Text as Data, Problem Set 1 - Spring '24

Sharif Kazemi

2024-02-14

```
library(tm)
library(quanteda)
library(quanteda.textplots)
library(quanteda.textstats)
library(rvest)
library(tidyverse)
library(jsonlite)
library(ggcorrplot)

load("cnn_fox_corpus.rdata")
```

1 Tokenisation

1.1 Generate a tokenized version of the corpus where tokens are sentences. (Hint: look at the documentation for the tokens() function).

1.1.1 (a) Inspect the tokenized version of article number 4. How many sentences (tokens) does it have?

```
jan_tokens = tokens(cnn_fox_corpus, what = "sentence")

q1a_summary <- summary(jan_tokens[4])

q1a_summary
```

```
##       Length Class  Mode
## text4 257    -none- character
```

1.1.2 (b) What is the second sentence in article number 600?

```
q1b_test = jan_tokens[600]

q1b = q1b_test[[1]][2]

q1b
```

The second sentence in article 600 is: "Anyway, let not your heart be troubled, we'll always be independent, we are not the media mob."

2 Document Feature Matrices

2.1 Generate three different DFM versions from the CNN and Fox News corpus, drawing on various preprocessing steps we discussed in class. (Note: you do not have to use sentences as tokens, you can use individual words or n-grams).

2.1.1 (a) What are the dimensions of each DFM?

```r
jan_tokens_cleaned_words = tokens(cnn_fox_corpus, what = "word",
                                  remove_numbers = TRUE,
                                  remove_punct = TRUE,
                                  remove_url = TRUE,
                                  remove_symbols = TRUE)

jan_tokens_cleaned_words = tokens_tolower(jan_tokens_cleaned_words)

dfm_words_simple = dfm(jan_tokens_cleaned_words)

jan_tokens_cleaned_words_stem = tokens_wordstem(jan_tokens_cleaned_words,
                                                language = "en")

jan_tokens_cleaned_words_stem_nostop = tokens_select(
  jan_tokens_cleaned_words_stem, pattern = stopwords('en'),
  selection = 'remove')

dfm_words_complex = dfm(jan_tokens_cleaned_words_stem_nostop)

dfm_words_complex_freq = dfm_trim(dfm_words_complex,min_termfreq = 10)
```

```r
dim_dfm_words_simple <- dim(dfm_words_simple)
dim_dfm_words_complex <- dim(dfm_words_complex)
dim_dfm_words_complex_freq <- dim(dfm_words_complex_freq)
```

The dimensions for words are: “633, 8163”

```r
jan_tokens_cleaned_sentences = tokens(cnn_fox_corpus, what = "sentence",
                                      remove_numbers = TRUE,
                                      remove_punct = TRUE,
                                      remove_url = TRUE,
                                      remove_symbols = TRUE)

dfm_sentences = dfm(jan_tokens_cleaned_sentences)

jan_tokens_cleaned_trigrams = tokens_ngrams(jan_tokens_cleaned_words_stem_nostop
                                            , n = 3)

dfm_trigrams_complex = dfm(jan_tokens_cleaned_trigrams)

dim_dfm_sentences <- dim(dfm_sentences)
dim_dfm_trigrams <- dim(dfm_trigrams_complex)
```

The dimensions for trigrams are: "633, 1614969"

The dimensions for sentences are: "633, 222204"

2.1.2 (b) Inspect the three DFMs you generated in step 2a using tools learned in class. In your opinion, which preprocessing version is best for this corpus? Why?

```
summary(colSums(dfm_words_complex_freq))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    10.0    17.0    39.0   230.8   134.0 18116.0
```

```
quantile(colSums(dfm_words_complex_freq))
```

```
##    0%   25%   50%   75%  100% 
##    10    17    39   134 18116
```

```
topfeatures(dfm_words_complex_freq, n= 10)
```

```
##    go   now peopl    --  just  know   say   get think right 
## 18116 16110 15284 14528 13523 13219 12667 12348 11869 11566
```

While these words seem a bit random at first glance, I would argue that they convey the chaos of the broadcast that day. Having watched it unfold live, the overwhelming sense from broadcasters was confusion so words like 'think,' 'go,' and 'now' do a good job at transmitting that sense.

```
summary(colSums(dfm_sentences))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.000   1.000   1.000   1.079   1.000 848.000
```

```
quantile(colSums(dfm_sentences))
```

```
##   0%  25%  50%  75% 100% 
##    1    1    1    1  848
```

```
topfeatures(dfm_sentences, n= 10)
```

```
##                      dr.                     rep.               thank you. 
##                      848                      403                      400 
##                     sen.               all right. (begin video clip)  rep. 
##                      313                      223                      199 
##                   watch.             take a look.                      end 
##                      190                      183                      179 
## (begin video clip)  sen. 
##                      133
```

