Capstone Week 2

Phil Renner

2/19/2022

Overview

This document provides the Milestone Report for week 2 of the Coursera Data Science Capstone project.

The objective of this report is to develop an understanding of the various statistical properties of the data set that can later be used when building the prediction model for the final data product - the Shiny application. Using exploratory data analysis, this report describes the major features of the training data and then summarizes my plans for creating the predictive model.

The model will be trained using a unified document corpus compiled from the following three sources of text data:

- Blogs
- News
- Twitter

The provided text data are provided in four different languages. This project will only focus on the English corpora.

Load the Data

The first step is to download, unzip, and open the data

```
library(knitr)
rm(list = ls(all.names = TRUE))



trainURL <- "https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip"
trainDataFile <- "data/Coursera-SwiftKey.zip"

if (!file.exists('data')) {
    dir.create('data')
}

if (!file.exists("data/final/en_US")) {
    tempFile <- tempfile()
    download.file(trainURL, tempFile)
    unzip(tempFile, exdir = "data")
    unlink(tempFile)
}
```

```
# blogs
blogsFileName <- "data/final/en_US/en_US.blogs.txt"
con <- file(blogsFileName, open = "r")
blogs <- readLines(con, encoding = "UTF-8", skipNul = TRUE)
close(con)

# news
newsFileName <- "data/final/en_US/en_US.news.txt"
con <- file(newsFileName, open = "r")
news <- readLines(con, encoding = "UTF-8", skipNul = TRUE)
close(con)

# twitter
twitterFileName <- "data/final/en_US/en_US.twitter.txt"
con <- file(twitterFileName, open = "r")
twitter <- readLines(con, encoding = "UTF-8", skipNul = TRUE)
close(con)

rm(con)
```

Summary of the Data

The next step is to understand what is in the data, including file sizes, number of lines, characters, and words in each file. It also includes the distribution of words per line.

```
library(stringi)
library(kableExtra)

# assign sample size
sampleSize = 0.003

# file size
fileSizeMB <- round(file.info(c(blogsFileName,
                                newsFileName,
                                twitterFileName))$size / 1024 ^ 2)

# num lines per file
numLines <- sapply(list(blogs, news, twitter), length)

# num characters per file
numChars <- sapply(list(nchar(blogs), nchar(news), nchar(twitter)), sum)

# num words per file
numWords <- sapply(list(blogs, news, twitter), stri_stats_latex)[4,]

# words per line
wpl <- lapply(list(blogs, news, twitter), function(x) stri_count_words(x))

# words per line summary
wplSummary = sapply(list(blogs, news, twitter),
             function(x) summary(stri_count_words(x))[c('Min.', 'Mean', 'Max.')])
rownames(wplSummary) = c('WPL.Min', 'WPL.Mean', 'WPL.Max')
```

Table 1:

File	FileSize	Lines	Characters	Words	WPL.Min	WPL.Mean	WPL.Max
en_US.blogs.txt	200 MB	899288	206824505	37570839	0	42	6726
en_US.news.txt	196 MB	77259	15639408	2651432	1	35	1123
en_US.twitter.txt	159 MB	2360148	162096241	30451170	1	13	47

```
summary <- data.frame(
    File = c("en_US.blogs.txt", "en_US.news.txt", "en_US.twitter.txt"),
    FileSize = paste(fileSizeMB, " MB"),
    Lines = numLines,
    Characters = numChars,
    Words = numWords,
    t(rbind(round(wplSummary)))
)

kable(summary,
      row.names = FALSE,
      align = c("l", rep("r", 7)),
      caption = "") %>% kable_styling(position = "left")
```

