Task 2: Exploratory Data Analysis

*Rohan Hadvani*

*19/08/2020*

Instructions

The first step in building a predictive model for text is understanding the distribution and relationship between the words, tokens, and phrases in the text. The goal of this task is to understand the basic relationships you observe in the data and prepare to build your first linguistic models.

The goal of this project is just to display that you’ve gotten used to working with the data and that you are on track to create your prediction algorithm. Please submit a report on R Pubs that explains your exploratory analysis and your goals for the eventual app and algorithm. This document should be concise and explain only the major features of the data you have identified and briefly summarize your plans for creating the prediction algorithm and Shiny app in a way that would be understandable to a non-data scientist manager.

You should make use of tables and plots to illustrate important summaries of the data set. The motivation for this project is to: 1. Demonstrate that you’ve downloaded the data and have successfully loaded it in. 2. Create a basic report of summary statistics about the data sets. 3. Report any interesting findings that you amassed so far. 4. Get feedback on your plans for creating a prediction algorithm and Shiny app.

Tasks to accomplish

1. **Exploratory analysis** - perform a thorough exploratory analysis of the data, understanding the distribution of words and relationship between the words in the corpora.
2. **Understand frequencies of words and word pairs** - build figures and tables to understand variation in the frequencies of words and word pairs in the data.

Basic setup

Load libraries

**library**(quanteda)

**library**(readtext)

**library**(stringi)

**library**(ggplot2)

**library**(cowplot)

**library**(reshape2)

Read in data

1. Load data

**if**(!file.exists('./final/en\_US/en\_US.blogs.txt')){

download.file('https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip',

destfile = paste0(getwd(), 'Coursera-SwiftKey.zip'),

method = 'curl', quiet = T)

unzip('./Coursera-SwiftKey.zip')

}

rawBlogs <- readtext(paste0(getwd(), '/final/en\_US/en\_US.blogs.txt'))

rawNews <- readtext(paste0(getwd(), '/final/en\_US/en\_US.news.txt'))

rawTwts <- readtext(paste0(getwd(), '/final/en\_US/en\_US.twitter.txt'))

1. Calculate number of lines

lines <- data.frame('source' = c('blog', 'news', 'twitter'),

'lines' = c(stri\_count\_fixed(rawBlogs, '\n'),

stri\_count\_fixed(rawNews, '\n'),

stri\_count\_fixed(rawTwts, '\n')))

lines

## source lines

## 1 blog 899287

## 2 news 1010241

## 3 twitter 2360147

Create corpus

1. Make individual corpus file

corpBlogs <- corpus(rawBlogs)

docvars(corpBlogs, 'Source') <- 'blogs'

corpNews <- corpus(rawNews)

docvars(corpNews, 'Source') <- 'news'

corpTwts <- corpus(rawTwts)

docvars(corpTwts, 'Source') <- 'twitter'

1. Combine into one corpus

corpAll <- corpBlogs + corpNews + corpTwts

1. Remove unecesary files to free up memory

rm(rawBlogs, rawNews, rawTwts)

rm(corpBlogs, corpNews, corpTwts)

Data analysis

Basic summary

1. Get the summary data for the corpus

sum <- summary(corpAll)

sum

## Corpus consisting of 3 documents:

##

## Text Types Tokens Sentences doc\_id Source

## text1 482434 42840147 2077533 en\_US.blogs.txt blogs

## text11 431664 39918314 1868674 en\_US.news.txt news

## text12 566950 36719645 2598128 en\_US.twitter.txt twitter

##

## Source: Combination of corpuses corpBlogs + corpNews and corpTwts

## Created: Wed Mar 21 23:21:34 2018

## Notes:

1. Plot the words and sentences for each data source

word <- ggplot(data = sum, aes(x = Source, y = Tokens, fill = Source)) +

geom\_col() +

guides(fill = FALSE) +

scale\_y\_continuous(expand = c(0, 0)) +

ylab('Word counts')

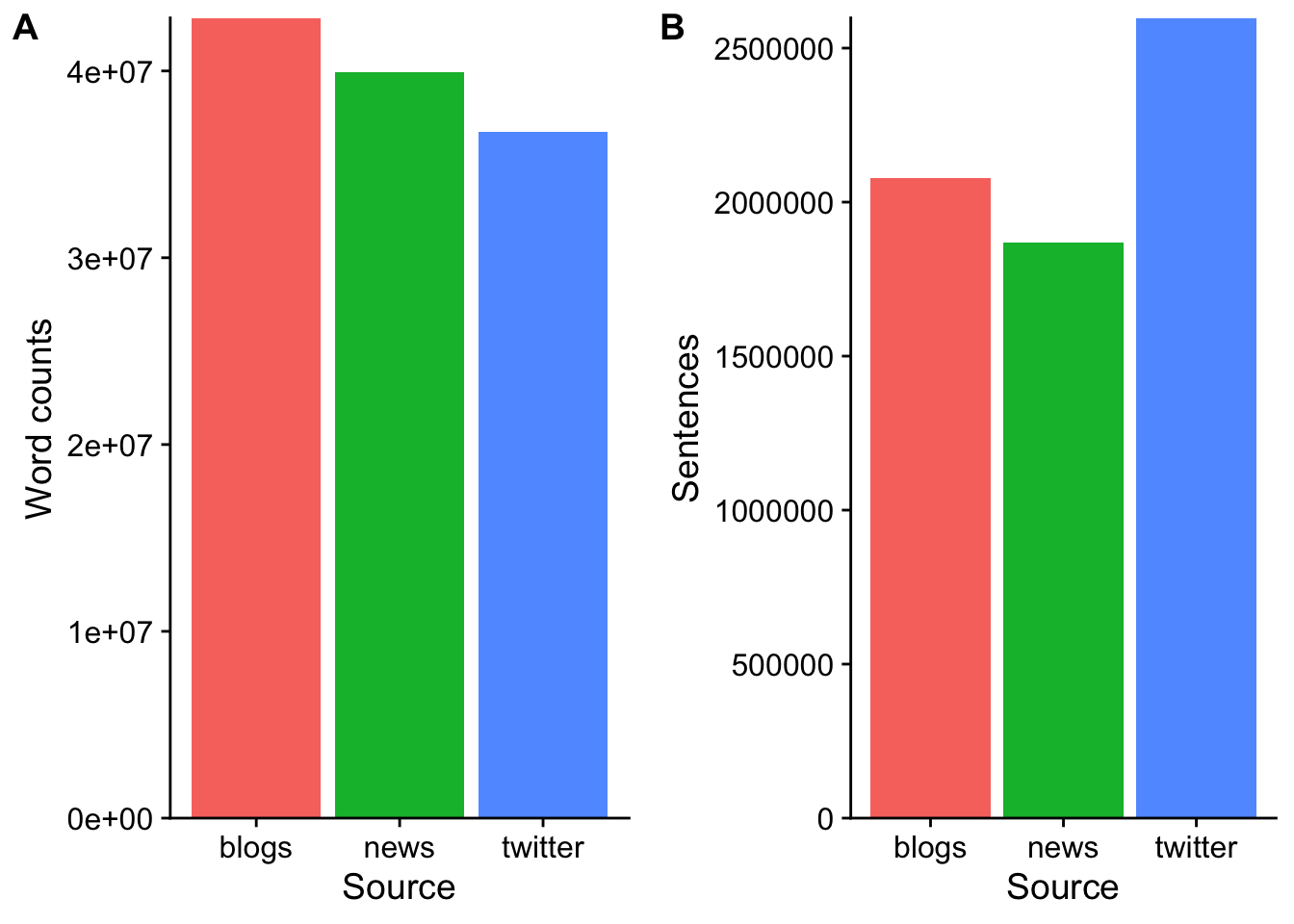
sentence <- ggplot(data = sum, aes(x = Source, y = Sentences, fill = Source)) +