Most sentences seem to be about short responses and commentary, so it's not very useful to us.

```
summary(colSums(dfm_trigrams_complex))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    1.000    1.000    1.000    1.195    1.000 3547.000
```

```
quantile(colSums(dfm_trigrams_complex))
```

```
##   0%  25%  50%  75% 100%
##    1    1    1    1 3547
```

```
topfeatures(dfm_trigrams_complex, n= 10)
```

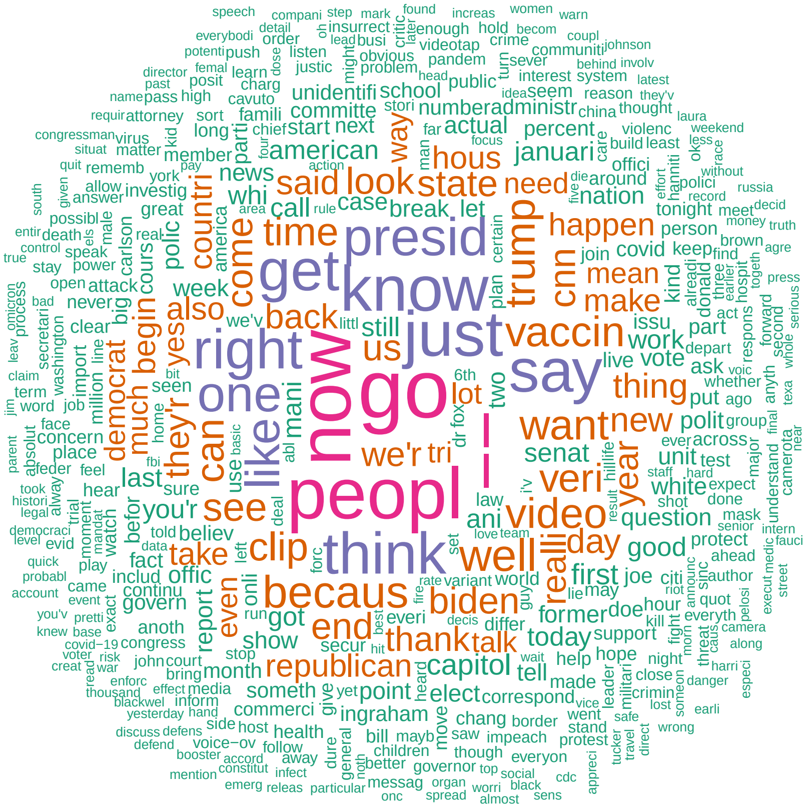
```
##      begin_video_clip        end_video_clip     presid_unit_state
##                  3547                  3538                   845
##   video_clip_ingraham       thank_veri_much video_clip_unidentifi
##                   673                   613                   432
##           join_us_now   former_presid_trump   next_commerci_break
##                   408                   406                   396
##    video_clip_carlson
##                   366
```

Trigrams seem to be similarly constrained by the mechanics of news broadcast, focusing on terms which guide the audience through the broadcast. Therefore, the usage is limited but we could spend some time removing phrases like 'video clip' and 'commercial break' to get more valuable material.

Overall, I'll go with the words dfm but I appreciate that trigrams and sentences might be more valuable long-term if we were to dedicate significant time in tailored cleaning.

2.1.3 (c) Create a wordcloud from your chosen DFM. Which word/phrase is the largest?

```
textplot_wordcloud(dfm_words_complex_freq, min_count = 20, random_order = FALSE,
                   rotation = .25,color = RColorBrewer::brewer.pal(4, "Dark2"))
```



3 Co-occurences

3.1 (a) The function dfm_subset() allows subsetting a DFM based on metadata. Use this function to generate two separate DFMs from your most preferred DFM from section 2b, where one DFM includes transcripts from CNN and the other from Fox News. What are the dimensions of each DFM?

