Data Science Capstone Week 2 Milestone Report

Nikhil

February 17, 2017

1. Introduction

This is a Week 2 Milestone Report for Coursera Data Science Capstone Project. The objective for week 2 is to do the Exploratory Data Analysis. This will further help us to build our own word prediction app and algorithm based on the current word user is typing. The Milestone Report is divided into following main sections.

- Getting and Reading the Data Sets
- Summarizing the Data Sets
- Sampling and Cleaning the Data Sets
- Plotting Sampled Corpus Data with Wordcloud
- Plotting NGrams Tokenization
- Further Development Plan

2. Getting and Reading the Data Sets

```
library (knitr)
library (dplyr)
library (doParallel)
library (stringi)
library (tm)
library (ggplot2)
library (wordcloud)
library (SnowballC)
library (slam)
library (qdap)
library (rJava)
library (RWeka)
echo = FALSE  # Make code always visible
options(scipen = 1)  # Turn off scientific notations for numbers

# downloading the data set zip file
wd_path = "d:/GitHubRepositories/Coursera/Data Science/10. Capstone Project/"
if (file.exists(paste(wd_path,"Coursera-SwiftKey.zip",sep=""))==FALSE)
{
  #setwd(wd_path)
  download.file("https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip",
                destfile = "Coursera-SwiftKey.zip")
  unzip("Coursera-SwiftKey.zip")
}
# setting the paths for data set text files
```

```
#setwd(wd_path)
blogs_path = paste(wd_path, "final/en_US/en_US.blogs.txt", sep="")
news_path = paste(wd_path, "final/en_US/en_US.news.txt", sep="")
twitter_path = paste(wd_path, "final/en_US/en_US.twitter.txt", sep="")

# making connections and reading the blog text file into data set objects
conn = file(blogs_path,open="rb")
blogs_dataset = readLines (conn, encoding="UTF-8")
close (conn)

# making connections and reading the news text file into data set objects
conn = file(news_path,open="rb")
news_dataset = readLines (conn, encoding="UTF-8")
close (conn)

# making connections and reading the twitter text file into data set objects
conn = file(twitter_path,open="rb")
twitter_dataset = readLines (conn, encoding="UTF-8")
close (conn)

rm(conn)
```

3. Summarizing the Data Sets

```
# lineCntBlogs = length(blogs_dataset)
# lineCntNews = length(news_dataset)
# lineCntTwitter = length(twitter_dataset)
#
# wordCntBlogs = sum(sapply(gregexpr("\\W+", blogs_dataset), length))+1
# wordCntNews = sum(sapply(gregexpr("\\W+", news_dataset), length))+1
# wordCntTwitter = sum(sapply(gregexpr("\\W+", twitter_dataset), length))+1
#
# longestLineBlog = max(nchar(blogs_dataset))
# longestLineNews = max(nchar(news_dataset))
# longestLineTwitter = max(nchar(twitter_dataset))


kable(data.frame (
  file.Name = c("en_US.blogs.txt", "en_US.news.txt", "en_US.twitter.txt"),
  file.Size.in.MB = format(c (file.info(blogs_path)$size/(1024^2),
                       file.info(news_path)$size/(1024^2),
                       file.info(twitter_path)$size/(1024^2)), big.mark = ","),
  format(t(rbind(sapply(list(blogs_dataset,news_dataset,twitter_dataset),stri_stats_general),
  WordCount = sapply(list(blogs_dataset,news_dataset,twitter_dataset),stri_stats_latex)[4,])),
  big.mark=",")#,
  # total.Lines = format(c(lineCntBlogs, lineCntNews, lineCntTwitter), big.mark = ","),
  # total.Word.Count = format(
  #                     c(wordCntBlogs, wordCntNews, wordCntTwitter), big.mark = ","),
  # avg.Words.per.Line = format (
  #                      c(
  #                        (wordCntBlogs / lineCntBlogs),
  #                        (wordCntNews / lineCntNews),
```

```
  #                         (wordCntTwitter / lineCntTwitter)
  #                        )
  #                       , big.mark = ","),
  # max.chars.per.Line = format (
  #                       c(longestLineBlog, longestLineNews, longestLineTwitter)
  #                       , big.mark = ",")
))
```

file.Name	file.Size.in.MB	Lines	LinesNEmpty	Chars	CharsNWhite	WordCount
en_US.blogs.txt	200.4242	899,288	899,288	206,824,382	170,389,539	37,570,839
en_US.news.txt	196.2775	1,010,242	1,010,242	203,223,154	169,860,866	34,494,539
en_US.twitter.txt	159.3641	2,360,148	2,360,148	162,096,031	134,082,634	30,451,128

