Data Science Capstone Milestone Report

Carlos Barco

Overview and Introduction:

The aim of this report is to describe the three files that will be used to build the corpus employed to model a predictive algorithm for Swiftkey. This report does the following:

*** Downloads the english text data sets provided from news feeds, blogs and twitter.**

*** Proceseses the data and creates a sample set to analyze.**

*** Runs some summary statistics on the full English data set.**

*** Creates a text corpus that natural langugage processing code can use to mine the text data**

*** Runs some basic exploratory plots using word count and word pairing frequencies**

*** Talks about next steps**

01 Getting the data

Downloaded the zip file containing the text files from https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip (https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip)

02 Data Processing

Once we've configured R to load in necessary software packages, the first step is to load in our text data:

```
## Warning: package 'R.utils' was built under R version 3.4.2
```

```
## Warning: package 'tm' was built under R version 3.4.3
```

```
## Warning: package 'NLP' was built under R version 3.4.1
```

```
## Warning: package 'Rcpp' was built under R version 3.4.3
```

```
## Warning: package 'dplyr' was built under R version 3.4.2
```

Loading the data:

```
# Load in the english versions of our text files:
englishBlogs <- readLines("final/en_US/en_US.blogs.txt", encoding = "UTF-8", skipNul=TRUE
)
englishNews <- readLines("final/en_US/en_US.news.txt", encoding = "UTF-8", skipNul=TRUE)
englishTwitter <- readLines("final/en_US/en_US.twitter.txt", encoding = "UTF-8", skipNul=
TRUE)

# Create an aggregated sample of all of our text data:
SAMPLE_SIZE = 10000
sampleTwitter <- englishTwitter[sample(1:length(englishTwitter),SAMPLE_SIZE)]
sampleNews <- englishNews[sample(1:length(englishNews),SAMPLE_SIZE)]
sampleBlogs <- englishBlogs[sample(1:length(englishBlogs),SAMPLE_SIZE)]
textSample <- c(sampleTwitter,sampleNews,sampleBlogs)

# Write the aggregated sample to a text file:
writeLines(textSample, "sample/textSample.txt")
theSampleCon <- file("sample/textSample.txt")
theSample <- readLines(theSampleCon)
close(theSampleCon)
```

03 Basic Report of Summary Statistics about the data sets:

```
# File Sizes:
englishTwitterSize <- round(file.info("final/en_US/en_US.twitter.txt")$size / (1024*1024)
,0)
englishNewsSize <- round(file.info("final/en_US/en_US.news.txt")$size / (1024*1024),0)
englishBlogsSize <- round(file.info("final/en_US/en_US.blogs.txt")$size / (1024*1024),0)
englishSampleFileSize <- round(file.info("sample/textSample.txt")$size / (1024*1024),0)

# Line Counts:
numEnglishTwitterLines <- countLines("final/en_US/en_US.twitter.txt")[1]
numEnglishNewsLines <- countLines("final/en_US/en_US.news.txt")[1]
numEnglishBlogsLines <- countLines("final/en_US/en_US.blogs.txt")[1]
numEnglishSampleLines <- countLines("sample/textSample.txt")[1]

# Word Counts:
numWordsEnglishTwitter <- as.numeric(system2("wc", args = "-w < final/en_US/en_US.twitte
r.txt", stdout=TRUE))
numWordsEnglishNews <- as.numeric(system2("wc", args = "-w < final/en_US/en_US.news.txt",
 stdout=TRUE))
numWordsEnglishBlog <- as.numeric(system2("wc", args = "-w < final/en_US/en_US.blogs.txt"
, stdout=TRUE))
numWordsEnglishSample <- as.numeric(system2("wc", args = "-w < sample/textSample.txt", st
dout=TRUE))

# Creating a data frame:
fileSummary <- data.frame(
  fileName = c("Blogs","News","Twitter", "Aggregated Sample"),
  fileSize = c(round(englishBlogsSize, digits = 2),
               round(englishNewsSize,digits = 2),
               round(englishTwitterSize, digits = 2),
               round(englishSampleFileSize, digits = 2)),
  lineCount = c(numEnglishBlogsLines, numEnglishNewsLines, numEnglishTwitterLines, numEng
lishSampleLines),
  wordCount = c(numWordsEnglishBlog, numWordsEnglishNews, numWordsEnglishTwitter, numWord
sEnglishSample)
)
colnames(fileSummary) <- c("Name", "Size", "Num Lines", "Num Words")

fileSummary
```

```
##                Name Size Num Lines Num Words
## 1             Blogs  200    899288  37334690
## 2              News  196   1010242  34372720
## 3           Twitter  159   2360148  30374206
## 4 Aggregated Sample    5     30000    881240
```