geom\_col() +

scale\_y\_continuous(expand = c(0, 0)) +

guides(fill = FALSE)

plot\_grid(word, sentence, labels = 'AUTO')



Preprocessing

1. Write a function to tokenize the corpus and filter profanity

tokenization <- **function**(input, what = 'word', ngrams = 1L) {

*## This function calls the tokens function from quanteda*

*## takes an input (character, corpus, or token object)*

*## and returns the tokenized object*

*# step1: tokenize based on input values*

results <- tokens(x = input, what = what, ngrams = ngrams,

remove\_numbers = T, remove\_punct = T,

remove\_symbols = T, remove\_separators = T,

remove\_twitter = T, remove\_hyphens = T,

remove\_url = T)

*# step2: get a list of profanity*

**if** (!file.exists('badWords.txt')) {

download.file('https://raw.githubusercontent.com/shutterstock/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words/master/en',

dest = paste0(getwd(), 'badWords.txt'),

method = 'curl', quiet = T)

}

prof <- readLines('badWords.txt', skipNul = T)

*# step3: remove profanity*

results <- tokens\_remove(results, pattern = prof)

}

1. Tokenize words

tokWord <- tokenization(corpAll, what = 'word')

*#tokWord <- tokens\_tolower(tokWord)*

sumWord <- summary(tokWord)

sumWord

## Length Class Mode

## text1 37102051 -none- character

## text11 33853786 -none- character

## text12 29538860 -none- character

1. Tokenize sentences

tokSen <- tokenization(corpAll, what = 'sentence')

sumSen <- summary(tokSen)

sumSen

## Length Class Mode

## text1 990078 -none- character

## text11 726489 -none- character

## text12 1915673 -none- character

1. Plot the tokenized words and sentences for each data source

row.names(sumWord) <- c('blogs', 'news', 'twitter')

plotdf <- as.data.frame(sumWord[, 1])

tokWordP <- ggplot(data = plotdf, aes(x = row.names(sumWord), y = sumWord[, 1], fill = row.names(sumWord))) +

geom\_col() +

guides(fill = FALSE) +

xlab('Source') +

ylab('Word counts')

row.names(sumSen) <- c('blogs', 'news', 'twitter')

plotdf <- as.data.frame(sumSen[, 1])

tokSenP <- ggplot(data = sum, aes(x = row.names(sumSen), y = sumSen[, 1], fill = row.names(sumSen))) +

