Text Analysis - How To

The main references I have used are outlined below. All feature good examples of some of the analysis I did for the seminar and previous Natural Language Processing (NLP) projects.

1. https://towardsdatascience.com/r-packages-for-text-analysis-ad8d86684adb
2. https://www.tidytextmining.com/tidytext.html
3. https://acadgild.com/blog/text-mining-using-r
4. https://www.datacamp.com/community/tutorials/sentiment-analysis-R

Below are all the packages I have used for various text mining projects. Also, some of these are for machine learning models not featured in this document.

```
library(readtext)
library(magrittr)
library(assertr)
library(widyr)
library(tidyr)
library(ggplot2)
library(igraph)
library(ggraph)
library(reshape2)
library(wordcloud)
library(plyr)
library(text2vec)
library(tm)
library(readr)
library(SnowballC)
library(dplyr)
library(tidytext)
library(RSiteCatalyst)
library(proxy)
library(cluster)
library(fpc)
library(chron)
library(faraway)
library(MASS)
library(data.table)
library(splitstackshape)
library(recipes)
library(devtools)
library(textdata)
library(stringr)
library(leaps)
```

First bring in the sample set of data I sent and do some data munging to make it a workable data set. You'll need to change the directory for reading in the file.

```
########-change me!!-#########
comment_df<-read.csv(
"/Users/colemanstrickland/Documents/Coleman/Misc/NLP/LNK_Comments_Sample.csv"
,
header=T)
##############################

q1<-na.omit(as.character(comment_df$Q1))
q1r<-rep("q1", length(q1))
q2<-na.omit(as.character(comment_df$Q2))
q2r<-rep("q2", length(q2))
q3<-na.omit(as.character(comment_df$Q3))
q3r<-rep("q3", length(q3))
q4<-na.omit(as.character(comment_df$Q4))
q4r<-rep("q4", length(q4))
q5<-na.omit(as.character(comment_df$Q5))
q5r<-rep("q5", length(q5))
q6<-na.omit(as.character(comment_df$Q6))
q6r<-rep("q6", length(q6))
q7<-na.omit(as.character(comment_df$Q7))
q7r<-rep("q7", length(q7))

text_all<-data.frame(Text=c(q1,q2,q3,q4,q5,q6,q7),
                             Question=c(q1r,q2r,q3r,q4r,q5r,q6r,q7r))
#text_all
```

Next we need to create data structures from the various text mining packages. These can get confusing on how to build but the above tutorials really help.

```
text_lines<-tibble(line=1:length(text_all$Text),
                   text = as.character(text_all$Text))
#text_lines
```

Now we can start calculating sentiments. I used the 'afinn' lexicon for this but there are others (see my slides for details or see some of the extra code below).

```
#add sentiments
  #unnest each line to score individual words
all_lines <- text_lines  %>% ungroup() %>%
  unnest_tokens(word, text)

  #use the unnested words to score sentiments (don't score stopwords)
  #and put the lines back together to give an overal sentiment for each
response
s <- all_lines %>% filter(!word %in% stop_words$word) %>%
  inner_join(get_sentiments("afinn"), by="word") %>%
  group_by(line) %>%
```

```
  summarise(sentiment = sum(value)) %>% mutate(method="afinn")
#s

# below is doing the same thing but using the 'bing' lexicon
#use bing sentiment
s_bing <- all_lines %>% #filter(!word %in% stop_words$word) %>%
  inner_join(get_sentiments("bing"), by="word") %>%
  group_by(line) %>%
  count(line, index=line, sentiment) %>%
  spread(sentiment, n, fill=0) %>%
  mutate(sentiment = positive - negative) %>% summarise(sentiment =
sum(sentiment))
#s_bing
```