4. Sampling and Cleaning the Data Sets

Sampling Data Set is created by choosing 5% rows of the each and every actual data set. This sampling percentage can be changed by changing `samplingPercentage` parameter.

```
set.seed(5000) # setting the random seed for reproducibility
samplingPercentage = 0.01 # sampling percentage, can be changed depending up on the results

# creating sampling dataset
sample_blogs = sample(blogs_dataset,length(blogs_dataset)*samplingPercentage)
sample_news = sample(news_dataset,length(news_dataset)*samplingPercentage)
sample_twitter = sample(twitter_dataset,length(twitter_dataset)*samplingPercentage)
sample_dataset = c(sample_blogs,
                   sample_news,
                   sample_twitter)

# converting character vector encoding
sample_dataset = iconv(sample_dataset,"UTF-8","ASCII", sub="byte")
sample_blogs = iconv(sample_blogs,"UTF-8","ASCII", sub="byte")
sample_news = iconv(sample_news,"UTF-8","ASCII", sub="byte")
sample_twitter = iconv(sample_twitter,"UTF-8","ASCII", sub="byte")

#rm(blogs_dataset)
#rm(news_dataset)
#rm(twitter_dataset)

# save the sample data set into file
writeLines(sample_dataset,"./sample_dataset.txt")
```