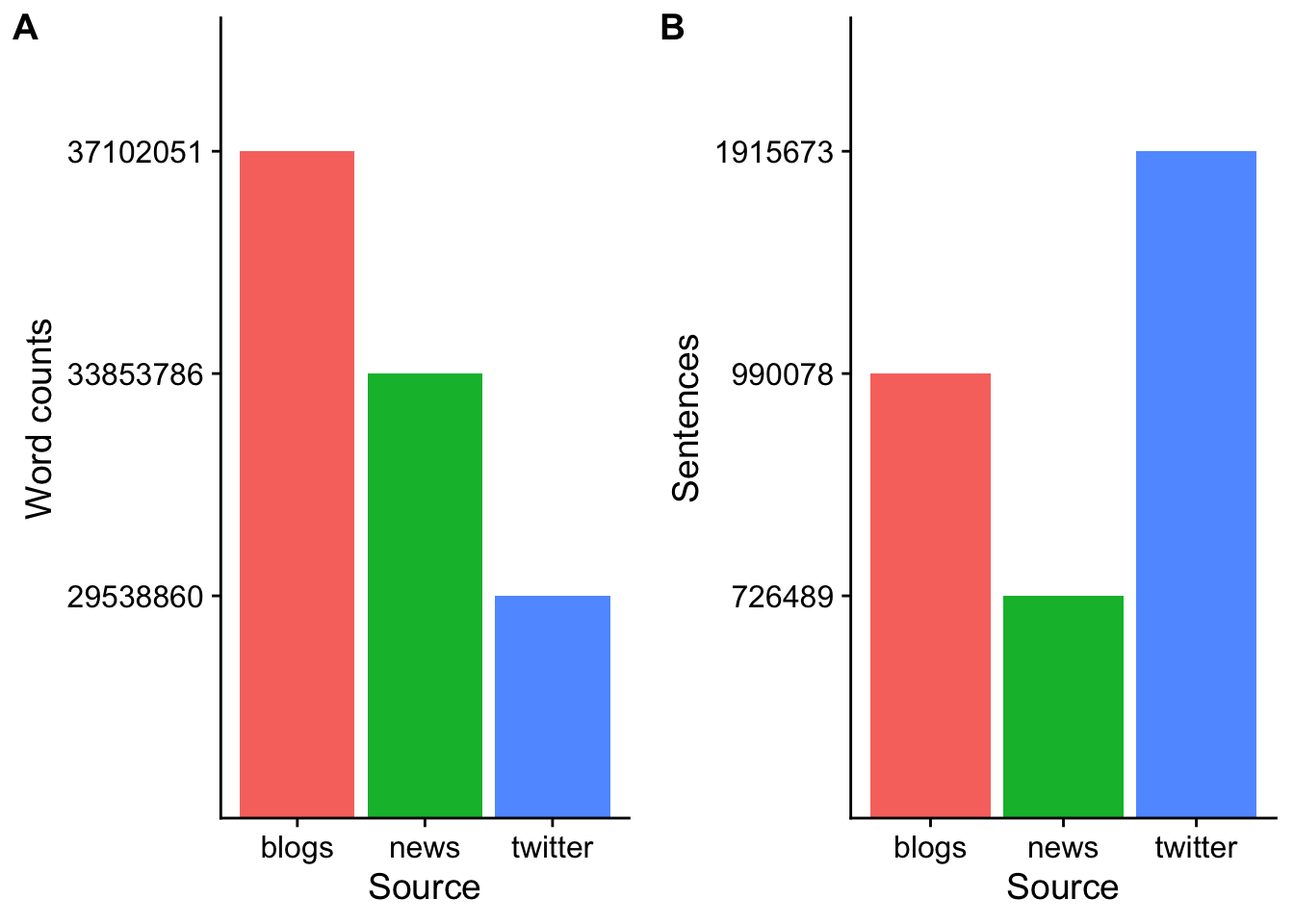
geom\_col() +

guides(fill = FALSE) +

xlab('Source') +

ylab('Sentences')

plot\_grid(tokWordP, tokSenP, labels = 'AUTO')



Distributions of word frequencies

Single words (1 gram)

* Make a document-feature matrix (dfm) object

dfmWord <- dfm(tokWord, tolower = T) *#make a dfm object*

* Calculate top 20 most frquent words and extract frequencies from individual source

topWordsAll <- topfeatures(dfmWord, n = 20)

*#find top 20 words in all sources*

topWordsBlog <- dfmWord[1, names(topWordsAll)]

*#extract frequency of the same 20 words from blogs*

topWordsNews <- dfmWord[2, names(topWordsAll)]

*#extract frequency of the same 20 words from news*

topWordsTwt <- dfmWord[3, names(topWordsAll)]

*#extract frequency of the same 20 words from twitter*

* Make a data frame for plotting

gram1dist <- as.data.frame(topWordsAll)

gram1dist <- cbind(gram1dist,

t(as.data.frame(topWordsBlog)[, -1]),

t(as.data.frame(topWordsNews)[, -1]),

t(as.data.frame(topWordsTwt)[, -1]))

colnames(gram1dist) <- c('all', 'blogs', 'news', 'twitter')

gram1dist$words <- row.names(gram1dist)