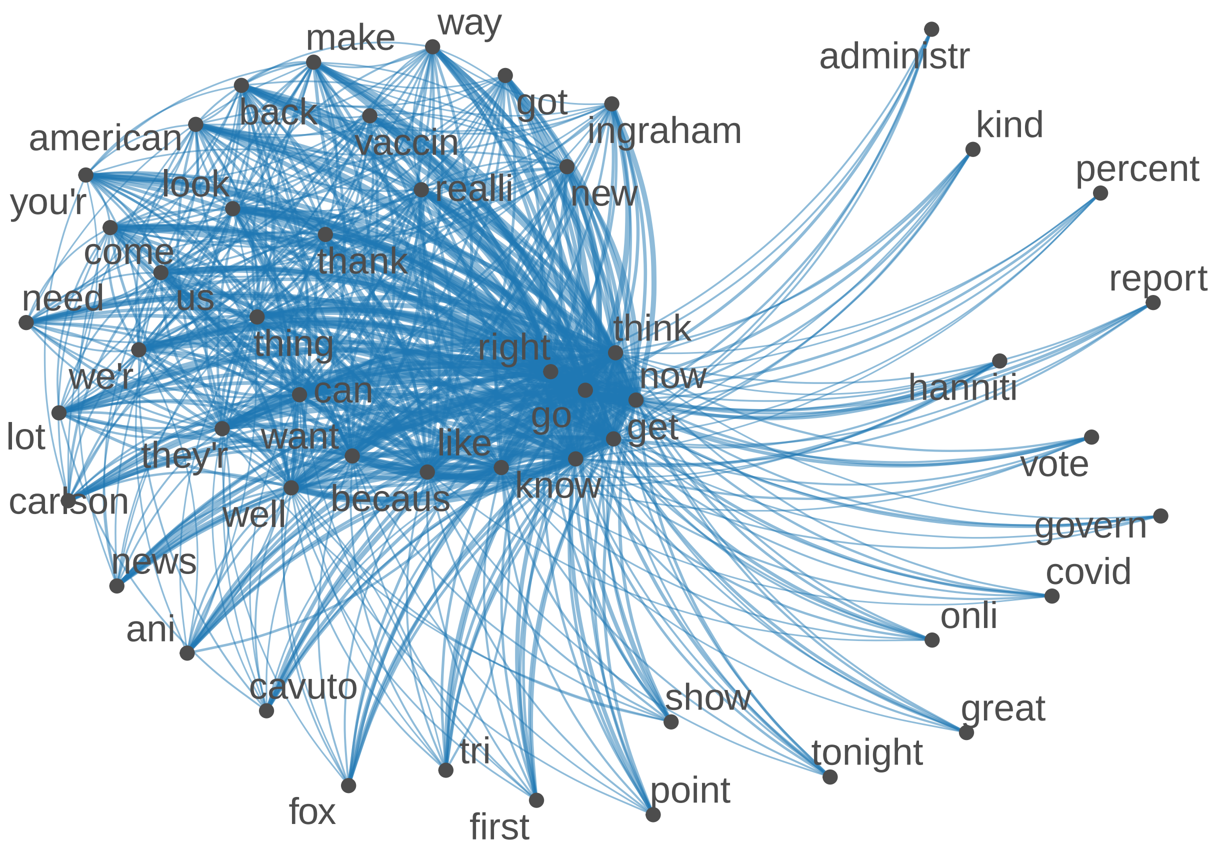
```r
# help(dfm_subset)

dfm_subset_cnn <- dfm_subset(dfm_words_complex_freq, Source == 'CNN')

dfm_subset_fox <- dfm_subset(dfm_words_complex_freq, Source == 'FOX')

dim_dfm_cnn <- dim(dfm_subset_cnn)
dim_dfm_fox <- dim(dfm_subset_fox)
```

The dimensions for CNN is “400, 8163”