Cleaning Data will involve
- Changing all alphabetic data to lower case
- Remvoing punctuations from the text
- Removing numbers from the text
- Stripping off extra whitespaces
- Removing Profane words
- Removing Stop words
- Performing Stemming
- Converting data into Plain Text format
- Calculating document term frequencies for Corpus

```
corpus_dataset = list ()
corpus_blogs = list()
corpus_news = list()
corpus_twitter = list()
dtMatrix = list()
dtMatrix_blogs = list()
dtMatrix_news = list()
dtMatrix_twitter = list()
# for (i in 1:length(sample_dataset))
# {
#   corpus_dataset[[i]] = Corpus(VectorSource(sample_dataset[[i]]))
#
#   # Changing all alphabetic data to lower case
#   corpus_dataset[[i]] = tm_map(corpus_dataset[[i]], tolower)
#
#   # Remvoing punctuations from the text
#   corpus_dataset[[i]] = tm_map(corpus_dataset[[i]], removePunctuation)
#
#   # Stripping off extra whitespaces
#   corpus_dataset[[i]] = tm_map(corpus_dataset[[i]], stripWhitespace)
#
#   # Removing Profane words
#   profanewords = readLines ("./badWords.txt")
#   corpus_dataset[[i]] = tm_map(corpus_dataset[[i]], removeWords, profanewords)
#
#   # Removing Stop words
#   corpus_dataset[[i]] = tm_map(corpus_dataset[[i]], removeWords, stopwords("english"))
#
#   # Performing Stemming
#   corpus_dataset[[i]] = tm_map(corpus_dataset[[i]], stemDocument)
#
#   # Converting data into Plain Text format
#   corpus_dataset[[i]] = tm_map(corpus_dataset[[i]], PlainTextDocument)
#
#   # Calculating document term frequencies for Corpus
#   dtMatrix[[i]] = DocumentTermMatrix(corpus_dataset[[i]],control = list(wordLengths=c(0,Inf)))
# }
corpus_dataset = Corpus(VectorSource(sample_dataset))
corpus_blogs = Corpus (VectorSource(sample_blogs))
corpus_news = Corpus (VectorSource(sample_news))
corpus_twitter = Corpus (VectorSource(sample_twitter))

# Changing all alphabetic data to lower case
corpus_dataset = tm_map(corpus_dataset, tolower)
corpus_blogs = tm_map(corpus_blogs, tolower)
corpus_news = tm_map(corpus_news, tolower)
corpus_twitter = tm_map(corpus_twitter, tolower)

# Remvoing punctuations from the text
corpus_dataset = tm_map(corpus_dataset, removePunctuation)
corpus_blogs = tm_map(corpus_blogs, removePunctuation)
corpus_news = tm_map(corpus_news, removePunctuation)
corpus_twitter = tm_map(corpus_twitter, removePunctuation)
```

```
# Stripping off extra whitespaces
corpus_dataset = tm_map(corpus_dataset, stripWhitespace)
corpus_blogs = tm_map(corpus_blogs, stripWhitespace)
corpus_news = tm_map(corpus_news, stripWhitespace)
corpus_twitter = tm_map(corpus_twitter, stripWhitespace)

# Removing Profane words
profanewords = readLines ("./badWords.txt")
corpus_dataset = tm_map(corpus_dataset, removeWords, profanewords)
corpus_blogs = tm_map(corpus_blogs, removeWords, profanewords)
corpus_news = tm_map(corpus_news, removeWords, profanewords)
corpus_twitter = tm_map(corpus_twitter, removeWords, profanewords)

# Removing Stop words
corpus_dataset = tm_map(corpus_dataset, removeWords, stopwords("english"))
corpus_blogs = tm_map(corpus_blogs, removeWords, stopwords("english"))
corpus_news = tm_map(corpus_news, removeWords, stopwords("english"))
corpus_twitter = tm_map(corpus_twitter, removeWords, stopwords("english"))

# Performing Stemming
corpus_dataset = tm_map(corpus_dataset, stemDocument)
corpus_blogs = tm_map(corpus_blogs, stemDocument)
corpus_news = tm_map(corpus_news, stemDocument)
corpus_twitter = tm_map(corpus_twitter, stemDocument)

# Converting data into Plain Text format
corpus_dataset = tm_map(corpus_dataset, PlainTextDocument)
corpus_blogs = tm_map(corpus_blogs, PlainTextDocument)
corpus_news = tm_map(corpus_news, PlainTextDocument)
corpus_twitter = tm_map(corpus_twitter, PlainTextDocument)

# Calculating document term frequencies for Corpus
dtMatrix = DocumentTermMatrix(corpus_dataset, control = list(wordLengths=c(0,Inf)))
dtMatrix_blogs = DocumentTermMatrix(corpus_blogs, control = list(wordLengths=c(0,Inf)))
dtMatrix_news = DocumentTermMatrix(corpus_news, control = list(wordLengths=c(0,Inf)))
dtMatrix_twitter = DocumentTermMatrix(corpus_twitter, control = list(wordLengths=c(0,Inf)))
```

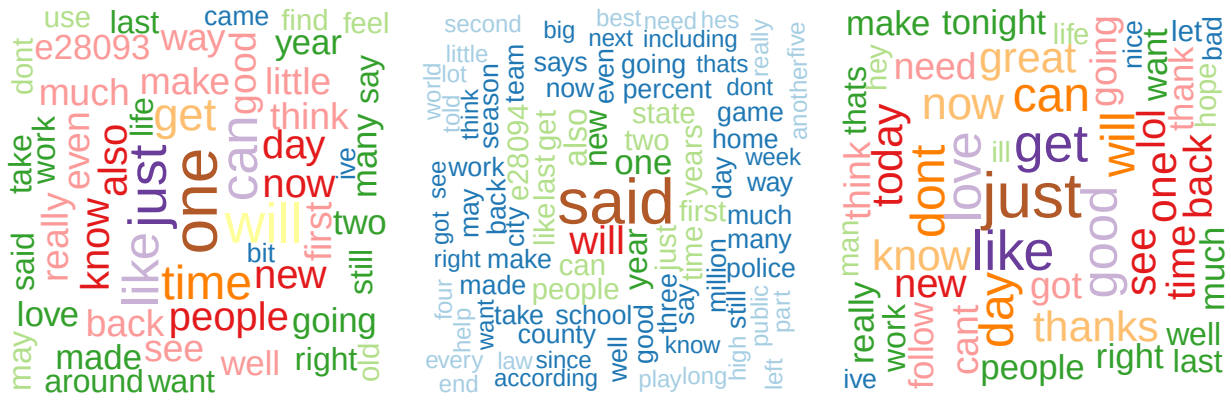
5. Plotting Sampled Corpus Data with Wordcloud

