# Data Science Capstone Project – Milestone Report

# 1. Introduction

The objective of Data Science Capstone is to apply data science in the area of natural language processing and construct a Shiny application that accepts text input from users and predicts the next word. The provided training data set includes three files containing texts extracted from blogs, news sites and twitter, which are used to build prediction models. This milestone report provides descriptions of basic exploratory analysis on this training data set. First, basic summaries of the three files such as word and line counts are given. Second, basic histograms to illustrate features (n-grams) of the text data are plotted. Finally, conclusions and plans for next steps are discussed.

# 2. Basic Summaries of Corpora

The training data set (avaible here (https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip)) comprises four corpora for four different languages. This report focusses on analyzing the English corpus containing three plain text files: a set of blogs posts, a set of news articles, and a set of Twitter messages. These files are loaded into memory as follows.

```
twitter <- readLines("en_US.twitter.txt", encoding="UTF-8")
news <- readLines("en_US.news.txt", encoding="UTF-8")
blogs <- readLines("en_US.blogs.txt", encoding="UTF-8")
```

## 2.1. Corpus Size

The sizes of these original training data files are around 200 MBs, as summarized in the following table.

```
fileSizeBlogs <- file.info("en_US.blogs.txt")$size   / 1024^2
fileSizeNews <- file.info("en_US.news.txt")$size    / 1024^2
fileSizeTwitter <- file.info("en_US.twitter.txt")$size / 1024^2
fileSize <- c(fileSizeBlogs, fileSizeNews, fileSizeTwitter)
fileName <- c("en_US.blogs.txt", "en_US.news.txt", "en_US.twitter.txt")
fileSizeInfo <- data.frame(fileName, fileSize)
colnames(fileSizeInfo) <- c("File_Name", "File_Size(MB)")
fileSizeInfo
```

```
##           File_Name File_Size(MB)
## 1   en_US.blogs.txt      200.4242
## 2    en_US.news.txt      196.2775
## 3 en_US.twitter.txt      159.3641
```

# 2.2. Statistics of Corpus

Other information such as number of lines, number of non-empty lines, number of characters, and number of non-white characters are also summarized in the following statistic table.

```
library(stringi)
statsBlogs <- stri_stats_general( blogs )
statsNews <- stri_stats_general( news )
statsTwitter <- stri_stats_general( twitter )
stats <- rbind(statsBlogs, statsNews, statsTwitter)
row.names(stats) <- c("en_US.blogs", "en_US.news", "en_US.twitter")
colnames(stats) <- c("Lines", "LinesNonEmpty", "Chars", "CharsNonWhite")
stats
```

```
##                 Lines LinesNonEmpty     Chars CharsNonWhite
## en_US.blogs    899288        899288 206824382     170389539
## en_US.news    1010242       1010242 203223154     169860866
## en_US.twitter 2360148       2360148 162096031     134082634
```

# 2.3. Word Count Distribution of Corpus

To have an overview of word count statistics, the number of words per line in each corpus file is computed. Then, the distibution of these counts per corpus is summarized and plotted.