04 Create and Clean Corpus:

From our sample text file we can create a text corpus, in order to give to our natural language processing code the tools for conduct the word analysis:

```
# Setup The Text Mining Class:
cname <- file.path(".", "sample")
finalCorpus <- Corpus(DirSource(cname))

# Convert corpus to lowercase:
finalCorpus <- tm_map(finalCorpus, content_transformer(tolower))

# Remove more transforms:
toSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))
finalCorpus <- tm_map(finalCorpus, toSpace, "/|@|\\|")

# Remove punctuation:
finalCorpus <- tm_map(finalCorpus, removePunctuation)

# Remove numbers:
finalCorpus <- tm_map(finalCorpus, removeNumbers)

# Strip whitespace:
finalCorpus <- tm_map(finalCorpus, stripWhitespace)

# Initiate stemming:
finalCorpus <- tm_map(finalCorpus, stemDocument)
```

04 Create Our 'N-Grams' for Exploratory Data Analysis:

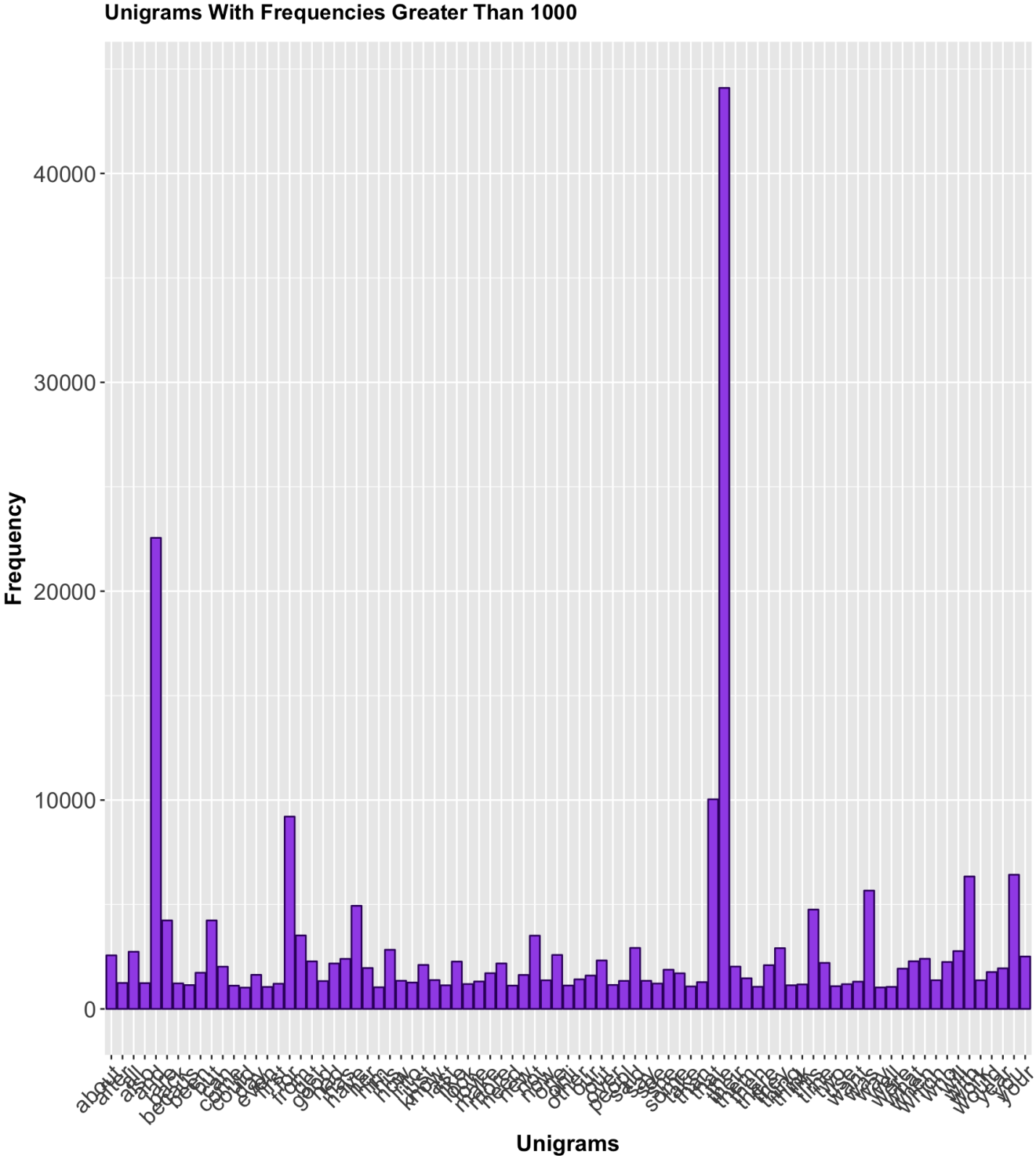
Next we create 'N-Grams'

```
# Create a unigram:
unigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 1, max = 1))
unigram <- DocumentTermMatrix(finalCorpus, control = list(tokenize = unigramTokenizer))
unigramFreq <- sort(colSums(as.matrix(unigram)), decreasing=TRUE)
unigramWordFreq <- data.frame(word=names(unigramFreq), freq=unigramFreq)

# Create a bigram:
bigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
bigram <- DocumentTermMatrix(finalCorpus, control = list(tokenize = bigramTokenizer))
bigramFreq <- sort(colSums(as.matrix(bigram)), decreasing=TRUE)
bigramWordFreq <- data.frame(word=names(bigramFreq), freq=bigramFreq)

# Create a trigram:
trigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 3, max = 3))
trigram <- DocumentTermMatrix(finalCorpus, control = list(tokenize = trigramTokenizer))
trigramFreq <- sort(colSums(as.matrix(trigram)), decreasing=TRUE)
trigramWordFreq <- data.frame(word=names(trigramFreq), freq=trigramFreq)
```

