Natural Language Processing Project

Dinh Tuan Phan

5/2/2021

Synopsis

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.

Course Dataset

The training data set listed below will be used to get started and will be the basis for most of the capstone.

- Capstone Dataset

Environment setup

Load the necessary libraries for the analysis. Install these packages before loading.

```r
library(stringi)
library(kableExtra)
library(ggplot2)
library(gridExtra)
library(tm)
library(wordcloud)
library(RColorBrewer)
library(RWeka)
set.seed(007)
# Because the memory limitation, I might only need 0.5% of total available words for training data
sampleSize = 0.005
```

Load data

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

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

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

Data summary

We run basic explanatory analysis about three corpora.

1. File size

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

2. Number of lines per file

```r
numLines <- sapply(list(blogs, news, twitter), length)
```

3. Number of characters per file

```r
numChars <- sapply(list(nchar(blogs), nchar(news), nchar(twitter)), sum)
```

4. Number of words per file

```r
numWords <- sapply(list(blogs, news, twitter), stri_stats_latex)[4,]
```

5. Numbers of words per line

```r
wpl <- lapply(list(blogs, news, twitter), function(x) stri_count_words(x))
```

6. Number words per line summary

```r
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')

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")
```

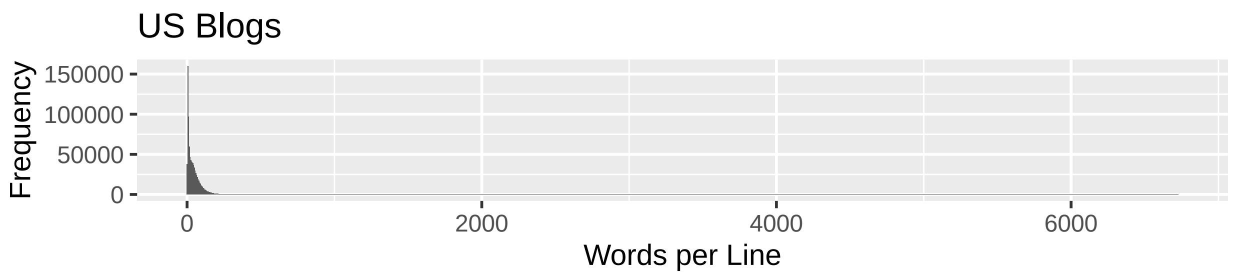
Histogram of Words per Line

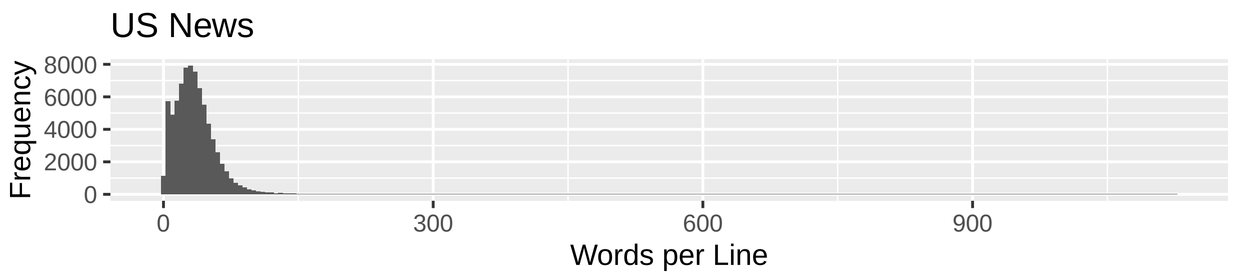
Histogram of words per line for the three text corpora.