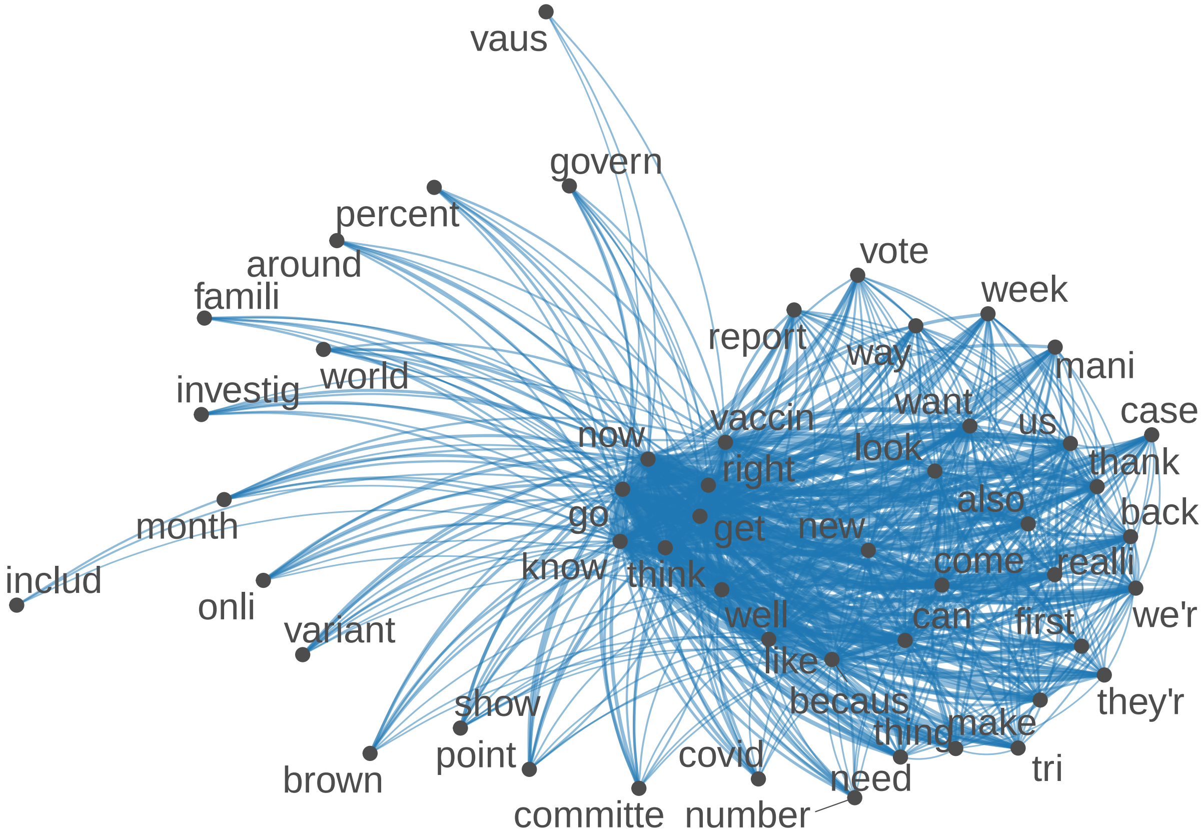
The dimensions for Fox is “233, 8163”

3.2 **(b) Generate a document level Feature Co-occurrence Matrix (FCM) from the Fox News DFM and create a figure that shows the relationship between the features. Which features have the highest co-occurrence? (Tip: calculating an FCM can be computationally intensive if the DFM from which it is derived has many dimensions. To speed up the calculation, make sure that your DFM has $< 10,000$ features.)**

```r
jan_fcm_fox = fcm(dfm_subset_fox, context = "document", count="frequency")

# trimmed earlier to only contain terms with minimum 10 frequencies

features_fox = names(topfeatures(jan_fcm_fox, 50))

fcm_select(jan_fcm_fox, pattern = features_fox) %>%
  textplot_network()
```



```r
jan_fcm_cnn = fcm(dfm_subset_cnn, context = "document", count="frequency")

features_cnn = names(topfeatures(jan_fcm_cnn, 50))

fcm_select(jan_fcm_cnn, pattern = features_cnn) %>%
  textplot_network()
```



It's interesting that a lot of the words in the centre which are linked are terms describing ongoing events with a degree of uncertainty. For example, 'think,' 'go', 'right', & 'now.'

The only major differences between the two are for terms that are more towards the outer edge of the graph, with the Fox News one showing a propensity for Anchor names like Carlson and Ingraham - whilst it gives lower priority to vaccine than CNN does.