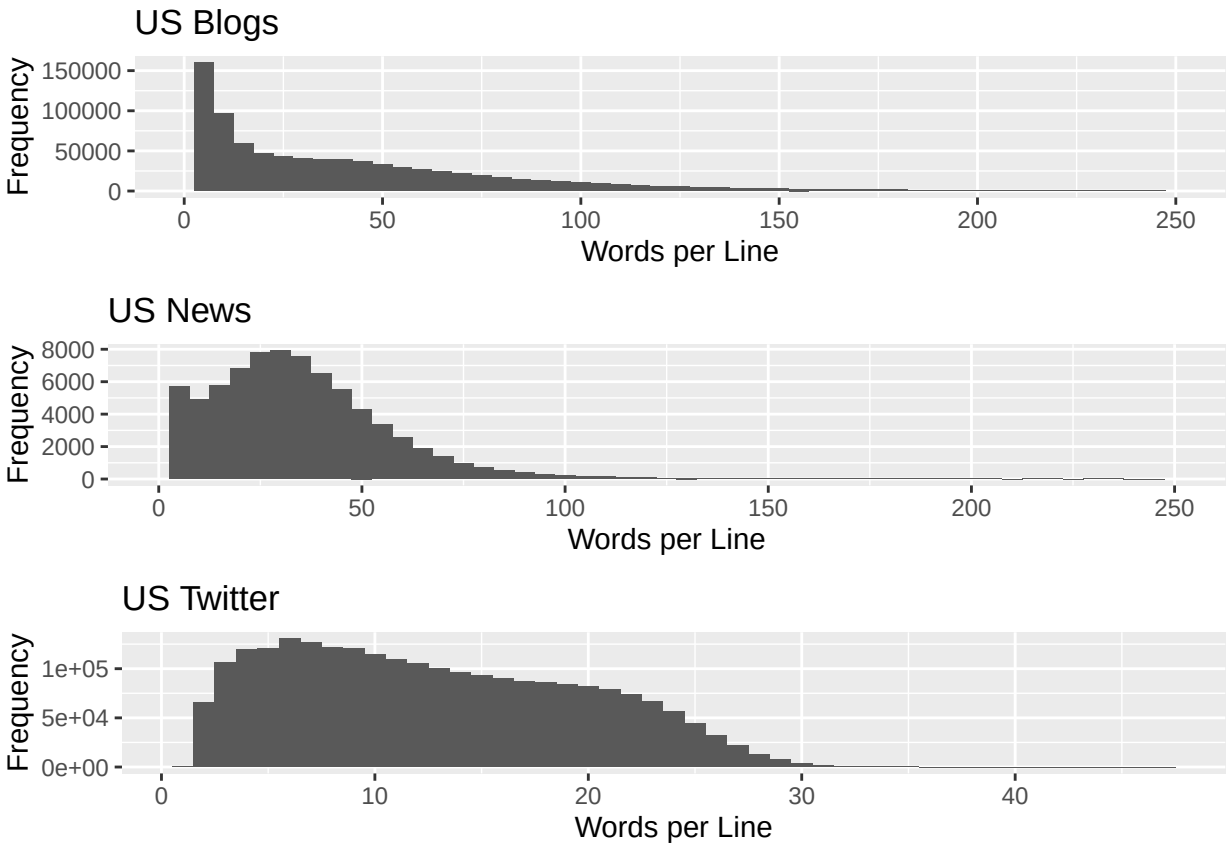
This initial summary of the data shows that each line in the corpora is fairly short, ranging from 13 words for the Twitter corpus to 42 for the blog corpus. However, these files are still very large, and to manage the large files we will need to select a 0.5% sample of each file for analysis and to create the predictive app.

Histograms of the Data

We will produce histograms of the line lengths for each of the three types of files.

```
# create histograms of words per line
library(ggplot2)
library(gridExtra)

plot1 <- qplot(wpl[[1]],
               geom = "histogram",
               main = "US Blogs",
               xlab = "Words per Line",
               ylab = "Frequency",
               binwidth = 5,
               xlim = c(0,250))

plot2 <- qplot(wpl[[2]],
               geom = "histogram",
               main = "US News",
               xlab = "Words per Line",
               ylab = "Frequency",
               binwidth = 5,
               xlim = c(0,250))

plot3 <- qplot(wpl[[3]],
               geom = "histogram",
```

```
                main = "US Twitter",
                xlab = "Words per Line",
                ylab = "Frequency",
                binwidth = 1)

plotList = list(plot1, plot2, plot3)
do.call(grid.arrange, c(plotList, list(ncol = 1)))
```



The histograms show the relatively short line length of each of the three types of texts in the analysis. There is some variation in the distribution, but each distribution is heavily skewed.