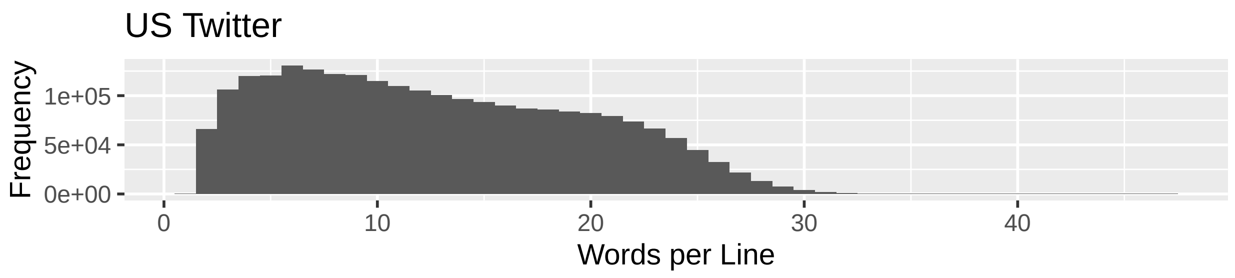
Table 1:

File	FileSize	Lines	Characters	Words	WPL.Min	WPL.Mean	WPL.Max
en_US.blogs.txt	200.424207687378 MB	899288	206824505	37570839	0	42	6726
en_US.news.txt	196.277512550354 MB	77259	15639408	2651432	1	35	1123
en_US.twitter.txt	159.364068984985 MB	2360148	162096241	30451170	1	13	47

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

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

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)))
```







```r
# free up some memory
rm(plot1, plot2, plot3)
```

Sample and Clean the Data

```r
# 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 <- "./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)
```

Build Corpus

```
badWordsFile <- "badWords.txt"

buildCorpus <- function (dataSet) {
    docs <- VCorpus(VectorSource(dataSet))
    toSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))

    # remove URL, Twitter handles and email patterns
    docs <- tm_map(docs, toSpace, "(f|ht)tp(s?)://(.*)[.][a-z]+")
    docs <- tm_map(docs, toSpace, "@[^\\s]+")
    docs <- tm_map(docs, toSpace, "\\b[A-Z a-z 0-9._ - ]*[@](.*?)[.]{1,3} \\b")

    # remove profane words from the sample data set
    con <- file(badWordsFile, open = "r")
    profanity <- readLines(con, encoding = "UTF-8", skipNul = TRUE)
    close(con)
    profanity <- iconv(profanity, "latin1", "ASCII", sub = "")
    docs <- tm_map(docs, removeWords, profanity)

    docs <- tm_map(docs, tolower)
    docs <- tm_map(docs, removeWords, stopwords("english"))
    docs <- tm_map(docs, removePunctuation)
    docs <- tm_map(docs, removeNumbers)
    docs <- tm_map(docs, stripWhitespace)
    docs <- tm_map(docs, PlainTextDocument)
    return(docs)
}

# build the corpus and write to disk (RDS)
corpus <- buildCorpus(sampleData)
saveRDS(corpus, file = "./final/en_US/en_US.corpus.rds")

# convert corpus to a dataframe and write lines/words to disk (text)
corpusText <- data.frame(text = unlist(sapply(corpus, '[', "content")), stringsAsFactors = FALSE)
con <- file("./final/en_US/en_US.corpus.txt", open = "w")
writeLines(corpusText$text, con)
close(con)