4 Zipf Law

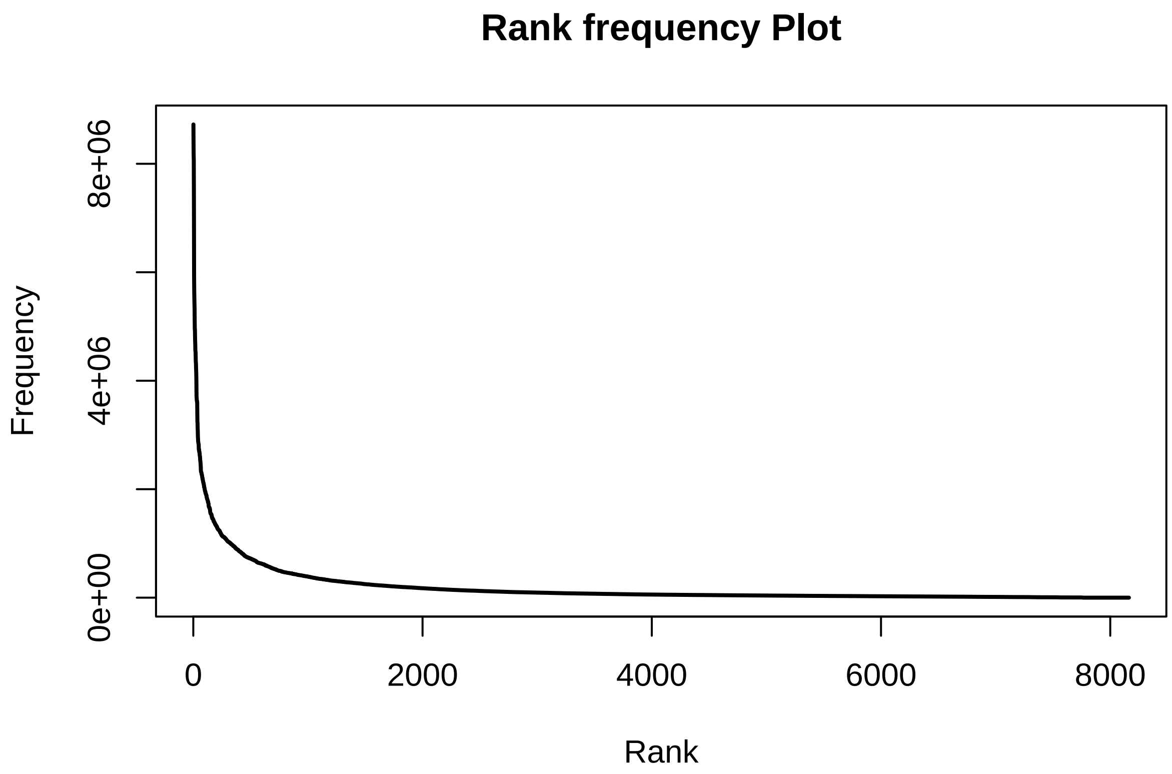
4.1 Generate rank frequency plots for the CNN and Fox News transcripts (Hint: you can use the FCMs that you generated in Question 3 for this purpose). Do they have a Zipfian distribution? What are the top 10 features for each source?

```
freqs_fox = colSums(jan_fcm_fox)

words_fox = colnames(jan_fcm_fox)

wordlist_fox = data.frame(words_fox, freqs_fox)

wordlist_fox = wordlist_fox[order(wordlist_fox[ ,"freqs_fox"], decreasing = TRUE), ]
```

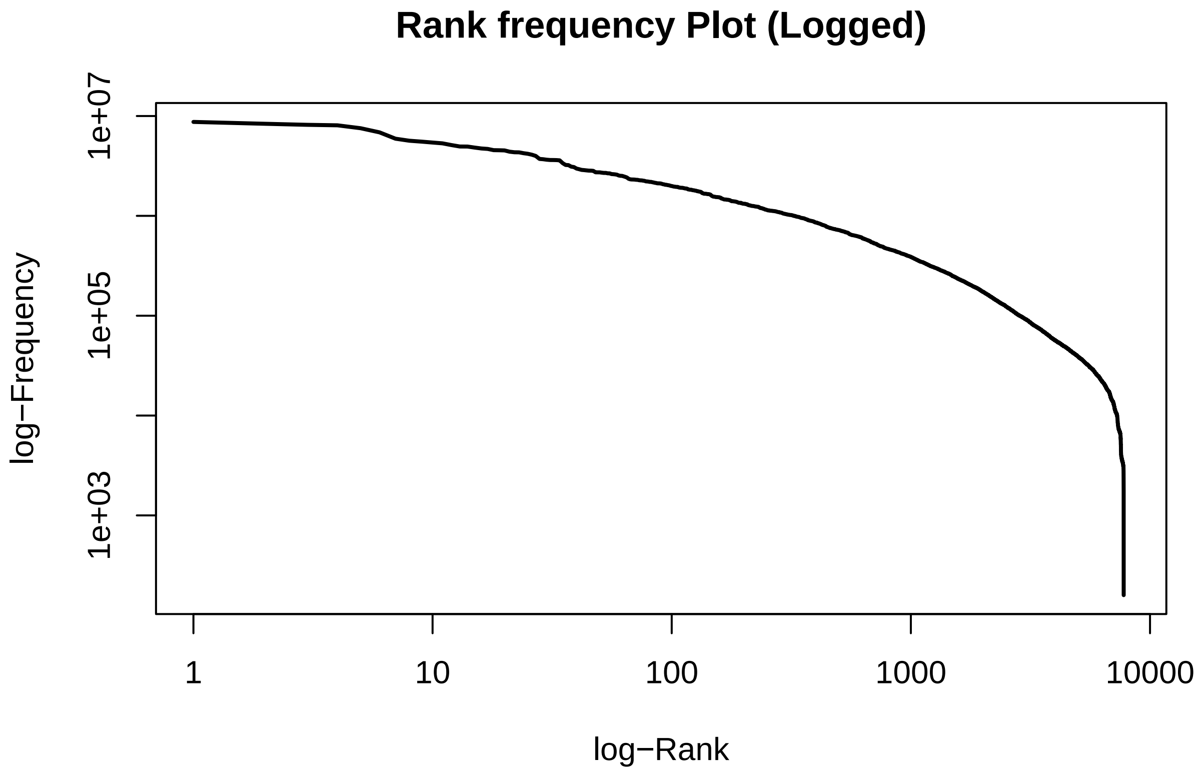
```
head(wordlist_fox, 10)
```

```
##           words_fox freqs_fox
## ingraham   ingraham   8724821
## like           like   8359516
## think         think   8161158
## know           know   8069689
## go               go   7546595
## get             get   6858778
## cavuto       cavuto   5924324
## they'r       they'r   5654151
## vaccin       vaccin   5537823
## carlson     carlson   5424570
```

```
# Plot the distribution. Does it look Zipfian?
plot(wordlist_fox$freqs_fox , type = "l", lwd=2,
     main = "Rank frequency Plot", xlab="Rank", ylab ="Frequency")
```

