Coursera Capstone Project Milestone Report

Niloufar Yousefi

Introduction

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

The goal of this report is to perform exploratory analysis to understand statistical properties of the data set that can later be used when building the prediction model for the final Shiny application. Here we will identify the major features of the training data and then summarize plans for the predictive model.

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

1. Blogs
2. News
3. Twitter

This project will only focus on the English corpora (4 languages available)

Setup

Load initial packages and clear the global workspace.

```
library(knitr)
rm(list = ls(all.names = TRUE))
setwd("~/Coursera/Data Science Capstone/Project")
```

Load the Data

Download, unzip and load the training data.

The files are downloaded and stored locally: * Blog: en_US.blogs.txt * News: en_US.news.txt * Twitter: en_US.twitter.txt

```
trainURL <- "https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Course
ra-SwiftKey.zip"
trainDataFile <- "data/Coursera-SwiftKey.zip"
if (!file.exists('data')) {
```

```r
    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)
```

```
## Warning in readLines(con, encoding = "UTF-8", skipNul = TRUE): incomplete
final
## line found on 'data/final/en_US/en_US.news.txt'
```

```r
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)
```

Data Summary

We look at the summary of each of the three text corpora before building the unified document corpus and cleaning the data. It includes file sizes, number of lines, number of characters, and number of words for each source file. Also included are basic statistics on the number of words per line (min, mean, and max).

Initial Data Summary

File	FileSize	Lines	Characters	Words	WPL.Min	W
en_US.blogs.txt	200 MB	899288	206824505	37570839	0	
en_US.news.txt	196 MB	77259	15639408	2651432	1	
en_US.twitter.txt	159 MB	2360148	162096241	30451170	1	

The source code for the above table is attached as A.1 Basic Data Summary in the Appendix section. An initial investigation of the data shows that on average, each text corpora has a relatively low number of words per line. Blogs tend to have more words per line, followed by news and then twitter which has the least words per line. The lower number of words per line for the Twitter data is expected given that a tweet is limited to a certain number of characters. Even when Twitter doubled its character count from 140 to 280 characters in 2017, research shows that only 1% of tweets hit the 280-character limit, and only 12% of tweets are longer than 140 characters. Perhaps after so many years, users were simply trained to the 140-character limit.
Another important observation in this initial investigation shows that the text files are fairly large. To improve processing time, a smaller sample size of 0% will be obtained from all three data sets and then combined into a unified document corpus for subsequent analyses later in this report as part of preparing the data.

Histogram of Words per Line

The relatively low number of words in the three source files charted earlier in this section is also visible in the histogram plots shown above. This observation seems to support a general trend towards short and concise communications that may be useful later in the project.

The source code for the above plot is attached as A.2 Histogram of Words per Line in the Appendix section.

Prepare the Data

Prior to performing exploratory data analysis, the three data sets will be sampled at 0% to improve performance. All non-English characters will be removed from the subset of data and then combined into a single data set. The combined sample data set will be written to disk which contains 6,672 lines and 141,607 words.

The next step is to create a corpus from the sampled data set. A custom function named `buildCorpus` will be employed to perform the following transformation steps for each document:

1. Remove URL, Twitter handles and email patterns by converting them to spaces using a custom content transformer
2. Convert all words to lowercase
3. Remove common English stop words
4. Remove punctuation marks
5. Remove numbers
6. Trim whitespace
7. Remove profanity
8. Convert to plain text documents

The corpus will then be written to disk in two formats: a serialized R object in RDS format and as a text file. Finally, the first 10 documents (lines) from the corpus will be displayed.