kable(head(corpusText$text, 10),
      row.names = FALSE,
      col.names = NULL,
      align = c("l"),
      caption = "First 10 Documents") %>% kable_styling(position = "left")
```

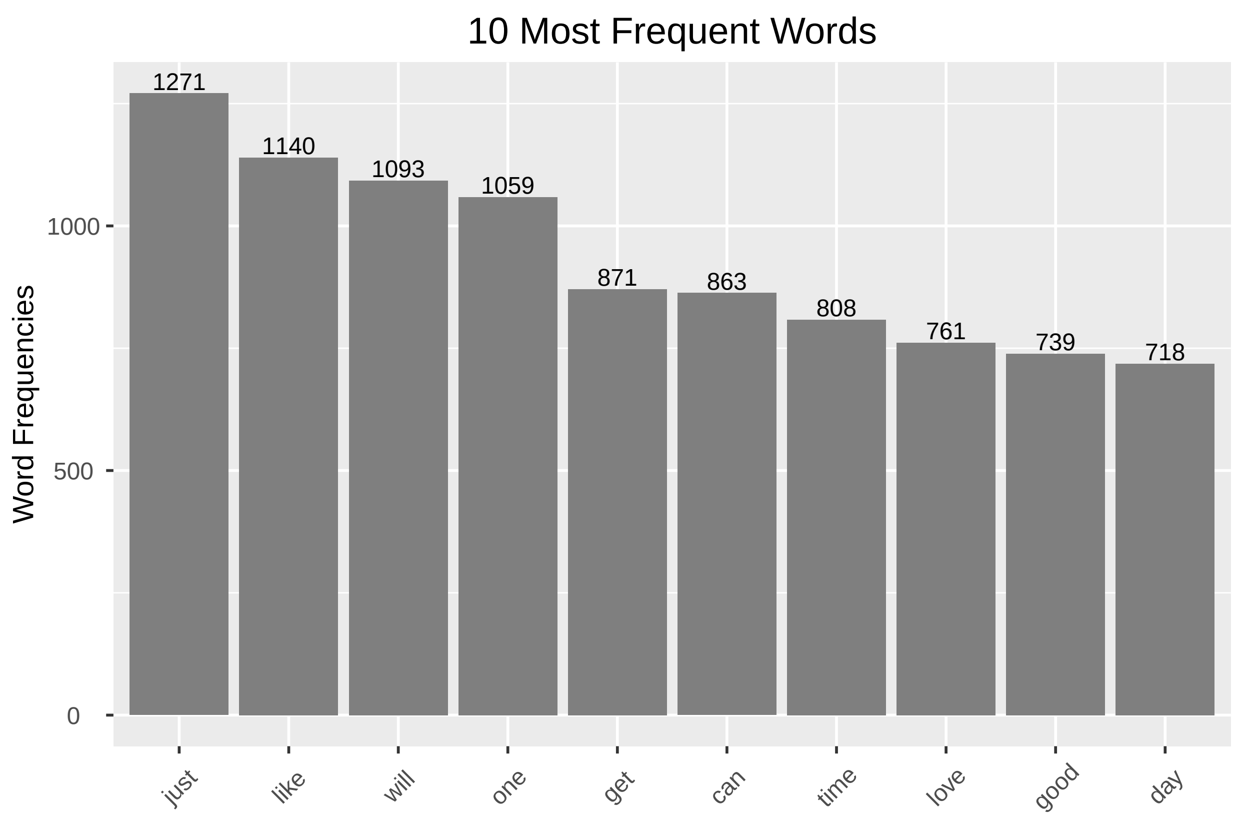
```
# remove variables no longer needed to free up memory
rm(sampleData)
```

Table 2: First 10 Documents

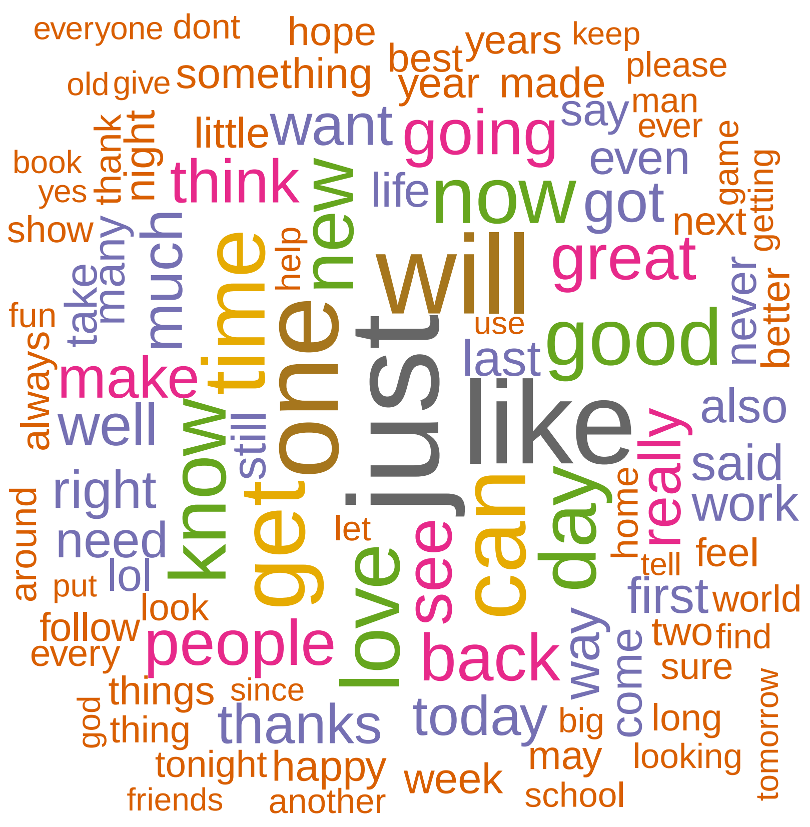
background
teenager left chased savagely beaten gang asians suspected
editorial continues
good morning everyone name anton im student mipt today presentation im going speak software design patterns presentat
holding phone away head couple inches helps better yet put cell speaker phone
mobile malware potential even powerful traditional pcbased malware inherent traits smartphones tablets almost include ca
afghanistan asadabad child murdered taliban rocket
s facade appear historical site however fact thomass hospital area eight hundred years early accounts ad name hospital loca
april fabulous fun month little babushkas big hit lots new changes may returning designer month will hazel parr will joinin
taken whimsical designs made socks

Word Frequencies