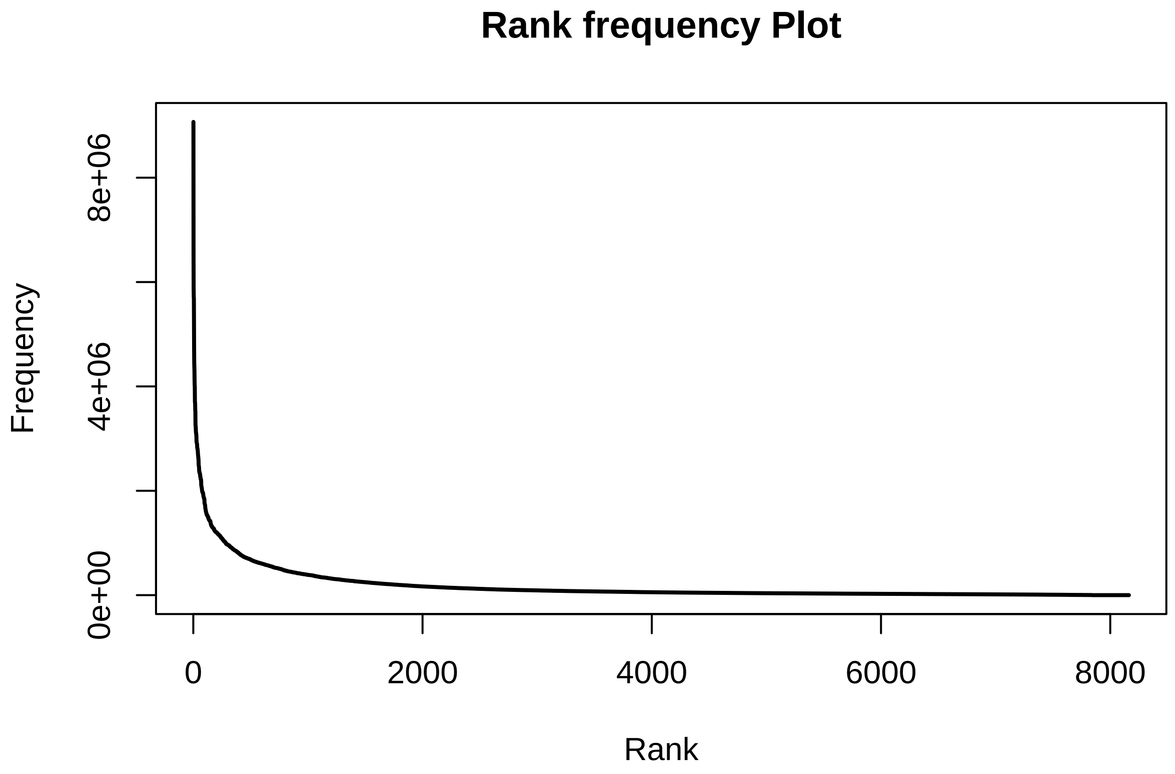


```
# Plot the logged distribution. Does it look like a line with slope -1?
plot(wordlist_fox$freqs_fox , type = "l", log="xy", lwd=2,
     main = "Rank frequency Plot (Logged)", xlab="log-Rank", ylab ="log-Frequency")
```