Plotting the individual wordclouds for Blogs, News and Twitter against the combined one helps us to the see impact of individual vis-a-vis combined data set. It will also help to predict different words depending upon the context.

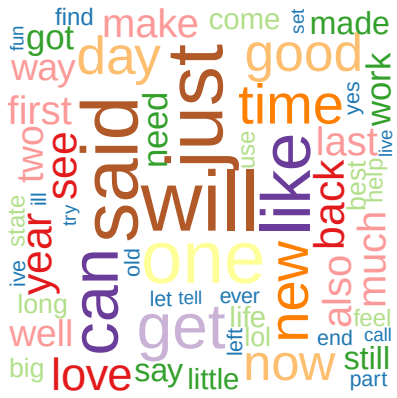
```
# Set random seed for reproducibility
set.seed(5000)
# Set Plotting in 1 row 3 columns
par(mfrow=c(1, 3))

wordcloud(corpus_blogs, max.words=100, random.order = FALSE, scale=c(3,1),
          rot.per=0.45, use.r.layout=FALSE, colors=brewer.pal(12,"Paired"))
title ("Word Cloud - US English Blogs")
wordcloud(corpus_news, max.words=100, random.order = FALSE, scale=c(3,1),
```

```
          rot.per=0.45, use.r.layout=FALSE, colors=brewer.pal(12,"Paired"))
title ("Word Cloud - US English News")
wordcloud(corpus_twitter, max.words=100, random.order = FALSE, scale=c(3,1),
          rot.per=0.45, use.r.layout=FALSE, colors=brewer.pal(12,"Paired"))
title ("Word Cloud - US English Twitter")
```

Word Cloud − US English Blog **Word Cloud − US English New** **Word Cloud − US English Twitt**




```
# Headings= c("Word Cloud - US English Blogs",
#             "Word Cloud - US English News",
#             "Word Cloud - US English Twitter")
#
# # Iterate each corpus and DTM and plot word cloud (Max = 100)
# for (i in 1:length(corpus_dataset)) {
#     wordcloud(words = colnames(dtMatrix[[i]]), freq = slam::col_sums(dtMatrix[[i]]),
#         scale = c(3, 1), max.words = 100, random.order = FALSE, rot.per = 0.45,
#         use.r.layout = FALSE, colors = brewer.pal(12, "Paired"))
#     title(Headings[i])
# }
wordcloud(corpus_dataset, max.words=200, random.order = FALSE,
          rot.per=0.45, use.r.layout=FALSE, colors=brewer.pal(12,"Paired"))
```