```r
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)
```

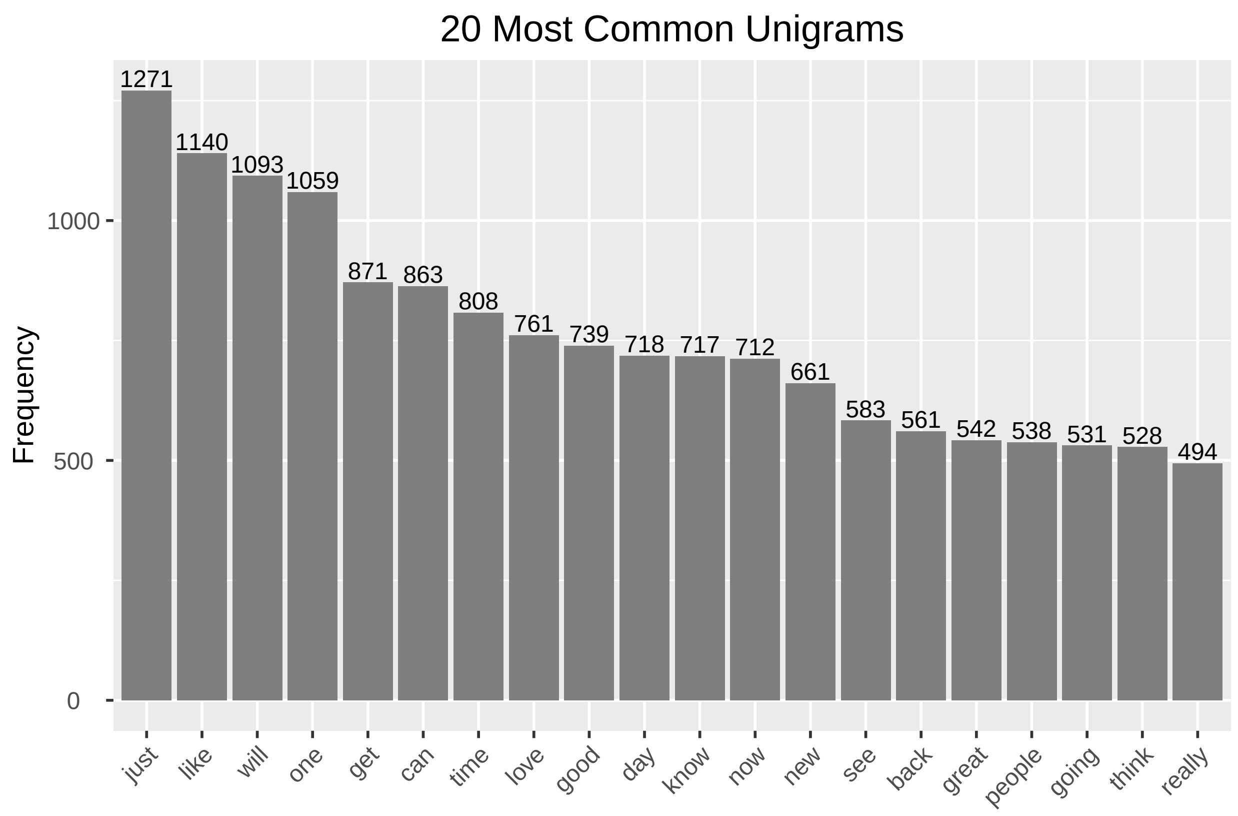
Tokenizing and N-Gram Generation

Tokenize Functions