Prepare the Data and Build the Corpus

Prior to doing the exploratory analysis, we will need to assemble the corpus. We will sample the text files at 0.5% to improve efficiency and to work within the constraints of our computing power. The program will then perform the following steps to clean up the data:

- Remove URL, Twitter handles and email patterns by converting them to spaces using a custom content transformer
- Convert all words to lowercase
- Remove common English stop words
- Remove punctuation marks
- Remove numbers
- Trim whitespace
- Remove profanity

Table 2: First 10 Documents

spluttered cant can give raise double salary
overall best book series absolutely stellar liked first books series love one tons action building tension shifters vamps lots et
index uses split residential commercial property implied results tranche nama loans also assumed uk ireland mix residentia
im currently living davis house surrounded nature city bike ride merely three minutes best part around place bunch orang
fb came helped finish phrases realized one made sense none actually make sense
technique watercoloring
particular project adds nice addition treats plan make family friends coworkers instead placing cookies etc plastic paper pl
bags will making long beach ronald mcdonald house put chocolates hand cream lip gloss little fancy favor boxes hybrid sta
leading hollywood directors washington political figures leading way america quickly returning paganism shrouded world s
commando

- Convert to plain text documents

```
# set seed for reproducability
set.seed(88)

# sample all three data sets
sampleBlogs <- sample(blogs, length(blogs) * sampleSize, replace = FALSE)
sampleNews <- sample(news, length(news) * sampleSize, replace = FALSE)
sampleTwitter <- sample(twitter, length(twitter) * sampleSize, replace = FALSE)

# remove all non-English characters from the sampled data
sampleBlogs <- iconv(sampleBlogs, "latin1", "ASCII", sub = "")
sampleNews <- iconv(sampleNews, "latin1", "ASCII", sub = "")
sampleTwitter <- iconv(sampleTwitter, "latin1", "ASCII", sub = "")

# combine all three data sets into a single data set and write to disk
sampleData <- c(sampleBlogs, sampleNews, sampleTwitter)
sampleDataFileName <- "data/final/en_US/en_US.sample.txt"
con <- file(sampleDataFileName, open = "w")
writeLines(sampleData, con)
close(con)

# get number of lines and words from the sample data set
sampleDataLines <- length(sampleData);
sampleDataWords <- sum(stri_count_words(sampleData))

# remove variables no longer needed to free up memory
rm(blogs, news, twitter, sampleBlogs, sampleNews, sampleTwitter)
```