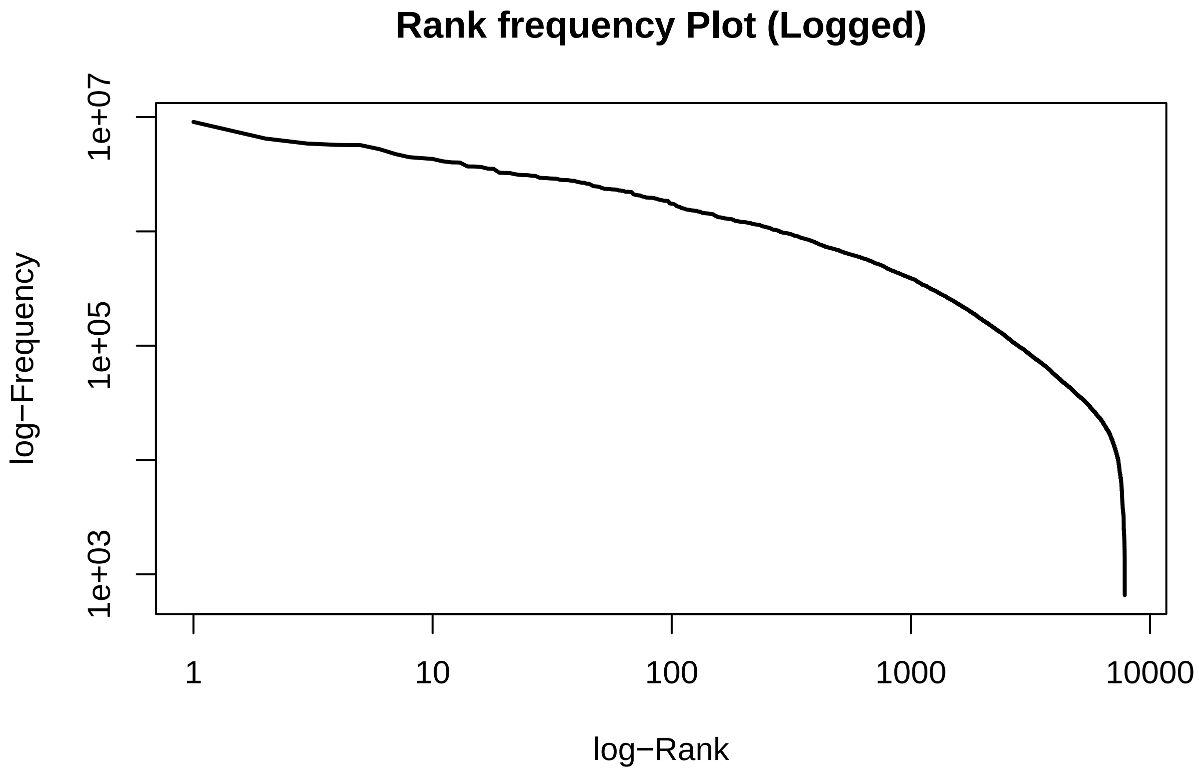


```r
freqs_cnn = colSums(jan_fcm_cnn)

words_cnn = colnames(jan_fcm_cnn)

wordlist_cnn = data.frame(words_cnn, freqs_cnn)

wordlist_cnn = wordlist_cnn[order(wordlist_cnn[ ,"freqs_cnn"], decreasing = TRUE), ]
```

```r
head(wordlist_cnn, 10)

# Plot the distribution. Does it look Zipfian?
plot(wordlist_cnn$freqs_cnn , type = "l", lwd=2,
     main = "Rank frequency Plot", xlab="Rank", ylab ="Frequency")
```



```
# Plot the logged distribution. Does it look like a line with slope -1?
plot(wordlist_cnn$freqs_cnn , type = "l", log="xy", lwd=2,
     main = "Rank frequency Plot (Logged)", xlab="log-Rank", ylab ="log-Frequency")
```



5 Web Scraping

5.1 Using the tools learned in class, scrape the content in the Wikipedia page describing the January 6th attack on the United States capitol. The URL of the page can be found here: https://en.wikipedia.org/wiki/January_6_United_States_Capitol_attack