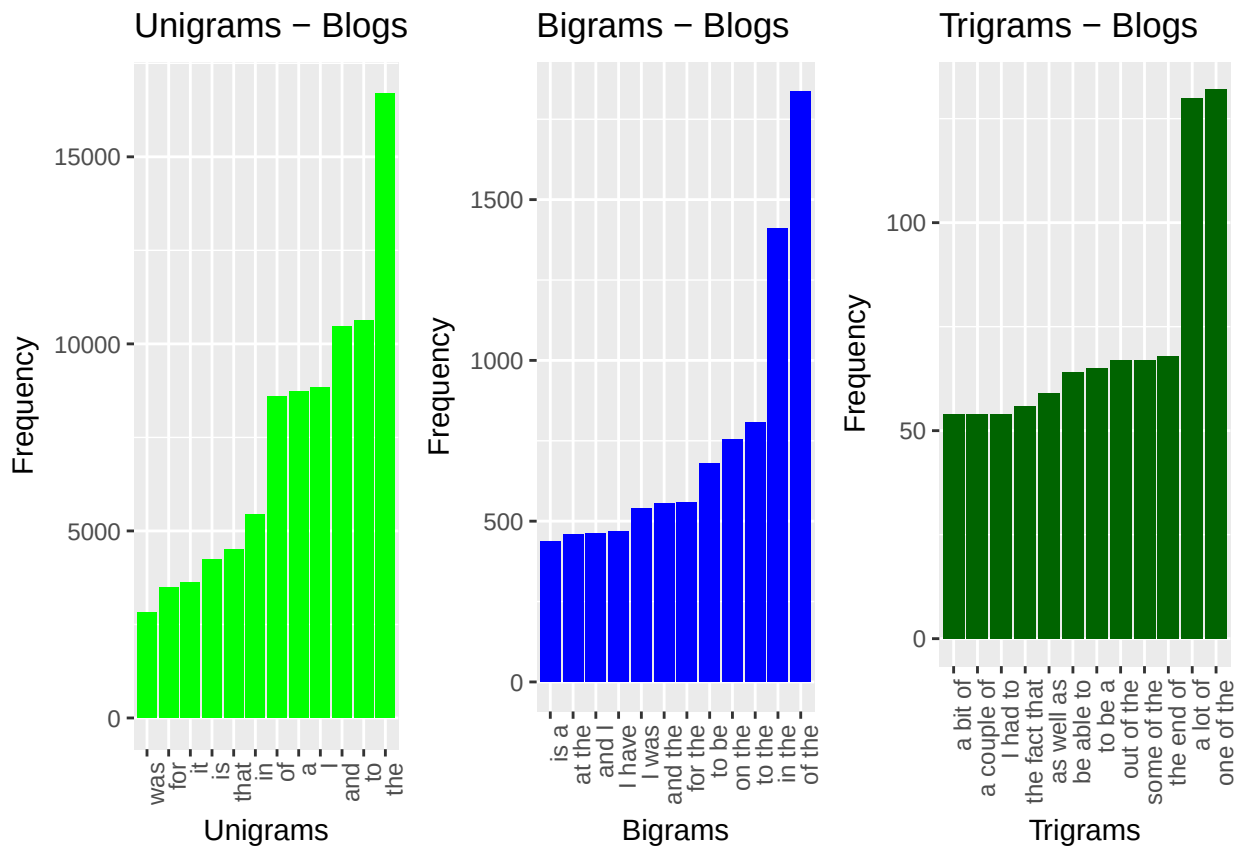
6. Plotting NGrams Tokenization

```
.jinit(parameters = "-Xmx128g")
```