Now with each comment's sentiment scored, I build a dataframe to house the info. In the above s tibble, a comment with a sentiment of 0 does not appear in the data structure; therefore, I need to make sure those are appropriately set aside in my dataframe.

```
sentiment<-c()
count<-0
for (i in 1:length(text_all[,1])){
  if (i==s$line[i-count] & i<=length(s$line)){
    sentiment[i]<-s$sentiment[i-count]
  }
  else{
    sentiment[i]<-0
    count<-count+1
  }
}
text_all$sentiment<-sentiment
text_lines2<-tibble(line=1:length(text_all$Text),
          text = as.character(text_all$Text),
          sentiment = as.character(text_all$sentiment))
#text_all
```

Now we can build a wordcloud to give us an idea about the overall sentiment of the comments. I built my wordcloud using the bing lexicon.

```
y <- text_lines2 %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)

## Joining, by = "word"

#y

#most positive and negative words
y$sentiment<-NULL
bing_word_counts <- y %>%
  inner_join(get_sentiments("bing"), by="word") %>%
  count(word, sentiment, sort = TRUE)
```

```
#bing_word_counts

#build wordcloud
set.seed(1)
y %>%
  inner_join(get_sentiments("bing"), by="word") %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("red", "blue"),
                   max.words = 30)
```

The bigram network is fairly comlicated. We first obtain on the bigrams in the document (note I'm using the tibble that has all the stopwords removed, see code above for the variable y). Next, I separte the bigrams into two words to be used as my nodes. Then I filter and count the two word occurences. In the end I have a tibble that contains my word 1 and word 2 bigrams along with the number of times they occured in the document.

```
#bigram network (n=2)
comment_bigrams <-  text_lines2 %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)
#comment_bigrams
bigrams_separated <- comment_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")
```

```
#bigrams_separated
bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)
bigrams_filtered<-na.omit(bigrams_filtered)
#bigrams_filtered
# new bigram counts:
bigram_counts <- bigrams_filtered %>%
  count(word1, word2, sort = TRUE)
#bigram_counts
```

Now I need to get the average sentiment of the comment in which the bigram was used (I used the sentiments calculated from the 'afinn' lexicon at the beginning of this file). The algorithm below iterates through each bigram, finds the comment in which the bigram was used, stores the sentiment of each comment, and finally averages the sentiments when it has found all the occurences of the bigrams in the document. Once the dataframe is constructed, I apply the "negative/positive/neutral" sentiment scoring based on the bigram's average sentiment score. Finally, I place the dataframe in an graph object to be used for graphing.

```
#build a bigram dataframe containing the average sentiments of the bigrams
avg_sent_total<-c()
for (i in 1:length(bigram_counts$word1)){
  avg_sent<-c()
  for (j in 1:length(bigrams_filtered$sentiment)){
    if (bigram_counts$word1[i]==bigrams_filtered$word1[j] &
        bigram_counts$word2[i]==bigrams_filtered$word2[j]){
      avg_sent<-c(avg_sent, as.integer(bigrams_filtered$sentiment[j]))
    }
  }
  avg_sent_total[i]<-mean(avg_sent)
}

#score the average bigrame sentiments as pos/neg/neutral
bigram_counts$sentiment<-avg_sent_total
bigram_counts$s<-bigram_counts$sentiment
bigram_counts$s[bigram_counts$sentiment<0] = "negative sentiment"
bigram_counts$s[bigram_counts$sentiment>0] = "positive sentiment"
bigram_counts$s[bigram_counts$sentiment==0] = "neutral"
bigram_counts$Sentiment<-as.factor(bigram_counts$s)

#create graph object
bigram_graph <- bigram_counts %>%
  filter(n >= 2) %>%
  graph_from_data_frame()
```

Now I plot the network using ggplot's ggraph package:

```
set.seed(1233) #the seed determines how the graph looks
a <- grid::arrow(type = "closed", length = unit(.1, "inches"))
```

```
ggraph(bigram_graph, layout = "fr") +
  geom_edge_link(aes(edge_colour=Sentiment), show.legend = T,
                 arrow = a, end_cap = circle(.05, 'inches')) +
  geom_node_point(color = "grey", size = 3) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```