First 10 Documents
began sophmore year high school school year began simple enough class worried earth science taken lower leve something enjoyed walked classroom noticed two things friends class collection dumbest people grown through school response nope promptly class ended went counselors office change schedule end got switched chemistry never worse teacher life kind teacher makes cringe anything remotely related subject comes outside school mad feeling grown love
declutters makes piles unpile reclutter every week cant believe three words arent words declutter unpile reclutte give whatfor pound fist knock change
also fact write one day will never thing will write next day perceptions meaning life whatever constantly chang fanatic journal keeper
took yr blended circles side brush tip just first petal amount time effort now can see big different two petals firs smooth blend

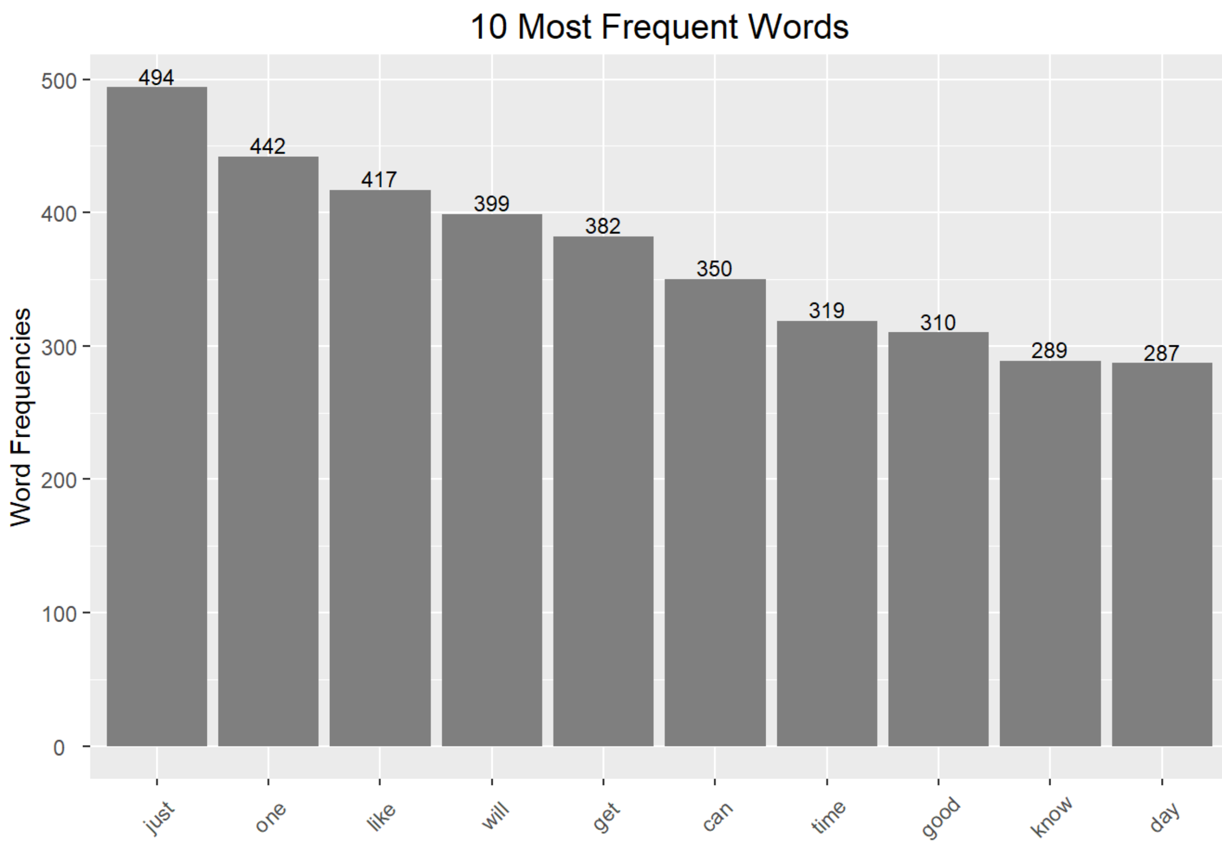
see juan saw young male rob meijers hiphop style clothing sneakers nikes
crime delinquency council issues annual media awards
familiar exact constitution tarot pack
bashing depression
posting photos taken last week installation process landscape futures instruments devices architectural inventio exhibition opens saturday august will february
april newhouse news service reported richard nixon commissioned rand study feasibility canceling election ranc work possible sources story said review fruitless