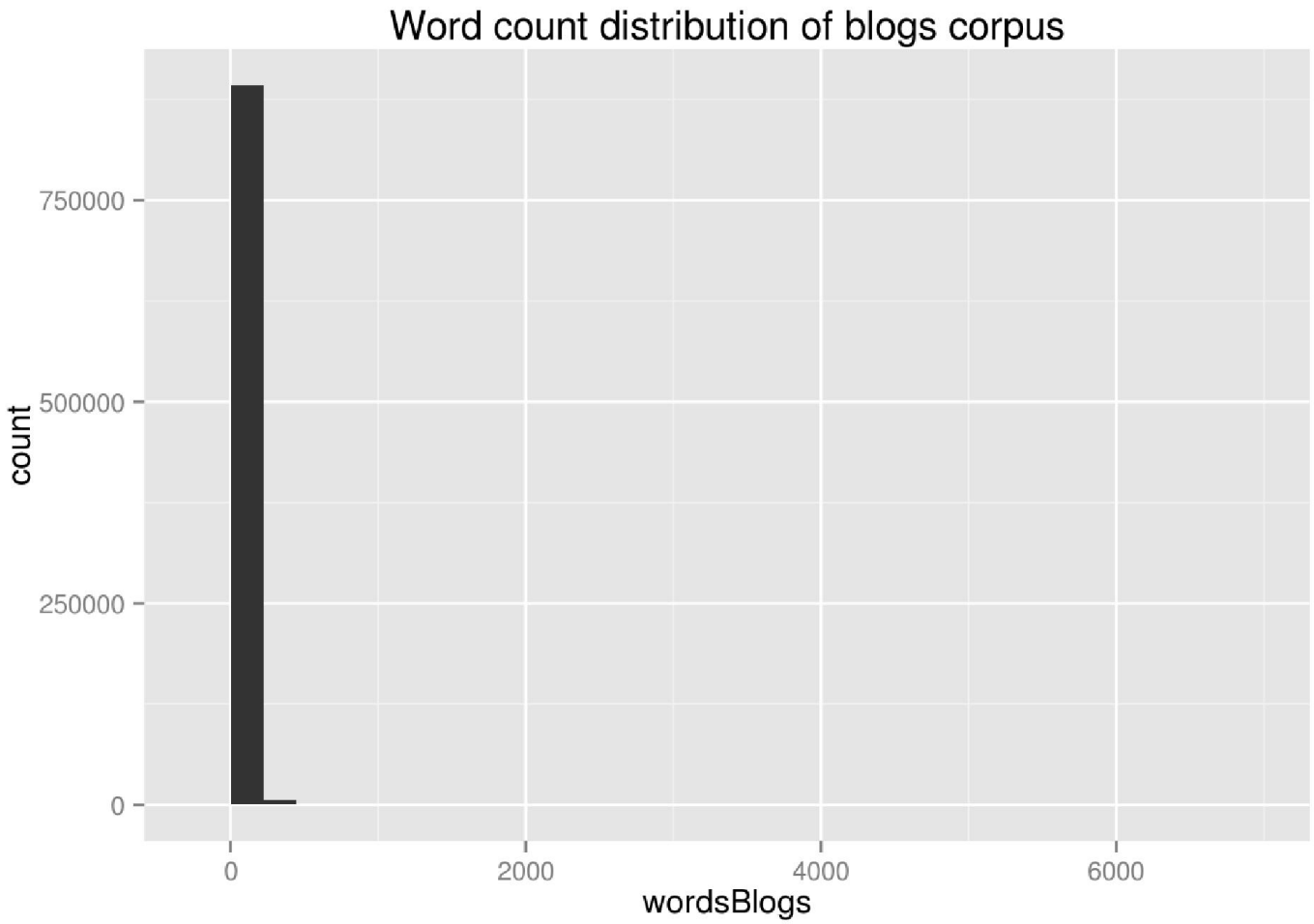
Specifically, word count statistics of blogs corpus is summarized and plotted as below.

```
library(ggplot2)
wordsBlogs <- stri_count_words(blogs)
summary(wordsBlogs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    9.00   28.00   41.75   60.00 6726.00