* Plot 1 gram distribution

df <- melt(gram1dist, id.vars = c('words', 'all')) *#convert to long format*

ggplot(data = df, aes(x = reorder(words, all), y = all)) +

geom\_col(aes(fill = variable)) +

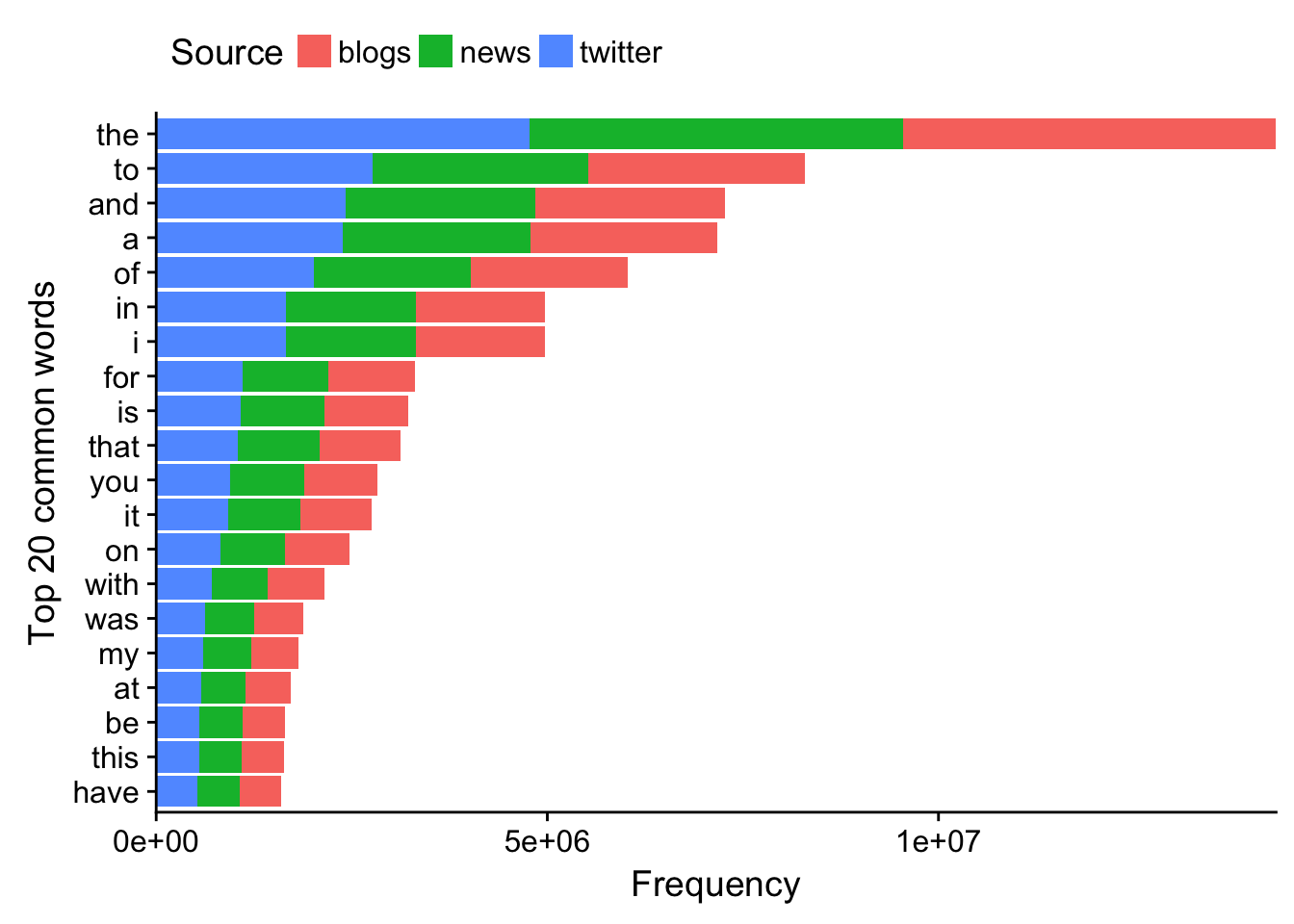
coord\_flip() +

ylab('Frequency') + xlab('Top 20 common words') +

scale\_y\_continuous(expand = c(0, 0)) +

guides(fill = guide\_legend(title = 'Source')) +

theme(legend.position = 'top')



* Plot the % of top 10 words across all sources

dfmWordPct <- dfm\_weight(dfmWord, scheme = 'prop') \* 100

dfWordPct <- data.frame(topfeatures(dfmWordPct))

colnames(dfWordPct) <- 'pct'

dfWordPct$words <- row.names(dfWordPct)

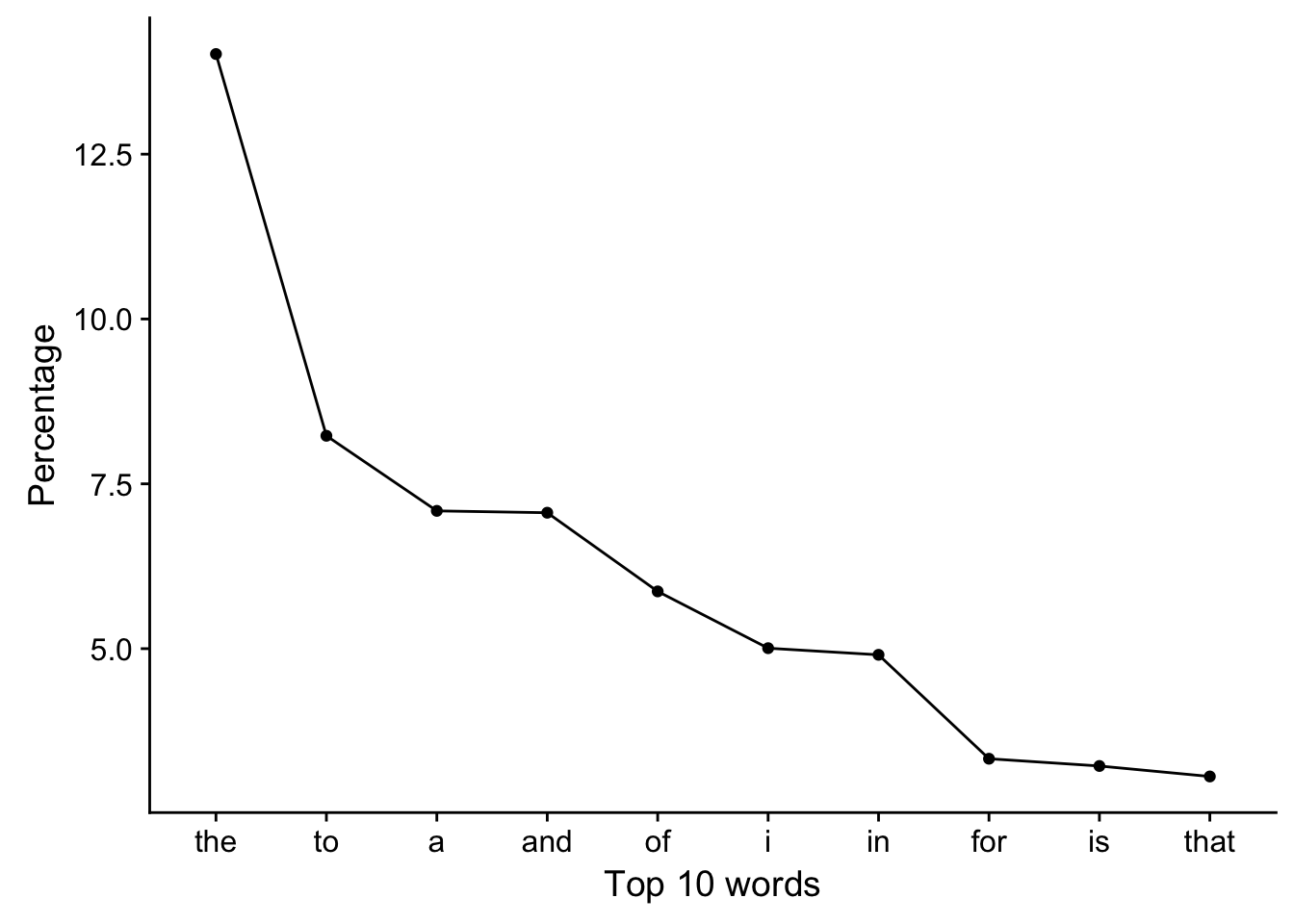
ggplot(data = dfWordPct, aes(x = words, y = pct, group = 1)) +

geom\_line() +

geom\_point() +

scale\_x\_discrete(limits = dfWordPct$words) +

xlab('Top 10 words') + ylab('Percentage')