05 Exploratory Plots:

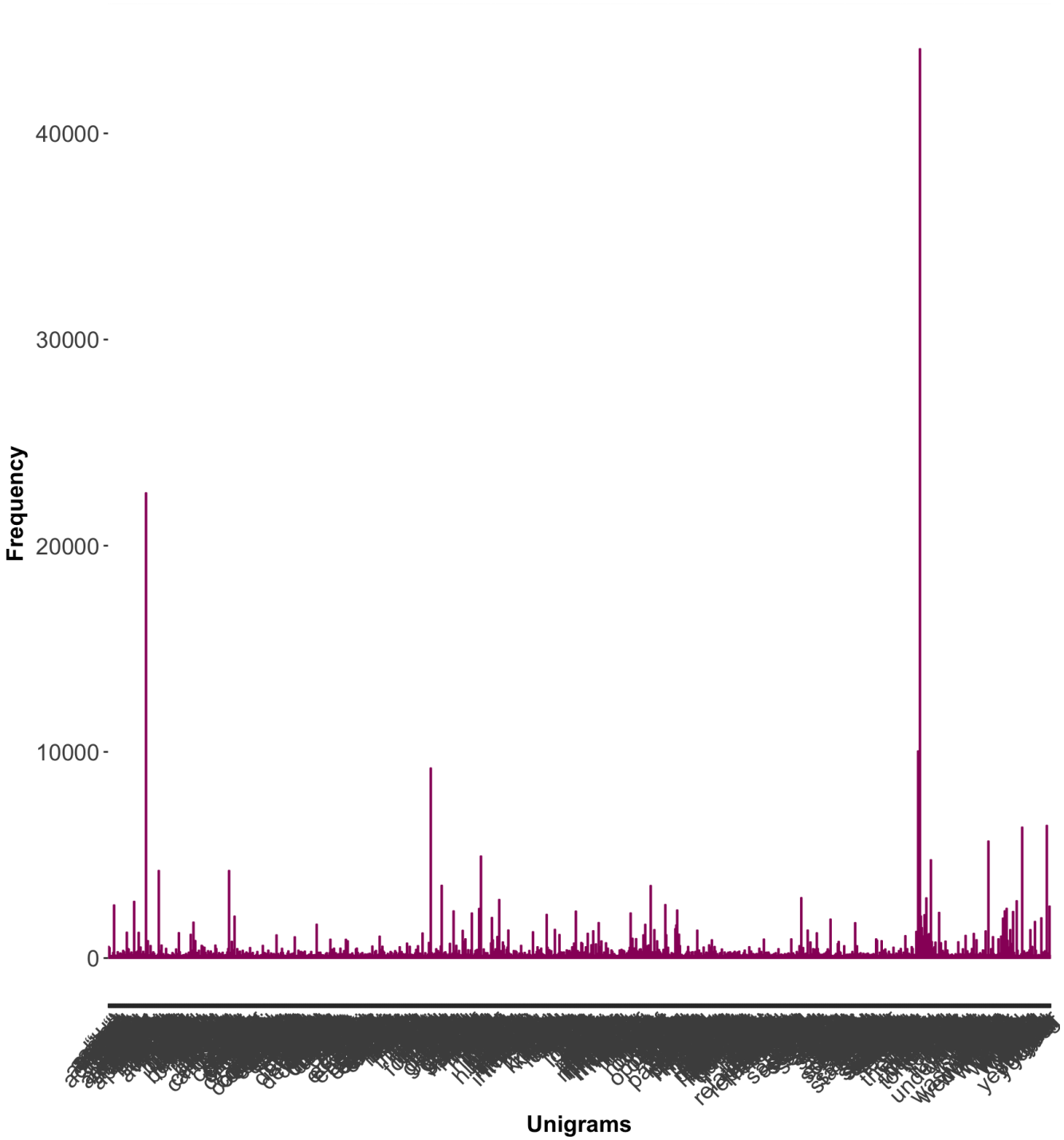
Unigrams: How many times do we see one word repeated in the text corpus.

```
# First Create A Plot of Our Unigrams:
unigramWordFreq %>% filter(freq > 1000) %>% ggplot(aes(word,freq)) +
  geom_bar(stat="identity", colour="#37006b", fill="#a257e9") +
  ggtitle("Unigrams With Frequencies Greater Than 1000") +
  xlab("Unigrams") + ylab("Frequency") +
  theme(axis.text.x=element_text(angle=45, hjust=1)) +
  theme(axis.text=element_text(size=14), axis.title=element_text(size=14,face="bold")) +
  theme(plot.title = element_text(lineheight=1.8, face="bold", vjust=3))
```



