for data transformation, no need to read, just run
topic_louvain = as.data.frame(t(topic_louvain))
colnames(topic_louvain) <-
  c('cluster1', 'cluster2', 'cluster3', 'cluster4', 'cluster5', 'cluster6',
```

```
    'cluster7', 'cluster8', 'cluster9', 'cluster10', 'cluster11', 'cluster12')

topic_louvain = topic_louvain[-1, ]
topic_louvain['word'] <- rownames(topic_louvain)
topic_louvain[names(topic_louvain)!='word'] <-
  data.frame(apply(topic_louvain[names(topic_louvain)!='word'],
                   2,
                   as.numeric))
rownames(topic_louvain) <- NULL # reset index

# each row: average tf-ipair_dist_list score of a word in each cluster
head(topic_louvain)

# For each cluster, take 5 words have the highest tf-ipair_dist_list scores.
topic_louvain = data.frame(
  cluster1=topic_louvain[order(-topic_louvain$cluster1), ][1:5, 'word'],
  cluster2=topic_louvain[order(-topic_louvain$cluster2), ][1:5, 'word'],
  cluster3=topic_louvain[order(-topic_louvain$cluster3), ][1:5, 'word'],
  cluster4=topic_louvain[order(-topic_louvain$cluster4), ][1:5, 'word'],
  cluster5=topic_louvain[order(-topic_louvain$cluster5), ][1:5, 'word'],
  cluster6=topic_louvain[order(-topic_louvain$cluster6), ][1:5, 'word'],
  cluster7=topic_louvain[order(-topic_louvain$cluster7), ][1:5, 'word'],
  cluster8=topic_louvain[order(-topic_louvain$cluster8), ][1:5, 'word'],
  cluster9=topic_louvain[order(-topic_louvain$cluster9), ][1:5, 'word'],
  cluster10=topic_louvain[order(-topic_louvain$cluster10), ][1:5, 'word'],
  cluster11=topic_louvain[order(-topic_louvain$cluster11), ][1:5, 'word'],
  cluster12=topic_louvain[order(-topic_louvain$cluster12), ][1:5, 'word']
)
topic_louvain
```