Remove common words for 1 gram analysis

* The list above shows the common words in English. Remove the stopwords via remove = stopwords('english') and allow stem with stem = T for more flexible analysis

dfmWordTrim <- dfm(tokWord, tolower = T, stem = T, remove = stopwords('english'))

* Identify the top 20 most frquent words

*# Extract top 20 words from each source*

topWordsTrimAll <- topfeatures(dfmWordTrim, n = 20)

*#find top 20 words in all sources*

topWordsTrimBlog <- dfmWordTrim[1, names(topWordsTrimAll)]

*#extract frequency of the same 20 words from blogs*

topWordsTrimNews <- dfmWordTrim[2, names(topWordsTrimAll)]

*#extract frequency of the same 20 words from news*

topWordsTrimTwt <- dfmWordTrim[3, names(topWordsTrimAll)]

*#extract frequency of the same 20 words from twitter*

*# Make data frame*

gram1distT <- as.data.frame(topWordsTrimAll)

gram1distT <- cbind(gram1distT,

t(as.data.frame(topWordsTrimBlog)[, -1]),

t(as.data.frame(topWordsTrimNews)[, -1]),

t(as.data.frame(topWordsTrimTwt)[, -1]))

colnames(gram1distT) <- c('all', 'blogs', 'news', 'twitter')

gram1distT$words <- row.names(gram1distT)

*# Plot 1 gram distribution*

df <- melt(gram1distT, id.vars = c('words', 'all')) *#convert to long format*

ggplot(data = df, aes(x = reorder(words, all), y = all)) +

geom\_col(aes(fill = variable)) +

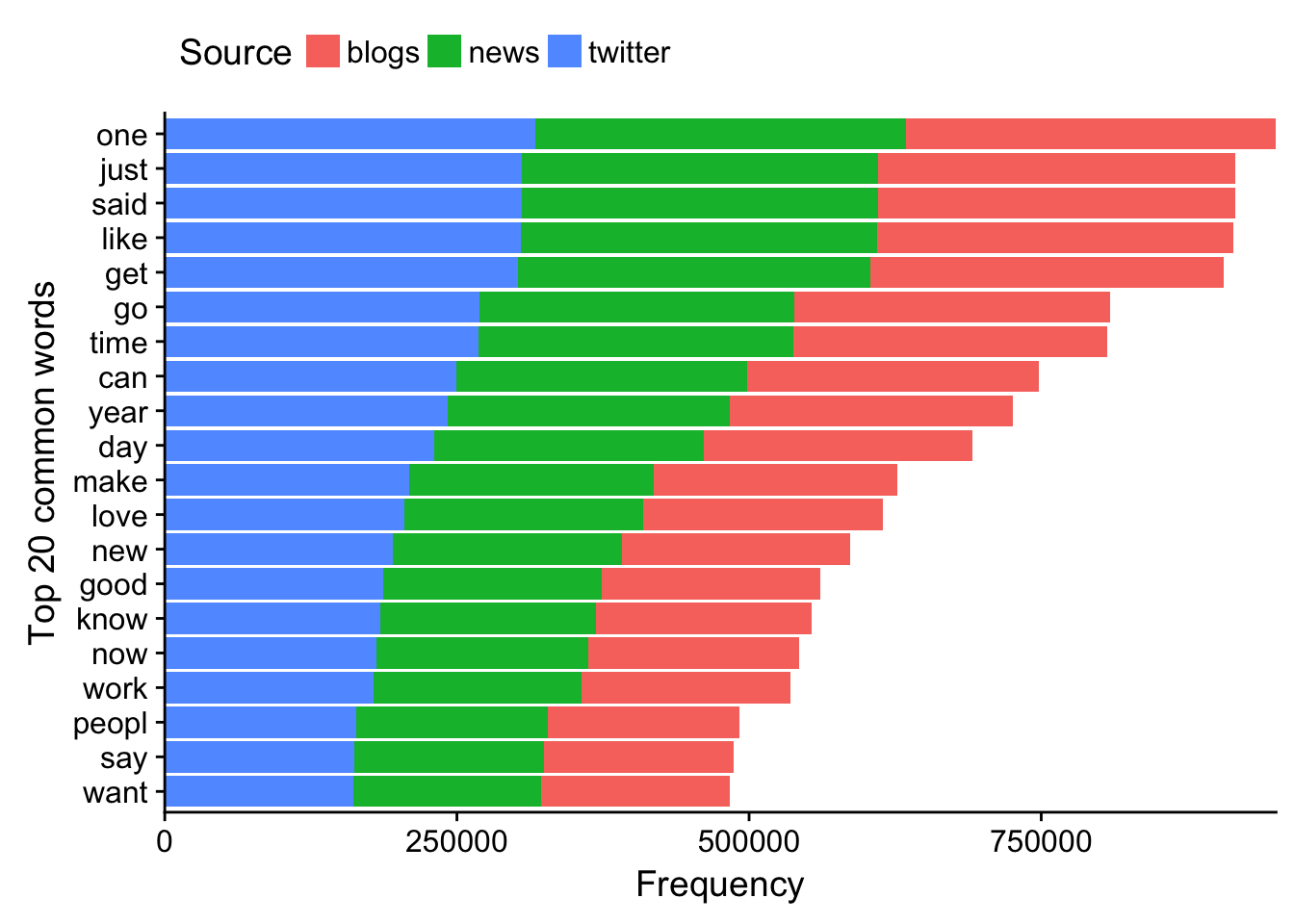
coord\_flip() +

ylab('Frequency') + xlab('Top 20 common words') +

scale\_y\_continuous(expand = c(0, 0)) +

guides(fill = guide\_legend(title = 'Source')) +

theme(legend.position = 'top')



* Plot the % of top 10 words across all sources

dfmWordTrimPct <- dfm\_weight(dfmWordTrim, scheme = 'prop') \* 100

dfWordTrimPct <- data.frame(topfeatures(dfmWordTrimPct))

colnames(dfWordTrimPct) <- 'pct'

dfWordTrimPct$words <- row.names(dfWordTrimPct)