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

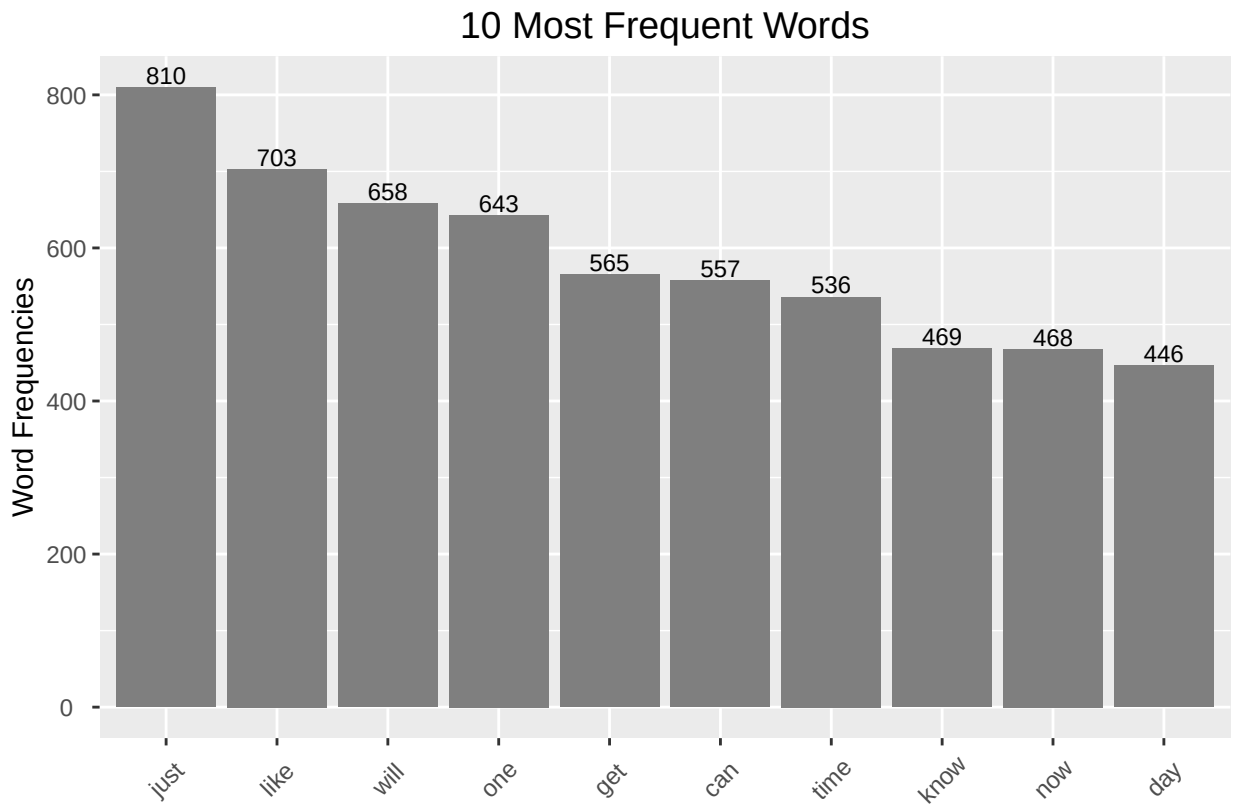
Exploratory Analysis

Now that we have prepared the text files and created a sample to enable analysis, we can do the exploratory analysis. The analysis will help us understand which words are most frequent in the corpus, through a bar chart and a word cloud.

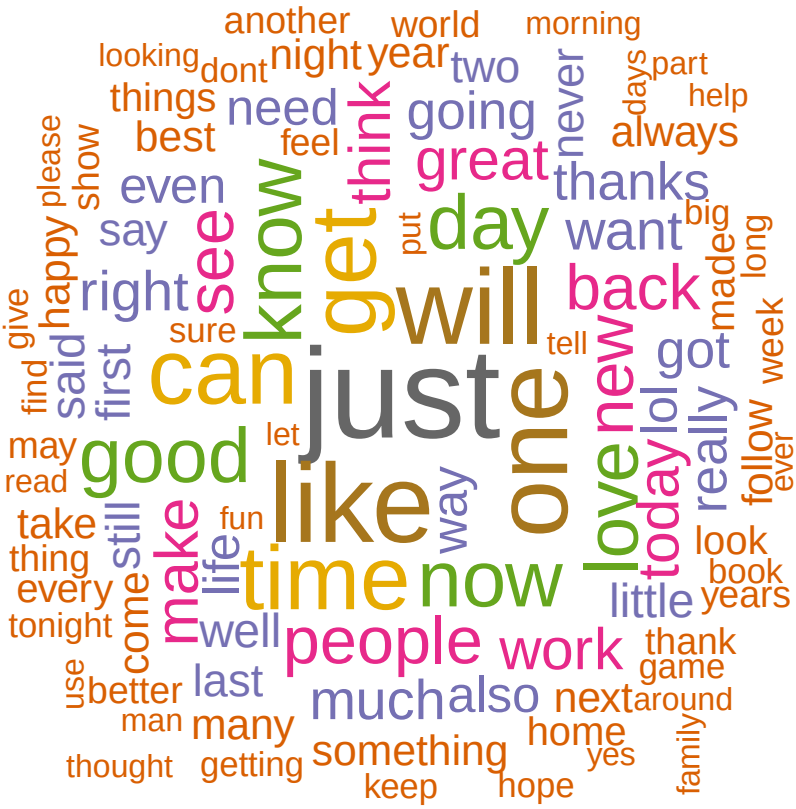
```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(RColorBrewer)

tdm <- TermDocumentMatrix(corpus)
freq <- sort(rowSums(as.matrix(tdm)), decreasing = TRUE)
wordFreq <- data.frame(word = names(freq), freq = freq)

# plot the top 10 most frequent words
g <- ggplot (wordFreq[1:10,], aes(x = reorder(wordFreq[1:10,]$word, -wordFreq[1:10,]$fre),
                                  y = wordFreq[1:10,]$fre ))
g <- g + geom_bar( stat = "Identity" , fill = I("grey50"))
g <- g + geom_text(aes(label = wordFreq[1:10,]$fre), vjust = -0.20, size = 3)
g <- g + xlab("")
g <- g + ylab("Word Frequencies")
g <- g + theme(plot.title = element_text(size = 14, hjust = 0.5, vjust = 0.5),
               axis.text.x = element_text(hjust = 0.5, vjust = 0.5, angle = 45),
               axis.text.y = element_text(hjust = 0.5, vjust = 0.5))
g <- g + ggtitle("10 Most Frequent Words")
print(g)
```




```
# construct word cloud
suppressWarnings (
    wordcloud(words = wordFreq$word,
              freq = wordFreq$freq,
              min.freq = 1,
              max.words = 100,
              random.order = FALSE,
              rot.per = 0.35,
              colors=brewer.pal(8, "Dark2"))
)
```