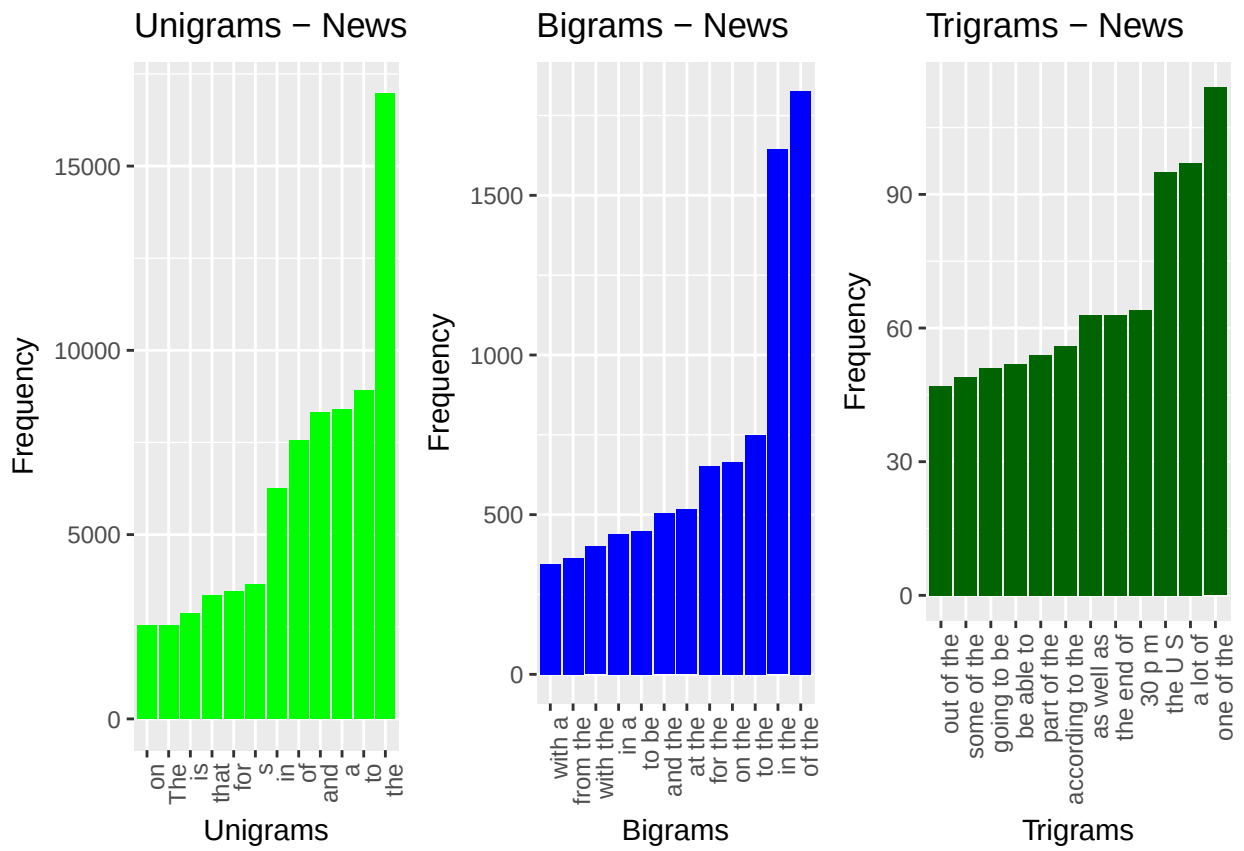
```
## [1] 0
```

```
# Define a function to make Unigram, Bigram and Trigram from the corpus
# And then Plot them together with ggplot2 and gridExtra packages
plot.Grams <- function (x=sampleBlogs, subTitle="Blogs", N=10)
{
  # Use RWeka to get unigram token
  Tokenizer1 <- RWeka::NGramTokenizer(x, Weka_control(min = 1, max = 1))
  Gram.1 <- data.frame(table(Tokenizer1))
  Gram.1 <- Gram.1[order(Gram.1$Freq, decreasing = TRUE),]
  colnames(Gram.1) <- c("Word", "Freq")
  Gram.1 <- head(Gram.1, N)
  g1 <- ggplot(Gram.1, aes(x=reorder(Word, Freq),y=Freq)) +
        geom_bar(stat="identity", fill="green") +
        ggtitle(paste("Unigrams", "-", subTitle)) +
        xlab("Unigrams") + ylab("Frequency") +
        theme(axis.text.x=element_text(angle=90, hjust=1))

  # Use RWeka to get bigram token
  Tokenizer2 <- RWeka::NGramTokenizer(x,
                                      Weka_control(min = 2,
                                                   max = 2,
                                                   delimiters = " \\r\\n\\t.,;:\"()?!"))
  Gram.2 <- data.frame(table(Tokenizer2))
```