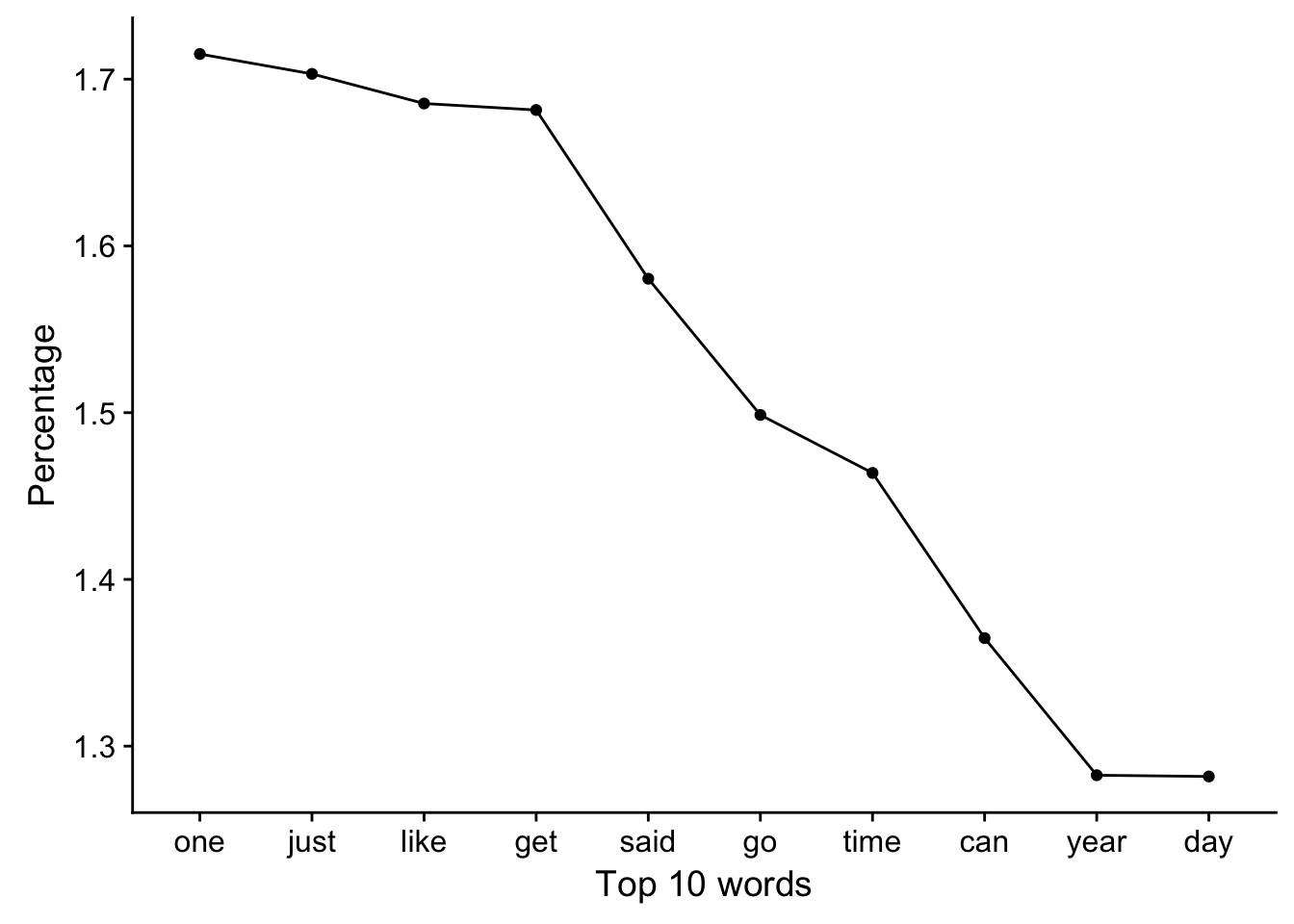
ggplot(data = dfWordTrimPct, aes(x = words, y = pct, group = 1)) +

geom\_line() +

geom\_point() +

scale\_x\_discrete(limits = dfWordTrimPct$words) +

xlab('Top 10 words') + ylab('Percentage')



2 grams

* Identify 2 grams and make a dfm object

tokWord2g <- tokens\_ngrams(tokWord, n = 2L, concatenator = ' ')

dfmWord2g <- dfm(tokWord2g, tolower = T)

* Calculate top 20 most frquent 2 grams and extract frequencies from individual source

top2gAll <- topfeatures(dfmWord2g, n = 20)

*#find top 20 2 grams in all sources*

top2gBlog <- dfmWord2g[1, names(top2gAll)]

*#extract frequency of the same 20 2 grams from blogs*

top2gNews <- dfmWord2g[2, names(top2gAll)]

*#extract frequency of the same 20 2 grams from news*

top2gTwt <- dfmWord2g[3, names(top2gAll)]

*#extract frequency of the same 20 2 grams from twitter*

* Make a data frame for plotting

gram2dist <- as.data.frame(top2gAll)

gram2dist <- cbind(gram2dist,

t(as.data.frame(top2gBlog)[, -1]),

t(as.data.frame(top2gNews)[, -1]),

t(as.data.frame(top2gTwt)[, -1]))

colnames(gram2dist) <- c('all', 'blogs', 'news', 'twitter')

gram2dist$words <- row.names(gram2dist)