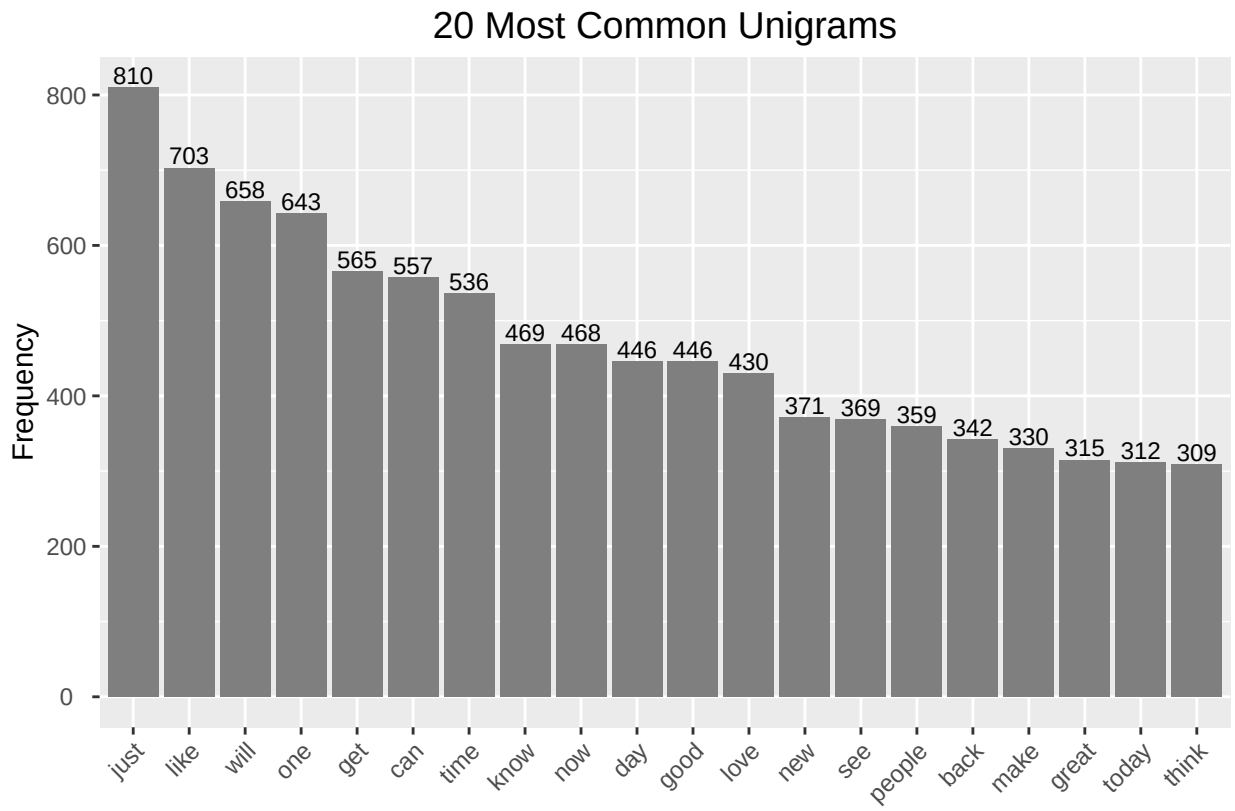

```
# remove variables no longer needed to free up memory
rm(tdm, freq, wordFreq, g)
```