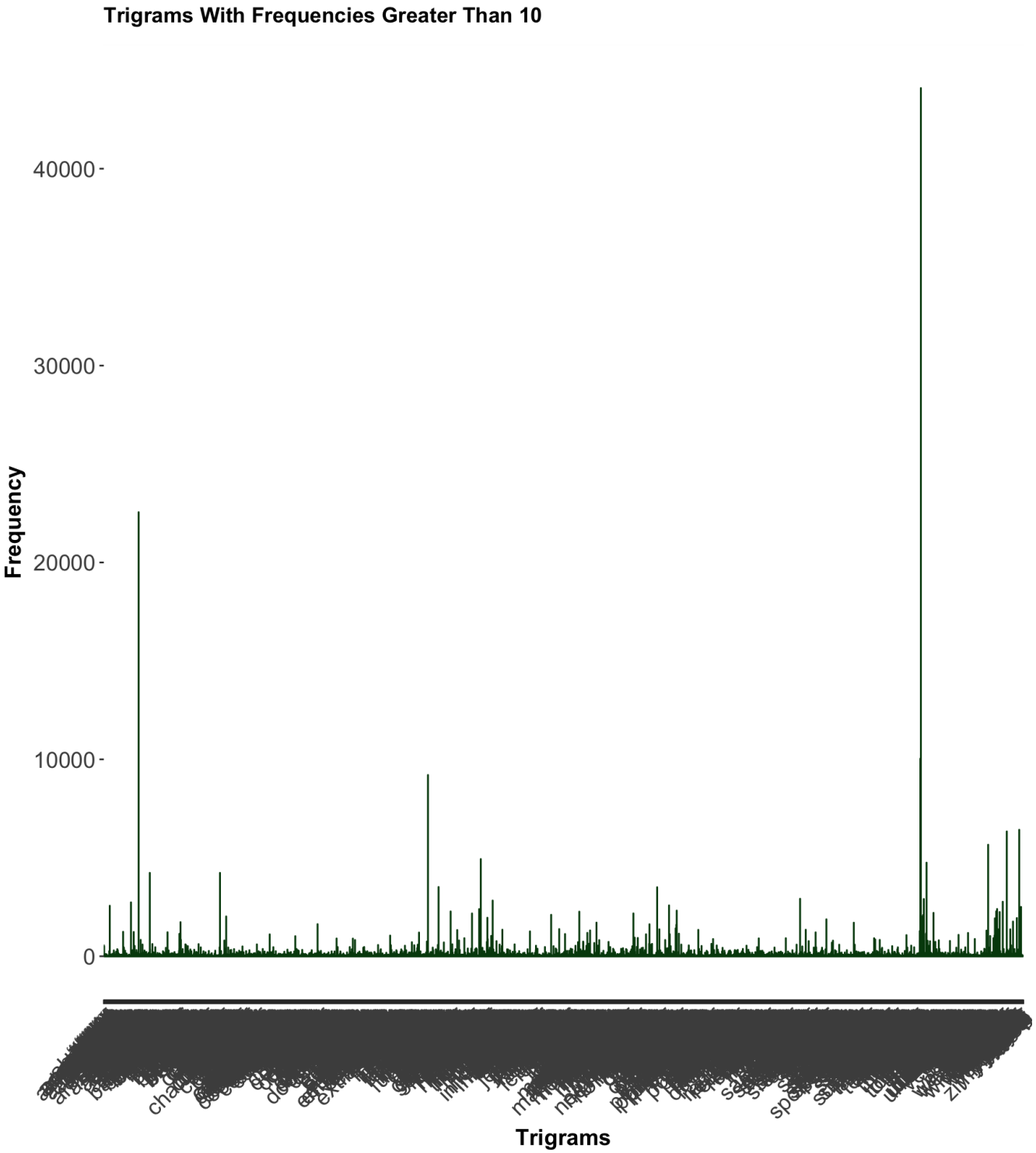
Bigrams: Sequence of two words.

```
# Plot bigrams:
bigramWordFreq %>% filter(freq > 100) %>% ggplot(aes(word,freq)) +
  geom_bar(stat="identity", colour="#990068", fill="#cf6aaf") +
  ggtitle("Bigrams With Frequencies Greater Than 100") +
  xlab("Unigrams") + ylab("Frequency") +
  theme(axis.text.x=element_text(angle=45, hjust=1)) +
  theme(axis.text=element_text(size=14), axis.title=element_text(size=14,face="bold")) +
  theme(plot.title = element_text(lineheight=1.8, face="bold", vjust=3))
```