```

```
qplot(wordsBlogs, main = "Word count distribution of blogs corpus")
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



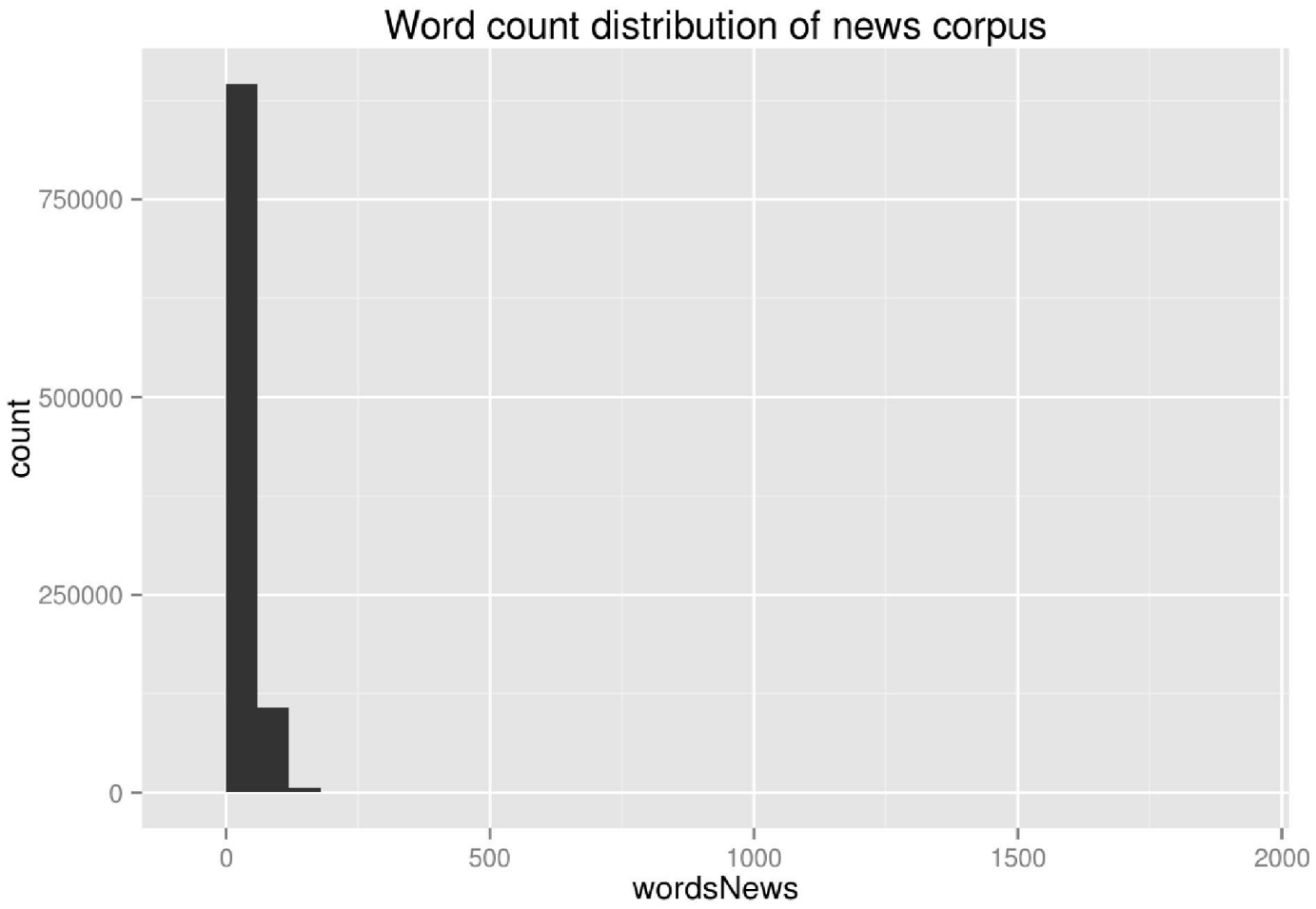

Similarly, word count statistics of news corpus is summarized and plotted as below.