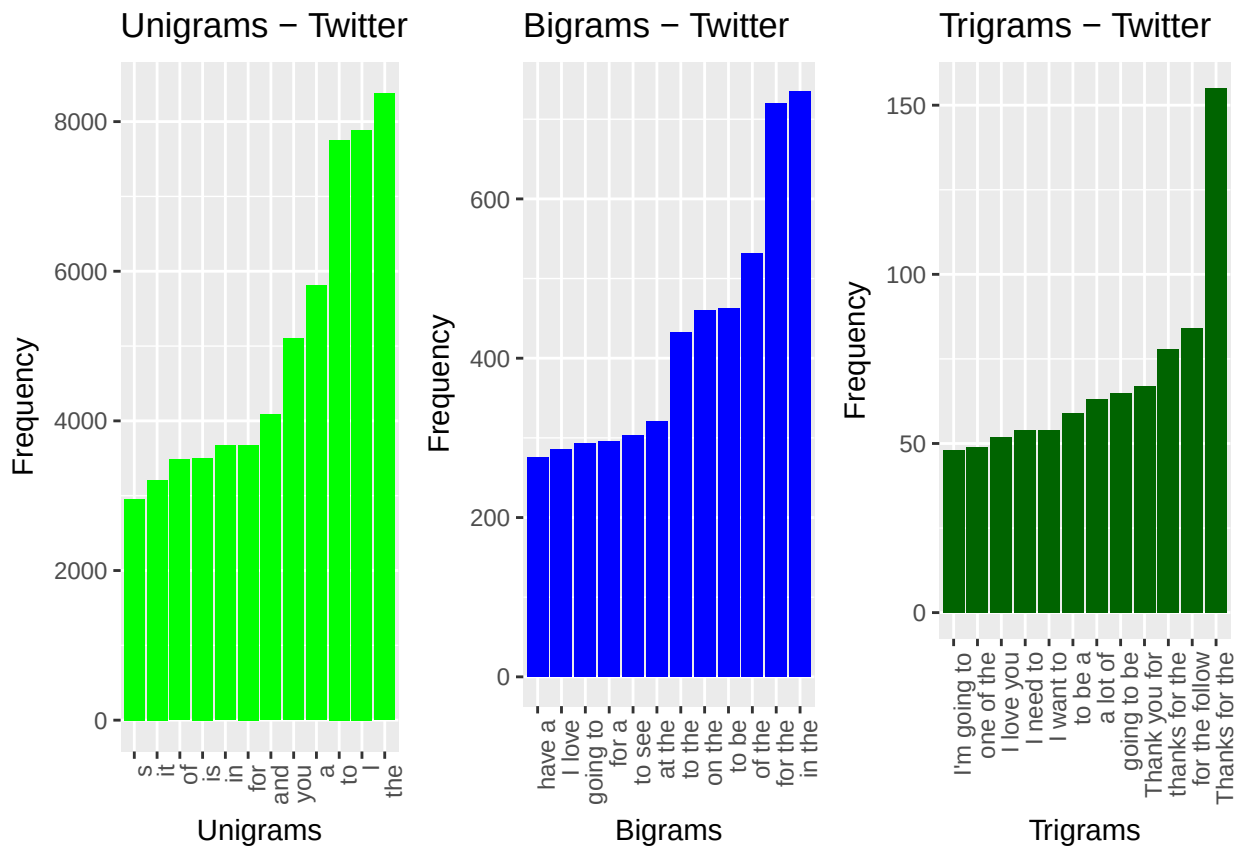
```
  Gram.2 <- Gram.2[order(Gram.2$Freq, decreasing = TRUE),]
  colnames(Gram.2) <- c("Word", "Freq")
  Gram.2 <- head(Gram.2, N)
  g2 <- ggplot(Gram.2, aes(x=reorder(Word, Freq),y=Freq)) +
          geom_bar(stat="identity", fill="blue") +
          ggtitle(paste("Bigrams", "-", subTitle)) +
          xlab("Bigrams") + ylab("Frequency") +
          theme(axis.text.x=element_text(angle=90, hjust=1))

  # Use RWeka to get trigram token
  Tokenizer3 <- RWeka::NGramTokenizer(x,
                                  Weka_control(min = 3, max = 3,
                                               delimiters = " \\r\\n\\t.,;:\"()?!"))
  Gram.3 <- data.frame(table(Tokenizer3))
  Gram.3 <- Gram.3[order(Gram.3$Freq, decreasing = TRUE),]
  colnames(Gram.3) <- c("Word", "Freq")
  Gram.3 <- head(Gram.3, N)
  g3 <- ggplot(Gram.3, aes(x=reorder(Word, Freq),y=Freq)) +
          geom_bar(stat="identity", fill="darkgreen") +
          ggtitle(paste("Trigrams", "-", subTitle)) +
          xlab("Trigrams") + ylab("Frequency") +
          theme(axis.text.x=element_text(angle=90, hjust=1))

  # Put three plots into 1 row 3 columns
  gridExtra::grid.arrange(g1, g2, g3, ncol = 3)
}

plot.Grams(x = sample_blogs, subTitle = "Blogs", N = 12)
```