Trigrams: sequence of three words.

```
# Plot trigrams:
trigramWordFreq %>% filter(freq > 10) %>% ggplot(aes(word,freq)) +
  geom_bar(stat="identity", colour="#00470d", fill="#4ebc63") +
  ggtitle("Trigrams With Frequencies Greater Than 10") +
  xlab("Trigrams") + ylab("Frequency") +
  theme(axis.text.x=element_text(angle=45, hjust=1)) +
  theme(axis.text=element_text(size=14), axis.title=element_text(size=14,face="bold")) +
  theme(plot.title = element_text(lineheight=1.8, face="bold", vjust=3))
```



06 Exploratory Plots 2:

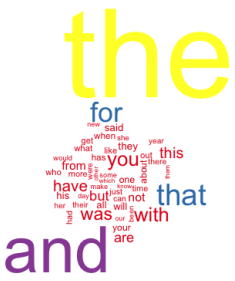
Word cloud plots inside our n-grams.

Unigram Word Cloud Plot:

```
set.seed(1991)
wordcloud(names(unigramFreq), unigramFreq, max.words=50, scale=c(5, .1), colors=brewer.pa
l(6, "Paired"))
```

Bigram Word Cloud Plot:

```
wordcloud(names(bigramFreq), bigramFreq, max.words=50, scale=c(5, .1), colors=brewer.pal(
6, "Set1"))
```



Trigram Word Cloud Plot:

```
wordcloud(names(trigramFreq), trigramFreq, max.words=50, scale=c(5, .1), colors=brewer.pa
l(6, "Dark2"))
```

07 Conclusions and Next Steps:

We follow with a prediction application. I'll have to create a prediction algorithm and ensure that it runs quickly for acceptable use as a web product. Also I might have to find a way than just using N-Gram tokenization to predict the next word in a sequence of words. The deployed Shiny app should ideally satisfy all of the algorithm requirements.