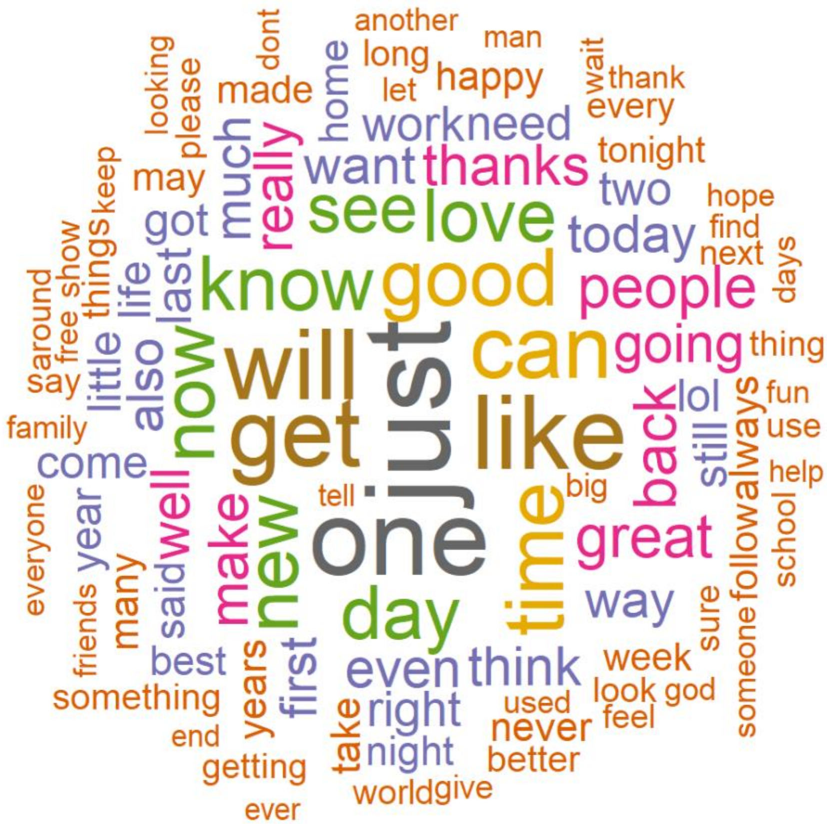
The source code for preparing the data is attached as A.3 Sample and Clean the Data and A.4 Build Corpus in the Appendix section. ## Exploratory Data Analysis

Exploratory data analysis will be performed to fulfill the primary goal for this report. Several techniques will be employed to develop an understanding of the training data which include looking at the most frequently used words, tokenizing and n-gram generation.

Word Frequencies

A bar chart and word cloud will be constructed to illustrate unique word frequencies.


10 Most Frequent Words
Word Frequencies
500
400
300
200
100
0
494
442
417
399
382
350
319
310
289
287
just
one
like
will
get
can
time
good
know
day