* Plot 2 gram distribution

df <- melt(gram2dist, id.vars = c('words', 'all')) *#convert to long format*

ggplot(data = df, aes(x = reorder(words, all), y = all)) +

geom\_col(aes(fill = variable)) +

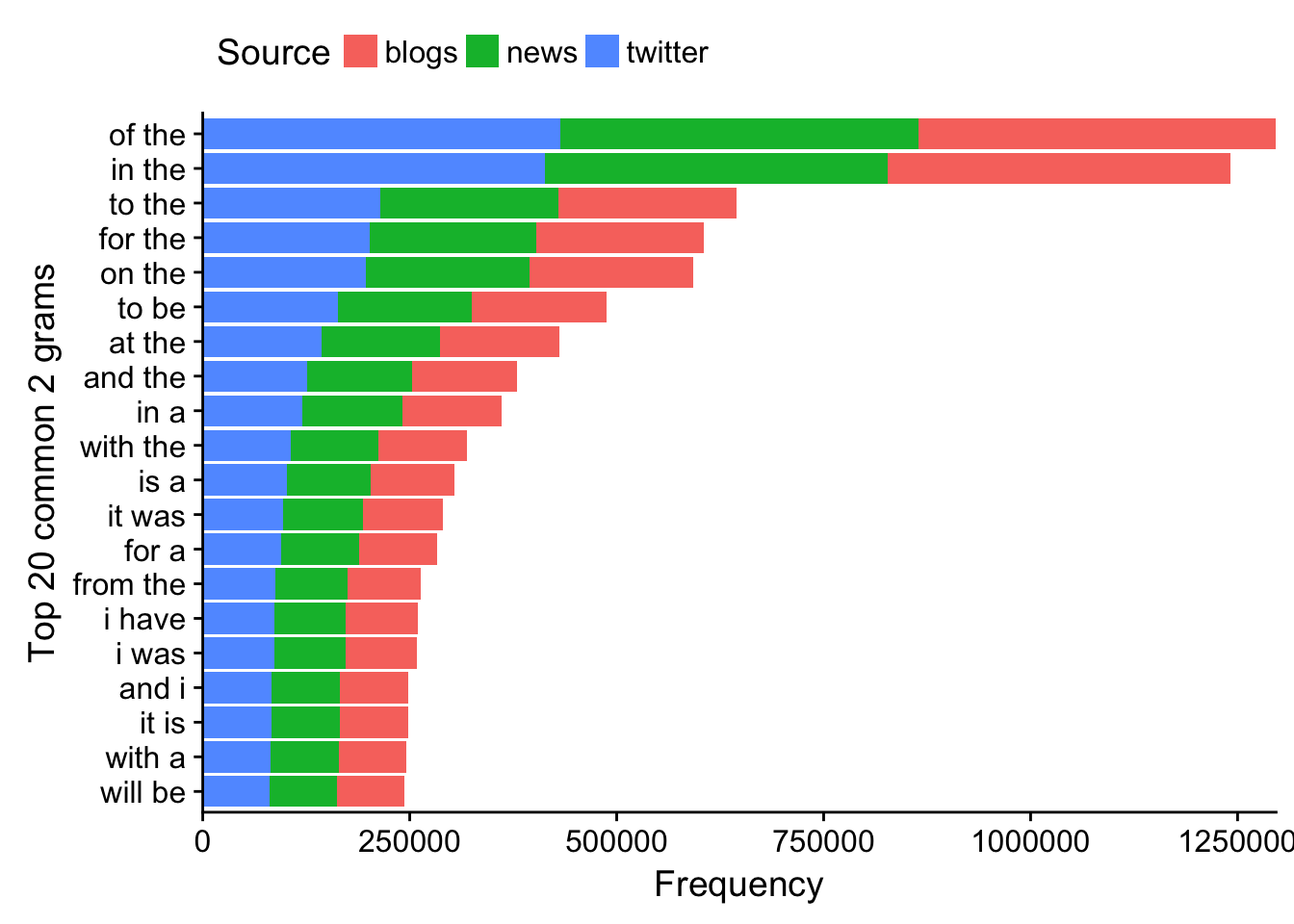
coord\_flip() +

ylab('Frequency') + xlab('Top 20 common 2 grams') +

scale\_y\_continuous(expand = c(0, 0)) +

guides(fill = guide\_legend(title = 'Source')) +

theme(legend.position = 'top')



3 grams

* Identify 3 grams and make a dfm object

*#Find 3 grams*

tokWord3g <- tokens\_ngrams(tokWord, n = 3L, concatenator = ' ')

dfmWord3g <- dfm(tokWord3g, tolower = T)

* Calculate top 20 most frquent 3 grams and extract frequencies from individual source

*#Identify top 20 most common 3 grams*

top3gAll <- topfeatures(dfmWord3g, n = 20)[1:20]

top3gBlog <- dfmWord3g[1, names(top3gAll)]

top3gNews <- dfmWord3g[2, names(top3gAll)]

top3gTwt <- dfmWord3g[3, names(top3gAll)]

* Make a data frame for plotting

gram3dist <- as.data.frame(top3gAll)

gram3dist <- cbind(gram3dist,

t(as.data.frame(top3gBlog)[, -1]),

t(as.data.frame(top3gNews)[, -1]),

t(as.data.frame(top3gTwt)[, -1]))

colnames(gram3dist) <- c('all', 'blogs', 'news', 'twitter')

gram3dist$words <- row.names(gram3dist)