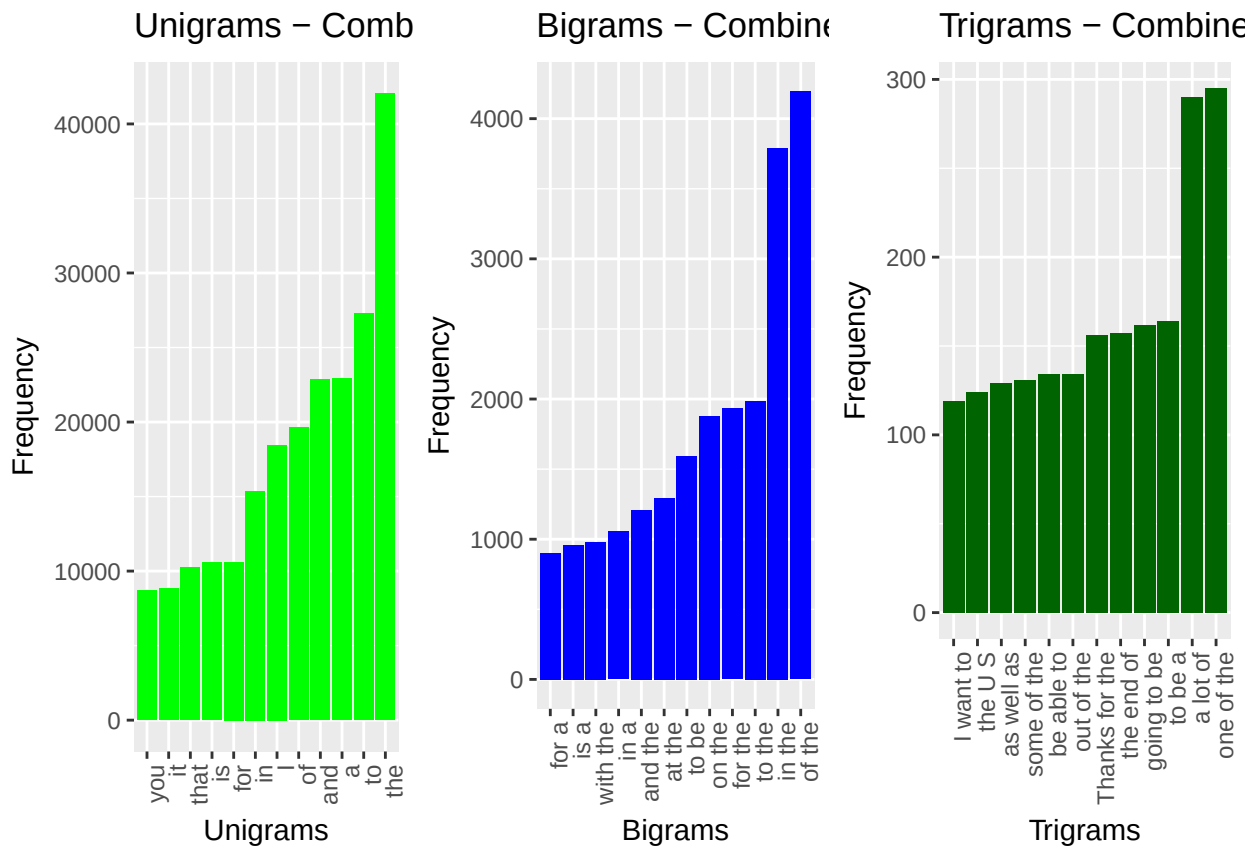

```
plot.Grams(x = sample_news, subTitle = "News", N = 12)
```




```
plot.Grams(x = sample_twitter, subTitle = "Twitter", N = 12)
```




```
plot.Grams(x = sample_dataset, subTitle = "Combined", N = 12)
```



7. Further Development Plan

After the exploratory analysis, we will build the predictive model(s) and eventually the data product.