N-Grams

The predictive model I will build for the application will handle uniqrams, bigrams, and trigrams. In this section, I have used the RWeka package to construct functions that tokenize the sample data and construct matrices of uniqrams, bigrams, and trigrams.

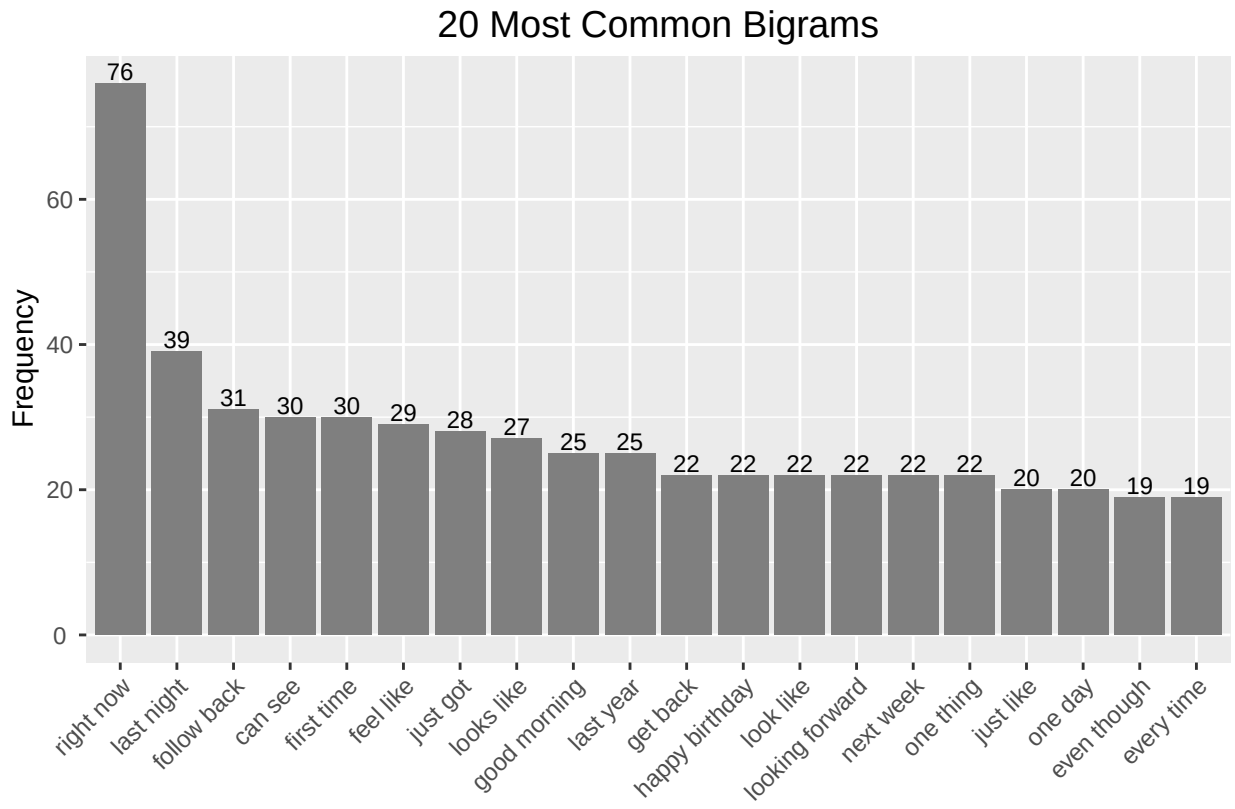
```
#tokenize for n-grams

library(RWeka)

unigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 1, max = 1))
bigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
trigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 3, max = 3))


# create term document matrix for the corpus
unigramMatrix <- TermDocumentMatrix(corpus, control = list(tokenize = unigramTokenizer))

# eliminate sparse terms for each n-gram and get frequencies of most common n-grams
unigramMatrixFreq <- sort(rowSums(as.matrix(removeSparseTerms(unigramMatrix, 0.99))), decreasing = TRUE)
unigramMatrixFreq <- data.frame(word = names(unigramMatrixFreq), freq = unigramMatrixFreq)
```

```r
# generate plot
g <- ggplot(unigramMatrixFreq[1:20,], aes(x = reorder(word, -freq), y = freq))
g <- g + geom_bar(stat = "identity", fill = I("grey50"))
g <- g + geom_text(aes(label = freq ), vjust = -0.20, size = 3)
g <- g + xlab("")
g <- g + ylab("Frequency")
g <- g + theme(plot.title = element_text(size = 14, hjust = 0.5, vjust = 0.5),
               axis.text.x = element_text(hjust = 1.0, angle = 45),
               axis.text.y = element_text(hjust = 0.5, vjust = 0.5))
g <- g + ggtitle("20 Most Common Unigrams")
print(g)
```



