Peer-graded Assignment: Milestone Report

Megha

02 September 2021

Introduction

This milestone report is for the Coursera Capstone Project for Data Scince Speciaization Course. The report provides a short overview of the exploratory analysis of the text data to be used together with a briefly summary of the the plans for creating the prediction algorithm and Shiny app in a way that would be understandable to a non-data scientist manager.

The motivation for this project is to:

1. Demonstrate that i have 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.

Initiatization, Data Reading and Summarization

Install and load requisite packages that are not already installed and loaded.

```r
knitr::opts_chunk$set(echo = FALSE)

list.of.packages <- c("dplyr", "ggplot2", "stringi", "knitr", "tm", "pwr", "wordcloud", "RColorBrewer", "kableExt
ra", "survey", "RSQLite", "SnowballC", "RWeka", "plotly", "devtools", "slam", "tokenizers", "quanteda", "ngram")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages, repos="http://cran.rstudio.com/")


library(dplyr)
library(ggplot2)
library(stringi)
library(tm)
library(wordcloud)
library(RColorBrewer)
library(SnowballC)
library(RWeka)
library(knitr)
library(kableExtra)
library(pwr)
library(survey)
library(RSQLite)
library(plotly)
library(devtools)
library(slam)
library(tokenizers)
library(quanteda)
library(ngram)
library(NLP)
# devtools::install_github("ropensci/plotly")
```

We start by reading the file from the souyrce https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip We then unzip the file to get the individual files in the zip file.

We note that the data is from three different text sources, i.e. blogs, news, and twitter feeds and are given in four (4) different languages: German, English (US), Finnish, and Russian. For this project, we shall use the the English (US) data sets.

We then get a summary of the English Data: file size, number of lines and number of words.

Engilsh Data Sets Summary

	File size, Mb	No. of Lines	No. of Words
Blogs	200.42	899288	37546246
News	196.28	1010242	34762395
Twitter	159.36	2360148	30093410

Data Sampling

Clearly, the data sets are very large and it is only logical that we take a non-biased sample to conduct the exploratory data analysis at this stage. We sample 2% of the number lines in each data set, then combine them into one data set that shall be used in building the corpus.

```
## [1] "The computed and combined sample size (lines) is 42695 with 1023145 words."
```

Exploratory Data Analysis

Building the Corpus

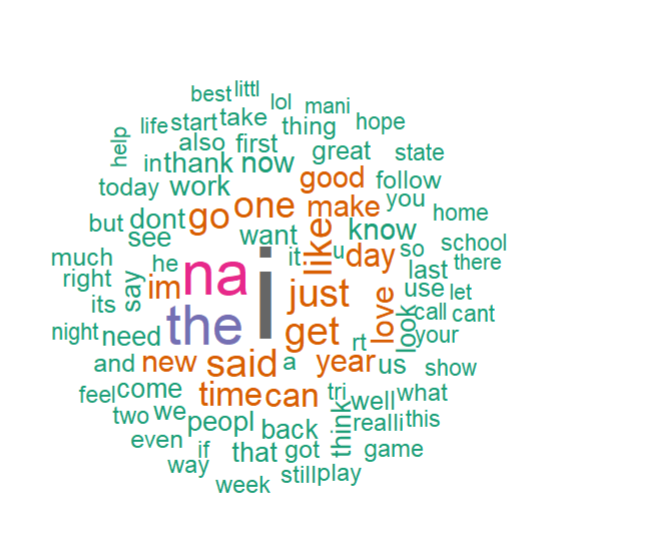
The combined sample is used to build the corpus and the following preprocessing is done to:

1. remove emoticons
2. Remove punctuation
3. Remove numbers
4. Remove stopwords
5. Stem the words (eg. "worked" will be changed to "work")
6. Convert to lower case
7. Remove white space