```
wordsNews <- stri_count_words(news)
summary(wordsNews)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   19.00   32.00   34.41   46.00 1796.00
```

```
qplot(wordsNews, main = "Word count distribution of news corpus")
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



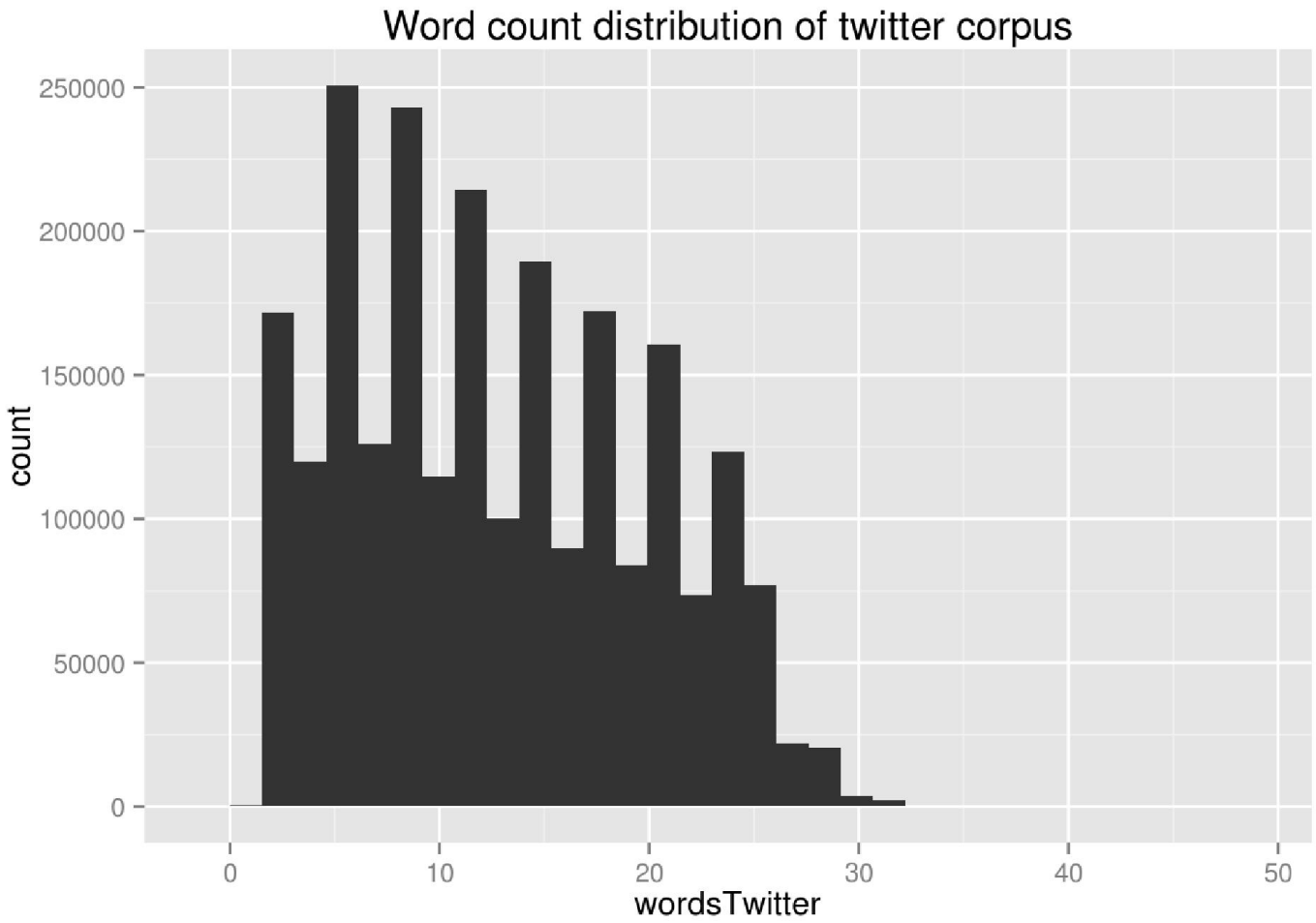

Lastly, word count statistics of twitter corpus is summarized and plotted as below.

```
wordsTwitter <- stri_count_words(twitter)
summary(wordsTwitter)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    7.00   12.00   12.75   18.00   47.00
```

```
qplot(wordsTwitter, main = "Word count distribution of twitter corpus")
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```




# 3. Further Exploratory Data Analysis: n-gram Feature of Corpus

N-gram models are widely used in natural language processing. An n-gram is a contiguous sequence of n words from a given sequence of text. An n-gram of size 1 is referred to as a “unigram”; size 2 is a “bigram” (or a “digram”); size 3 is a “trigram”. In this section, the unigrams, bigrams and trigrams of the copora are studied.

## 3.1. Data Subsetting and Cleansing

Note that building the n-gram models for the entire large corpus is computationally expensive. Therefore, the provided corpora are sampled and combined into a smaller corpus of reasonable size that is sufficient for exploratory data analysis.