* Plot 3 gram distribution

df <- melt(gram3dist, id.vars = c('words', 'all')) *#convert to long format*

ggplot(data = df, aes(x = reorder(words, all), y = all)) +

geom\_col(aes(fill = variable)) +

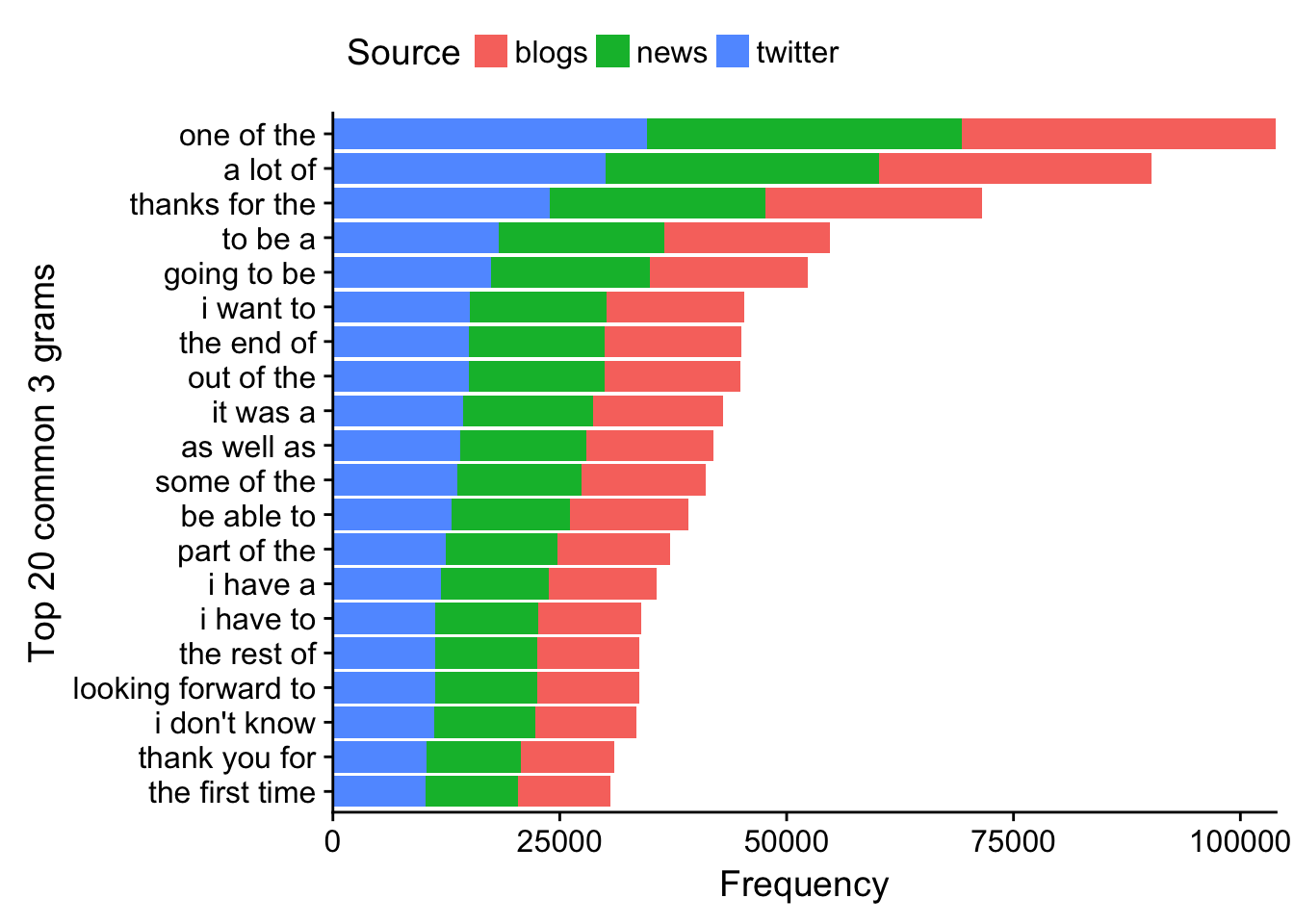
ylab('Frequency') + xlab('Top 20 common 3 grams') +

coord\_flip() +

scale\_y\_continuous(expand = c(0, 0)) +

guides(fill = guide\_legend(title = 'Source')) +

theme(legend.position = 'top')



Summary of n grams

* Construct a table with the % sparsity of 2 and 3 grams across each source

*#sparcity can be accessed by dfmWord2g for all sources and dfmWord2g[1, ] for individual source*

nGramSpars <- data.frame('source' = c('all', 'blog', 'news', 'twitter'),

'g2spars' = c(58.3, 57, 57, 60.8),

'g3spars' = c(62.9, 60.2, 61.9, 66.6))

nGramSpars

## source g2spars g3spars

## 1 all 58.3 62.9

## 2 blog 57.0 60.2

## 3 news 57.0 61.9

## 4 twitter 60.8 66.6

Word coverage

* Write a function that calculates the amount of unique words to reach a specified coverage

wordCoverage <- **function**(inputDfm, coverage){

*## This function takes in a dfm object and a target coverage*

*## and returns the number of words required to reach that coverage*

*## and the actual coverage reached*

freq <- dfm\_weight(inputDfm, scheme = 'prop') \* 100

*#calculate percentage frequency for each word*

totalWords <- nfeat(freq)

freq <- topfeatures(freq, n = totalWords)

coverageCount <- 0

wordN <- 0

**for** (i **in** 1:totalWords) {

**if** (coverageCount <= coverage) {

coverageCount <- coverageCount + freq[i]

wordN <- i

}

}

**return**(c(wordN, coverageCount))

}

* The number of words required to reach 50% or 90% coverage if all words are included

wordCoverage(dfmWord, 50) *#50%*

## the

## 7.00000 52.18241

wordCoverage(dfmWord, 90) *#90%*

## the

## 26.00000 90.35665

* The number of words required to reach 50% or 90% coverage if stop words are removed

wordCoverage(dfmWordTrim, 50) *#50%*

## one

## 57.00000 50.03899

wordCoverage(dfmWordTrim, 90) *#90%*

## one

## 183.00000 90.14836

Conclusions and discussions

1. Articles and prepositions are some of the most frequenctly
2. If the stop words are removed first, verbs such as say, like, go, get are among the most frequently used words
3. Some words in different tenses are double/triple-counted, as in the case of say/said. This could be something worth exploring to figure out how to view words in different tenses as the same
4. Similarly, it could be interesting to explore homonyms and separate them by contexts
5. As the n for n grams increase, the computation time and size dramatically increases  
   1 gram 50.1MB; 2 gram 1.2 GB; 3 gram 4.4 GB  
   It would be important to think about parallelization in R to reduce run time
6. Considering all words, the most frequently used 26 words constitutes 90% of usage; if stop words are removed, still only 183 words are required!

Future plans

1. Explore parallelization in R
2. Build basic n-gram model with Markov Chains
3. Inluce 4 grams if my computer can handle it
4. Calculate % of all 2 and 3 grams used. When predciting, return the most frequent regardless of n grams