```
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))
```

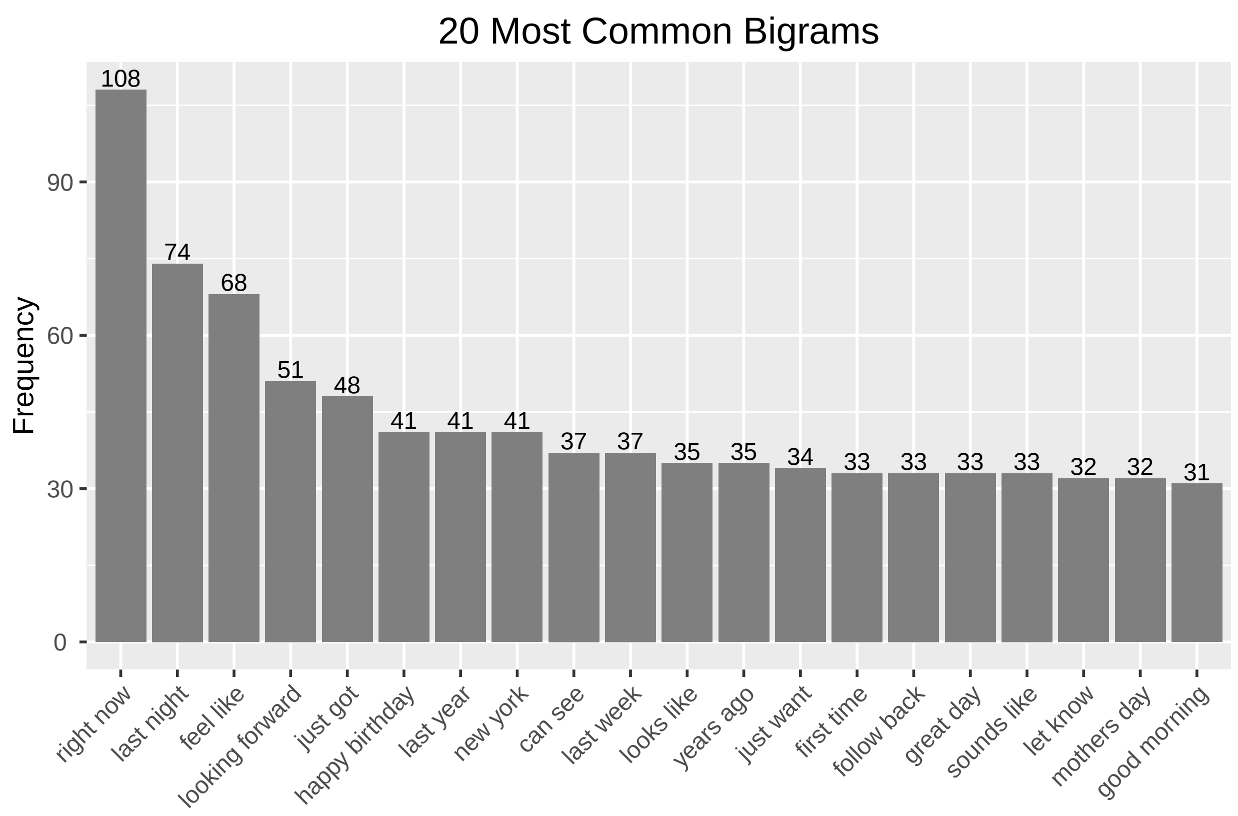
Unigrams