```
set.seed(2015)
numSample <- 20000
trainTwitter <- twitter[sample(1:length(twitter), numSample)]
trainNews <- news[sample(1:length(news), numSample)]
trainBlogs <- blogs[sample(1:length(blogs), numSample)]
trainData <- c(trainTwitter, trainNews, trainBlogs)
writeLines(trainData, "./trainData/corpus.txt")
```

The training corpus is loaded into memory. Further, to cleanse the data and make it ready for n-gram analysis, the the following transformations are performed: (1) convert all text to lowercase, (2) remove common punctuation, (3) remove numbers, (4) replace special characters such as /, @, | to whitespace, (5) strip whitespaces, (6) remove stop words, and (7) apply stemming to the words.

```
library(tm)
library(SnowballC)

dir <- file.path(".", "trainData")
docs <- Corpus(DirSource(dir))

docs <- tm_map(docs, content_transformer(tolower))
docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, removeNumbers)

ctf <- content_transformer(function(x, pattern) gsub(pattern, " ", x))
docs <- tm_map(docs, ctf, "/|@|\\|")

docs <- tm_map(docs, stripWhitespace)
docs <- tm_map(docs, removeWords, stopwords("english"))
docs <- tm_map(docs, stemDocument)
```

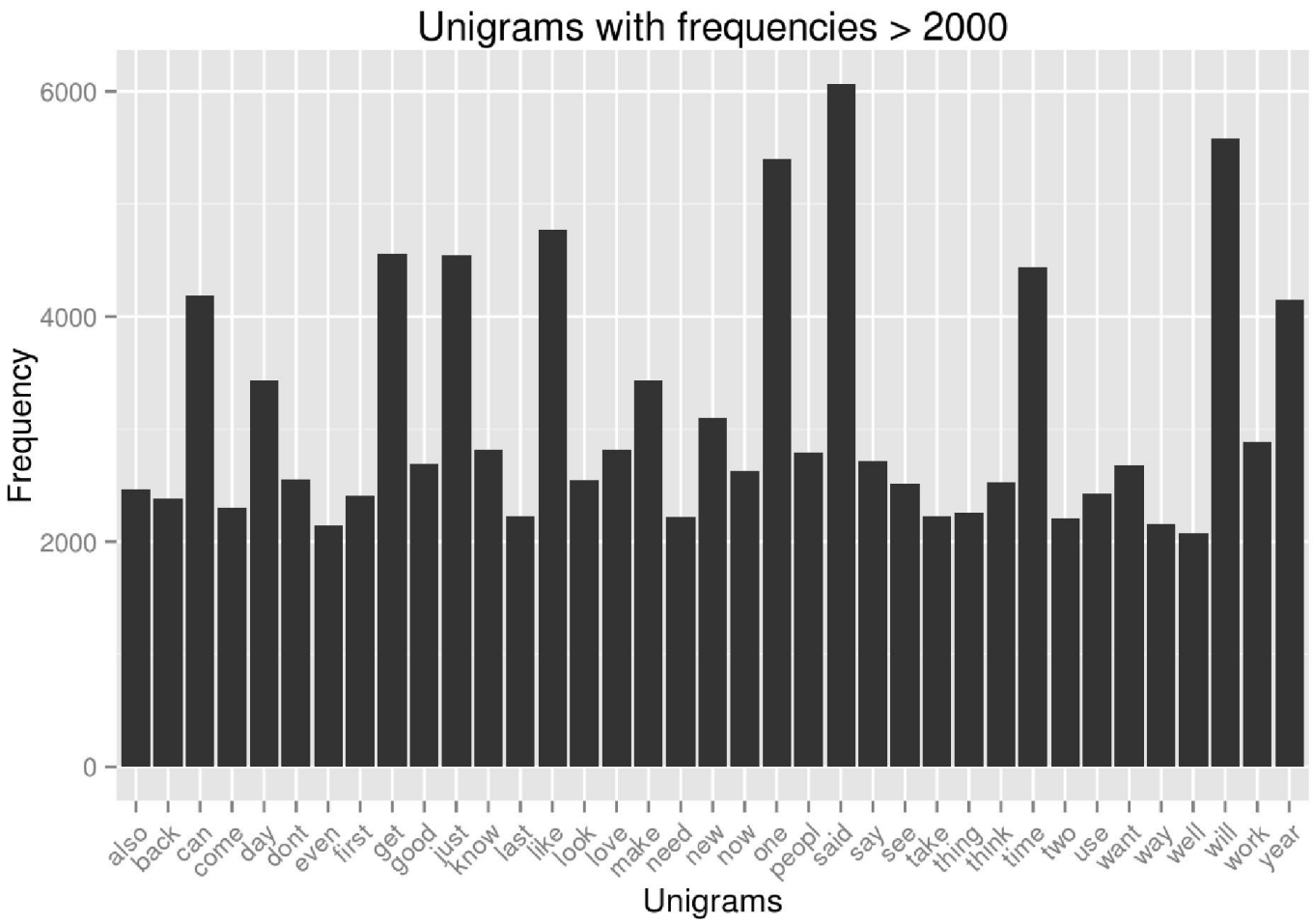
# 3.2. Creation and Analysis of n-gram Models

To explore word sequences and frequencies, three n-grams models including unigrams, bigrams and trigrams are created.

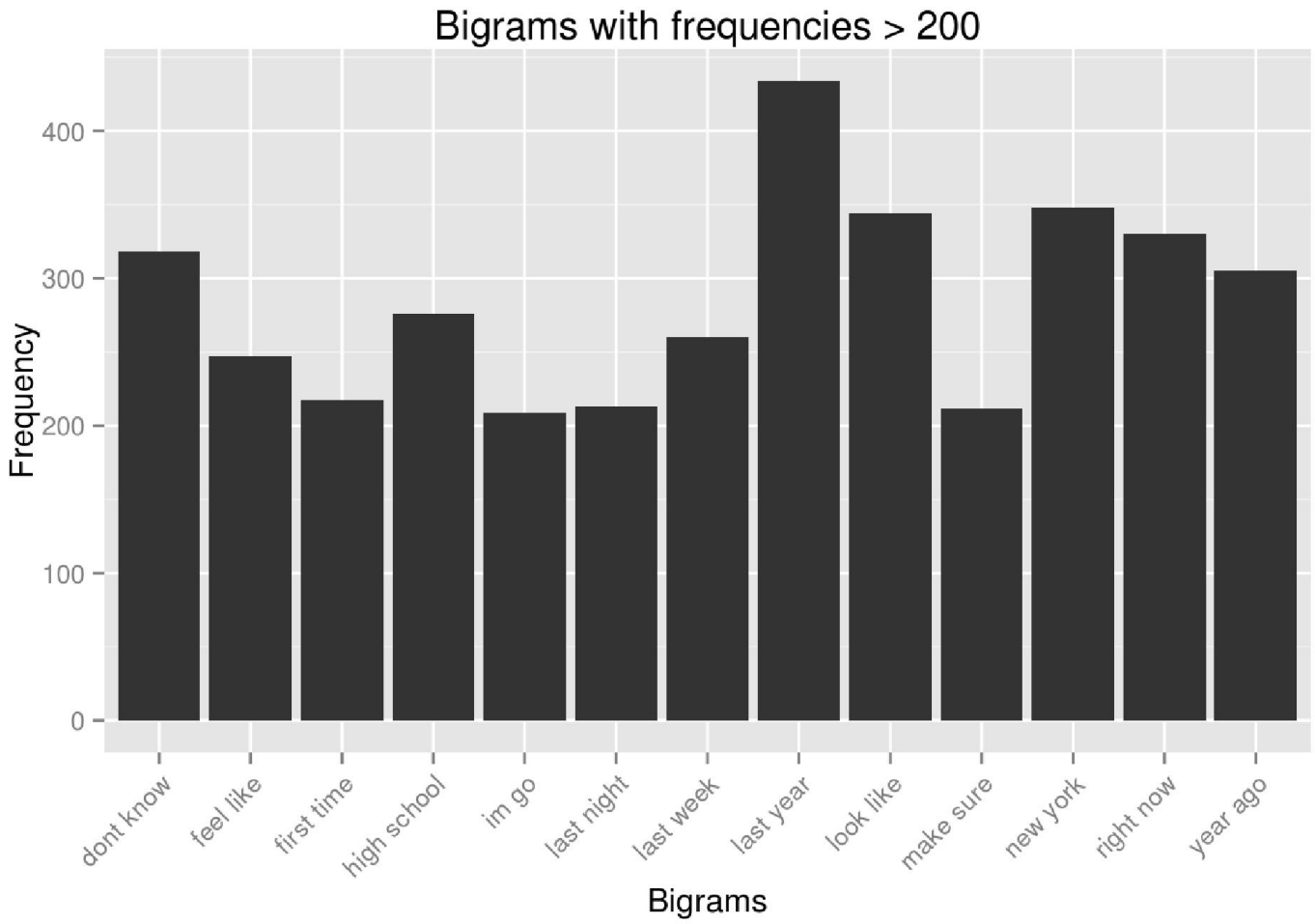
```
library(RWeka)

uniTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 1, max = 1))
uniGrams <- DocumentTermMatrix(docs, control = list(tokenize = uniTokenizer))

biTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
biGrams <- DocumentTermMatrix(docs, control = list(tokenize = biTokenizer))

triTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 3, max = 3))
triGrams <- DocumentTermMatrix(docs, control = list(tokenize = triTokenizer))
```

In addition, n-grams with high frequencies in the corpus are plotted in the following figures.

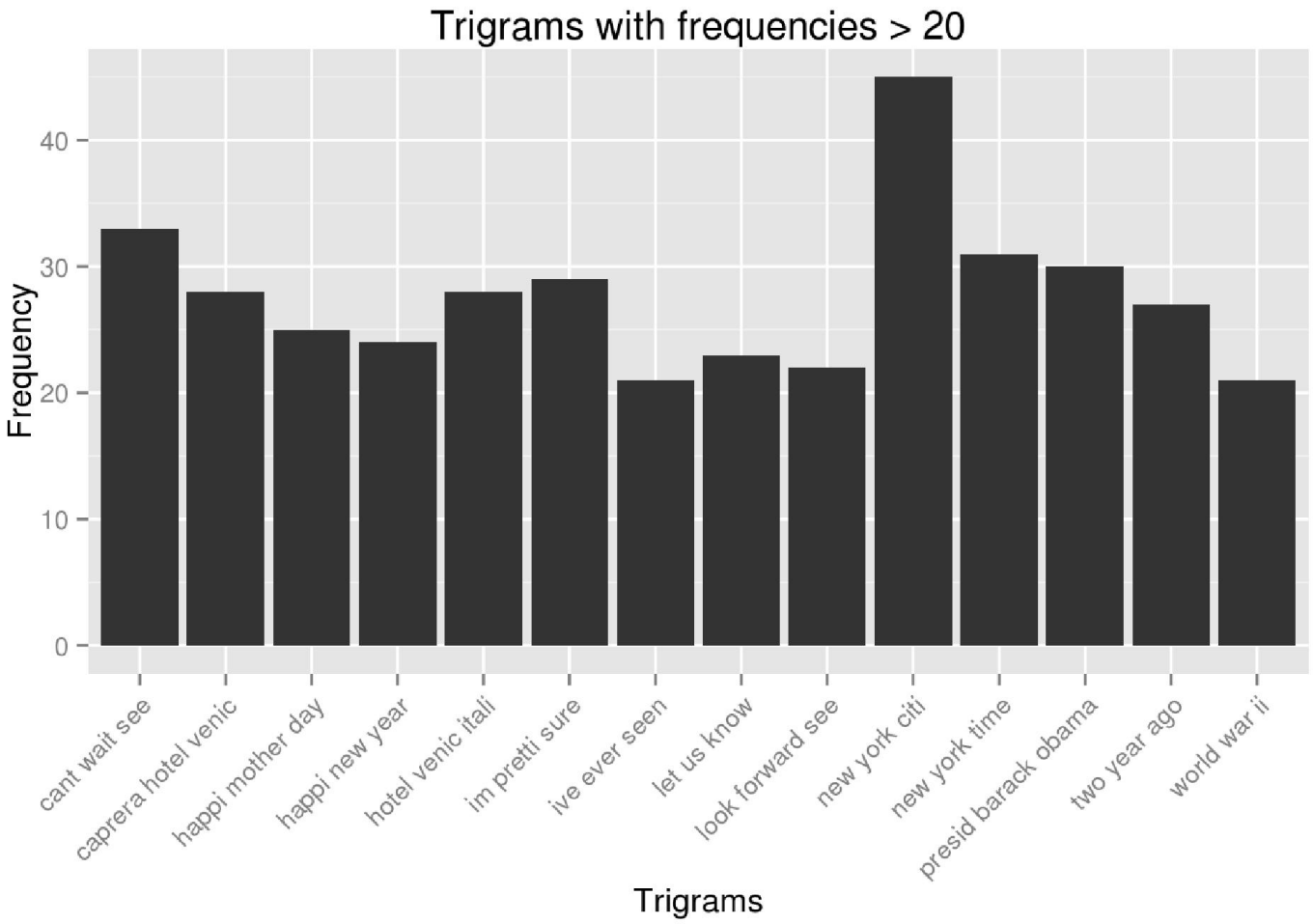
```
library(ggplot2)
library(dplyr)