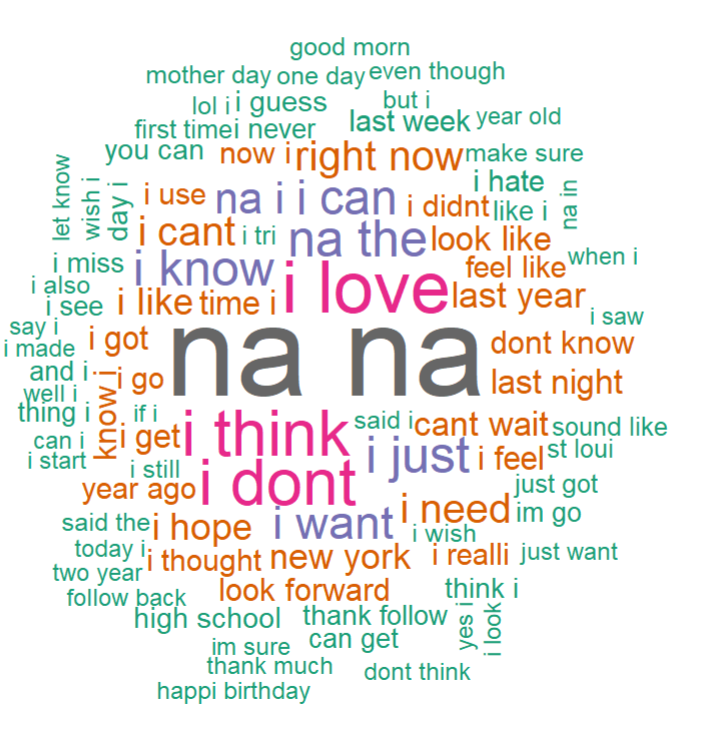
Tokenization (Identifying ngrams) and Word Clouds

We identify all combinations of adjacent words (ngrams) in the corpus. We limit ourselves to one, two and three words (unigrams, bigrams and trigrams). We then generate the word clouds.