```
# 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)

# 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)
```



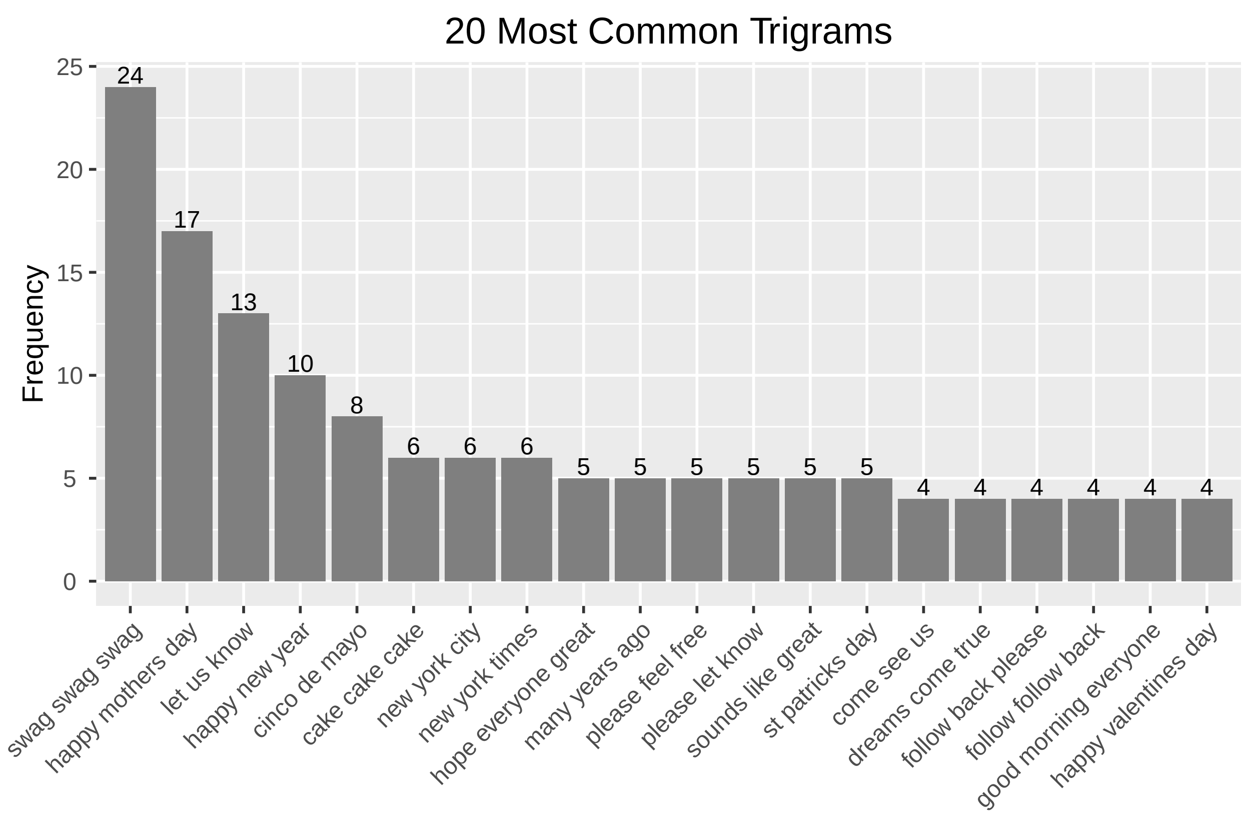
Bigrams

```
# 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)
```


20 Most Common Trigrams
Frequency
0
5
10
15
20
25
24
17
13
10
8
6
6
6
5
5
5
5
5
5
4
4
4
4
4
4
swag swag swag
happy mothers day
let us know
happy new year
cinco de mayo
cake cake cake
new york city
new york times
hope everyone great
many years ago
please feel free
please let know
sounds like great
st patricks day
come see us
dreams come true
follow back please
follow follow back
good morning everyone
happy valentines day