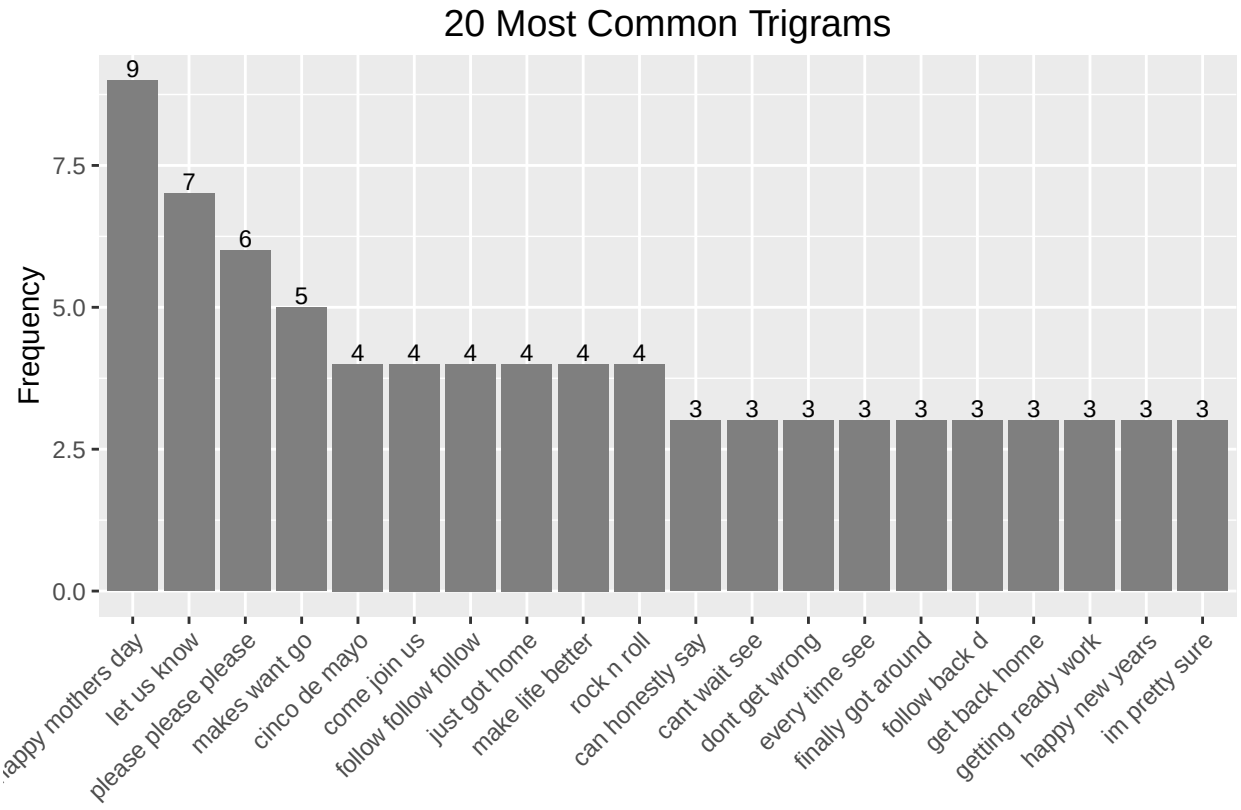

```r
#Bigrams

# create term document matrix for the corpus
bigramMatrix <- TermDocumentMatrix(corpus, control = list(tokenize = bigramTokenizer))

# eliminate sparse terms for each n-gram and get frequencies of most common n-grams
bigramMatrixFreq <- sort(rowSums(as.matrix(removeSparseTerms(bigramMatrix, 0.999))), decreasing = TRUE)
bigramMatrixFreq <- data.frame(word = names(bigramMatrixFreq), freq = bigramMatrixFreq)

# generate plot
g <- ggplot(bigramMatrixFreq[1:20,], aes(x = reorder(word, -freq), y = freq))
g <- g + geom_bar(stat = "identity", fill = I("grey50"))
g <- g + geom_text(aes(label = freq ), vjust = -0.20, size = 3)
```

```
g <- g + xlab("")
g <- g + ylab("Frequency")
g <- g + theme(plot.title = element_text(size = 14, hjust = 0.5, vjust = 0.5),
               axis.text.x = element_text(hjust = 1.0, angle = 45),
               axis.text.y = element_text(hjust = 0.5, vjust = 0.5))
g <- g + ggtitle("20 Most Common Bigrams")
print(g)
```




```
#Trigrams

# create term document matrix for the corpus
trigramMatrix <- TermDocumentMatrix(corpus, control = list(tokenize = trigramTokenizer))

# eliminate sparse terms for each n-gram and get frequencies of most common n-grams
trigramMatrixFreq <- sort(rowSums(as.matrix(removeSparseTerms(trigramMatrix, 0.9999))), decreasing = TRU
trigramMatrixFreq <- data.frame(word = names(trigramMatrixFreq), freq = trigramMatrixFreq)

# generate plot
g <- ggplot(trigramMatrixFreq[1:20,], aes(x = reorder(word, -freq), y = freq))
g <- g + geom_bar(stat = "identity", fill = I("grey50"))
g <- g + geom_text(aes(label = freq ), vjust = -0.20, size = 3)
g <- g + xlab("")
g <- g + ylab("Frequency")
g <- g + theme(plot.title = element_text(size = 14, hjust = 0.5, vjust = 0.5),
               axis.text.x = element_text(hjust = 1.0, angle = 45),
```

```
                axis.text.y = element_text(hjust = 0.5, vjust = 0.5))
g <- g + ggtitle("20 Most Common Trigrams")
print(g)
```



Building the Application

The final deliverable in the capstone project is to build a predictive algorithm that will be deployed as a Shiny app for the user interface. The Shiny app should take as input a phrase (multiple words) in a text box input and output a prediction of the next word.

The predictive algorithm will be developed using an n-gram model with a word frequency lookup similar to that performed in the exploratory data analysis section of this report. A strategy will be built based on the knowledge gathered during the exploratory analysis. For example, as n increased for each n-gram, the frequency decreased for each of its terms. So one possible strategy may be to construct the model to first look for the unigram that would follow from the entered text. Once a full term is entered followed by a space, find the most common bigram model and so on.

Another possible strategy may be to predict the next word using the trigram model. If no matching trigram can be found, then the algorithm would check the bigram model. If still not found, use the unigram model.

The final strategy will be based on the one that increases efficiency and provides the best accuracy.