uniFreq <- sort(colSums(as.matrix(uniGrams)), decreasing=TRUE)
uniWordFreq <- data.frame(word=names(uniFreq), freq=uniFreq)
uniWordFreqPlot <- filter(uniWordFreq, freq > 2000)

ggplot(data = uniWordFreqPlot, aes(word,freq)) +
    geom_bar(stat="identity") +
    ggtitle("Unigrams with frequencies > 2000") +
    xlab("Unigrams") + ylab("Frequency") +
    theme(axis.text.x=element_text(angle=45, hjust=1))
```




```
biFreq <- sort(colSums(as.matrix(biGrams)), decreasing=TRUE)
biWordFreq <- data.frame(word=names(biFreq), freq=biFreq)
biWordFreqPlot <- filter(biWordFreq, freq > 200)

ggplot(data = biWordFreqPlot, aes(word,freq)) +
    geom_bar(stat="identity") +
    ggtitle("Bigrams with frequencies > 200") +
    xlab("Bigrams") + ylab("Frequency") +
    theme(axis.text.x=element_text(angle=45, hjust=1))
```




```
triFreq <- sort(colSums(as.matrix(triGrams)), decreasing=TRUE)
triWordFreq <- data.frame(word=names(triFreq), freq=triFreq)
triWordFreqPlot <- filter(triWordFreq, freq > 20)

ggplot(data = triWordFreqPlot, aes(word,freq)) +
    geom_bar(stat="identity") +
    ggtitle("Trigrams with frequencies > 20") +
    xlab("Trigrams") + ylab("Frequency") +
    theme(axis.text.x=element_text(angle=45, hjust=1))
```




# 4. Findings and Plans

Basic exploratory analysis on the three corpora of US english text provided in this capstone has been done. Besides statistical summaries of the data, advanced features of the text corpora such as unigrams, bigrams, and trigrams are also computed and analyzed.

However, based on the author's experience, these n-gram models are still not sufficient to provide an accurate prediction model for predicting the next word. For example, these n-gram models either predict wrong words or cannot provide any answer to several questions in Quiz 2. Therefore, more advanced models and algorithms are needed in the next step of the capstone.