5.1.1 (a) How many paragraphs of text does the page have?

```
wiki_jan = read_html("https://en.wikipedia.org/wiki/January_6_United_States_Capitol_attack") %>%
  html_nodes(".mw-parser-output p") %>% # only taking the paragraphs, using 'p'
  html_text(trim=T) %>%
  str_squish

summary(wiki_jan)

# 190 paragraphs of text but need to sort out empty ones

# Let's make sure we captured only relevant paragraphs
head(wiki_jan, n=5) # first 4  are irrelevant
tail(wiki_jan, n=5) # last 2 are irrelevant
```

```r
# So let's remove the first 4 and last 2 elements
wiki_jan <- wiki_jan[5:187]
length_wiki_jan <- length(wiki_jan)
```

Now we're left with 183 paragraphs.

5.1.2 (b) Preprocess the text using the tools learned in class. What the the top 5 features?

```r
jan_tokens_wiki = tokens(wiki_jan,  remove_numbers = TRUE,
                      remove_punct = TRUE, remove_url = TRUE,
                      remove_symbols = TRUE) %>%
  tokens_wordstem(language = "en") %>%
  tokens_select(pattern = stopwords('en'), selection = 'remove')

dfm_jan_tokens_wiki = dfm(jan_tokens_wiki)

jan_tokens_wiki_freq = dfm_trim(dfm_jan_tokens_wiki,min_termfreq = 10)

top_jan_tokens_wiki_freq <- topfeatures(jan_tokens_wiki_freq, 5)
```

top_jan_tokens_wiki_freq

Top five features are "capitol, trump, attack, polic, januari." Their frequencies are listed below in order: 189, 164, 127, 105, 90

5.1.3 (c) Write code to scrape the images in the article. How many images are there?

```r
wiki_jan_images = read_html("https://en.wikipedia.org/wiki/January_6_United_States_Capitol_attack") %>%
  html_nodes("div.mw-page-container img.mw-file-element") %>%
  html_attr("src")

length_wiki_jan_images <- length(wiki_jan_images)
```

Now we're left with 72 images.

6 Cosine Similarity

6.1 Now we will examine the similarity between the Wikipedia page that you scraped in Question 5 and the CNN and Fox News transcripts.

6.1.1 (a) Create a single document of text from the Wikipedia page (hint: you can use the paste() function with the collapse argument for this purpose).

```r
wiki_jan_text_compressed = paste(wiki_jan, collapse = " ")

wiki_jan_text_compressed
```

6.1.2 (b) Create a DFM from the Wikipedia text using the same preprocessing steps that you used for your preferred DFM in Question 2b. What are the dimensions of the Wikipedia DFM?

```r
jan_tokens_wiki_compressed = tokens(wiki_jan_text_compressed,
                         remove_numbers = TRUE,
                         remove_punct = TRUE, remove_url = TRUE,
                         remove_symbols = TRUE) %>%
  tokens_wordstem(language = "en") %>%
  tokens_select(pattern = stopwords('en'), selection = 'remove')

dfm_jan_wiki_compressed = dfm(jan_tokens_wiki_compressed)

dfm_jan_wiki_compressed_freq = dfm_trim(dfm_jan_wiki_compressed,min_termfreq = 10)

dim_dfm_jan_wiki_compressed_freq = dim(dfm_jan_wiki_compressed_freq)
```

The dimensions are 1, 205 - because there’s only 1 document, as the question requested from us, and there are 206 unique tokens after our pre-processing

6.1.3 (c) Use the textstat_simil() function to calculate the cosine similarity between the Wikipedia page and the CNN and Fox News transcripts. (Tip: in the function, set x to be equal to the DFM of the news transcripts and y to be equal to the DFM of the Wikipedia page). Which transcript is most similar to the Wikipedia page? (you can simply provide the document number). Can you tell if this is a CNN or Fox News transcript?

```r
# ?textstat_simil

cosine_similarities = textstat_simil(x = dfm_words_complex_freq,
                                     y = dfm_jan_wiki_compressed_freq,
                                     method = "cosine",
                                     margin = "documents")

dim(cosine_similarities)
max(cosine_similarities)

which(cosine_similarities@x == max(cosine_similarities@x), arr.ind = TRUE)

# The max cosine belongs to document 323

dfm_words_complex_freq$heading[[323]]
```

Just guessing, this seems like a very standard headline except for the use of the word ‘insurrection’ so it has negative connotations. Given that, it’s more likely to be CNN.

```r
dfm_words_complex_freq$Source[[323]]
```

```
## [1] "CNN"
```

We were right!