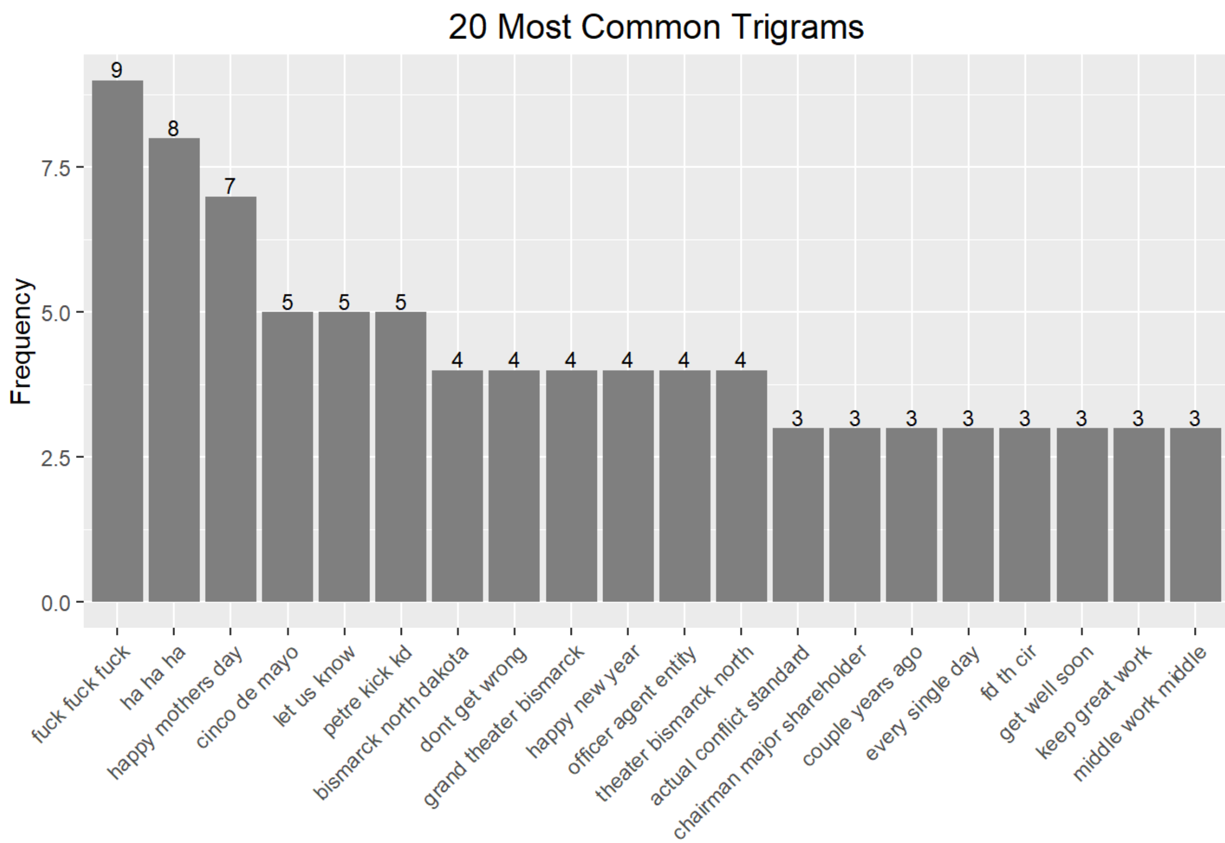
The source code for the word frequency bar chart and constructing the work cloud is attached as A.5 Word Frequencies in the Appendix section. ### Tokenizing and N-Gram Generation

The predictive model I plan to develop for the Shiny application will handle uniqrams, bigrams, and trigrams. In this section, I will use the `RWeka`package to construct functions that tokenize the sample data and construct matrices of uniqrams, bigrams, and trigrams.

Unigrams

Bigrams

Trigrams



The source code for this section is attached as A.6 Tokenizing and N-Gram Generation in the Appendix section. ## Way Forward

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.

Appendix

A.1 Basic Data Summary

Basic summary of the three text corpora.

```
library(stringi)
library(kableExtra)
# assign sample size
#sampleSize = 0.01
sampleSize = 0.002
# 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')
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")
```

A.2 Histogram of Words per Line

Histogram of words per line for the three text corpora.

```
library(ggplot2)
library(gridExtra)
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)))
# free up some memory
rm(plot1, plot2, plot3)
```

A.3 Sample and Clean the Data

```
# set seed for reproducability
set.seed(666)
```

```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 = FALS
E)
# 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)
```

A.4 Build Corpus

```r
library(tm)


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
")
```

```r
# remove profane words from the sample data set
    profanity<-readLines("http://www.cs.cmu.edu/~biglou/resources/bad-words.txt",encoding = "UTF-8", skipNul = TRUE)
    profanity<-profanity[-(which(profanity%in%c("screw","looser","^color")==TRUE))]


    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 = "data/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("data/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)
```

A.5 Word Frequencies

```r
library(wordcloud)
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 = 4
5),
               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)
```

A.6 Tokenizing and N-Gram Generation

Tokenize Functions

```r
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))
```

Unigrams

```r
# create term document matrix for the corpus

unigramMatrix <- TermDocumentMatrix(corpus, control = list(tokenize = unigram
Tokenizer))

# 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 = unigr
amMatrixFreq)

# 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

```r
# create term document matrix for the corpus

bigramMatrix <- TermDocumentMatrix(corpus, control = list(tokenize = bigramTo
kenizer))

# 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 = bigramM
atrixFreq)

# 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 = trigram
Tokenizer))

# 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 = TRUE)

trigramMatrixFreq <- data.frame(word = names(trigramMatrixFreq), freq = trigr
amMatrixFreq)

# 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)
```