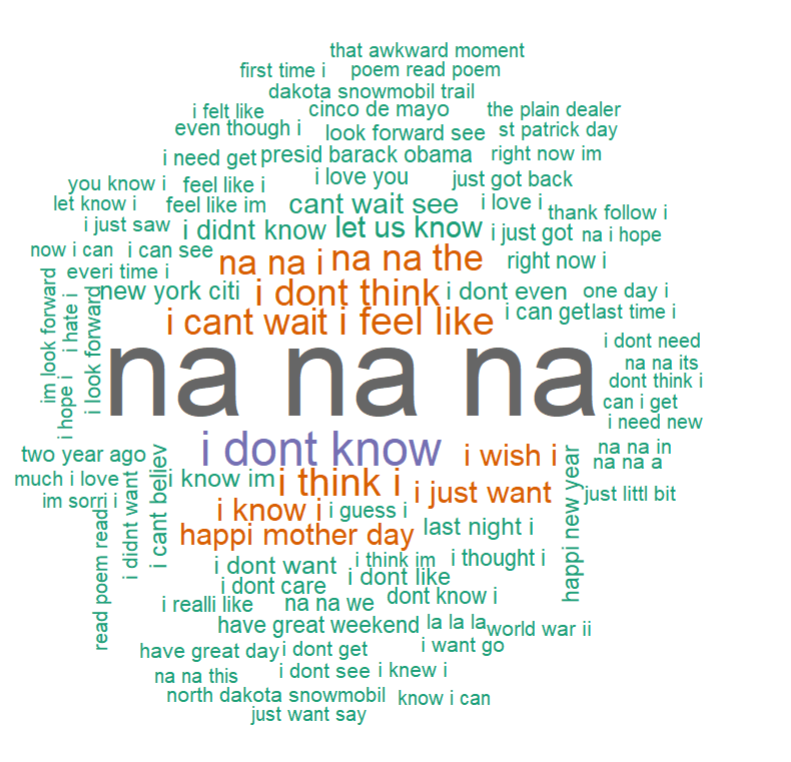
Unigram Word Cloud



Bigram Word Cloud

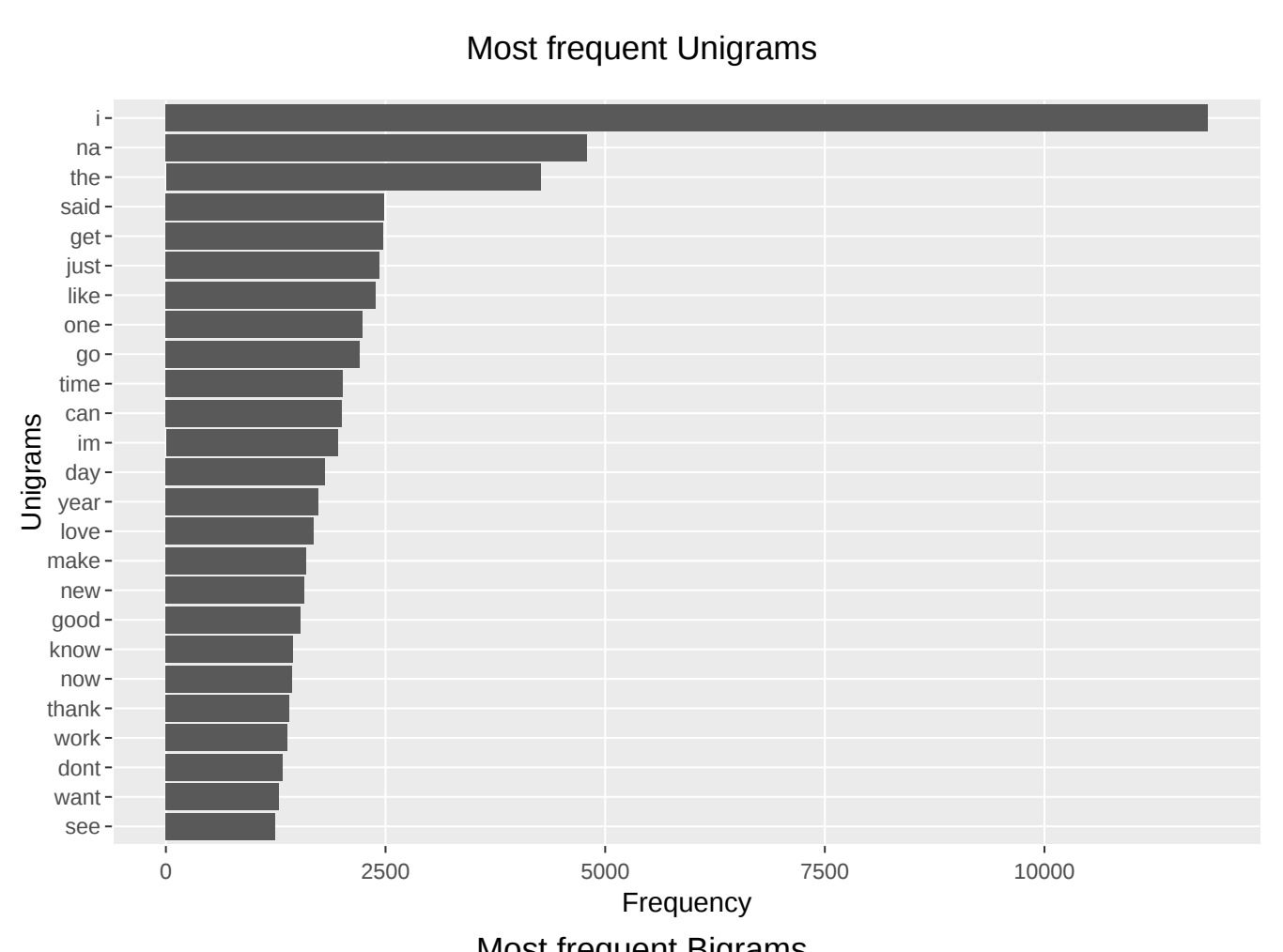


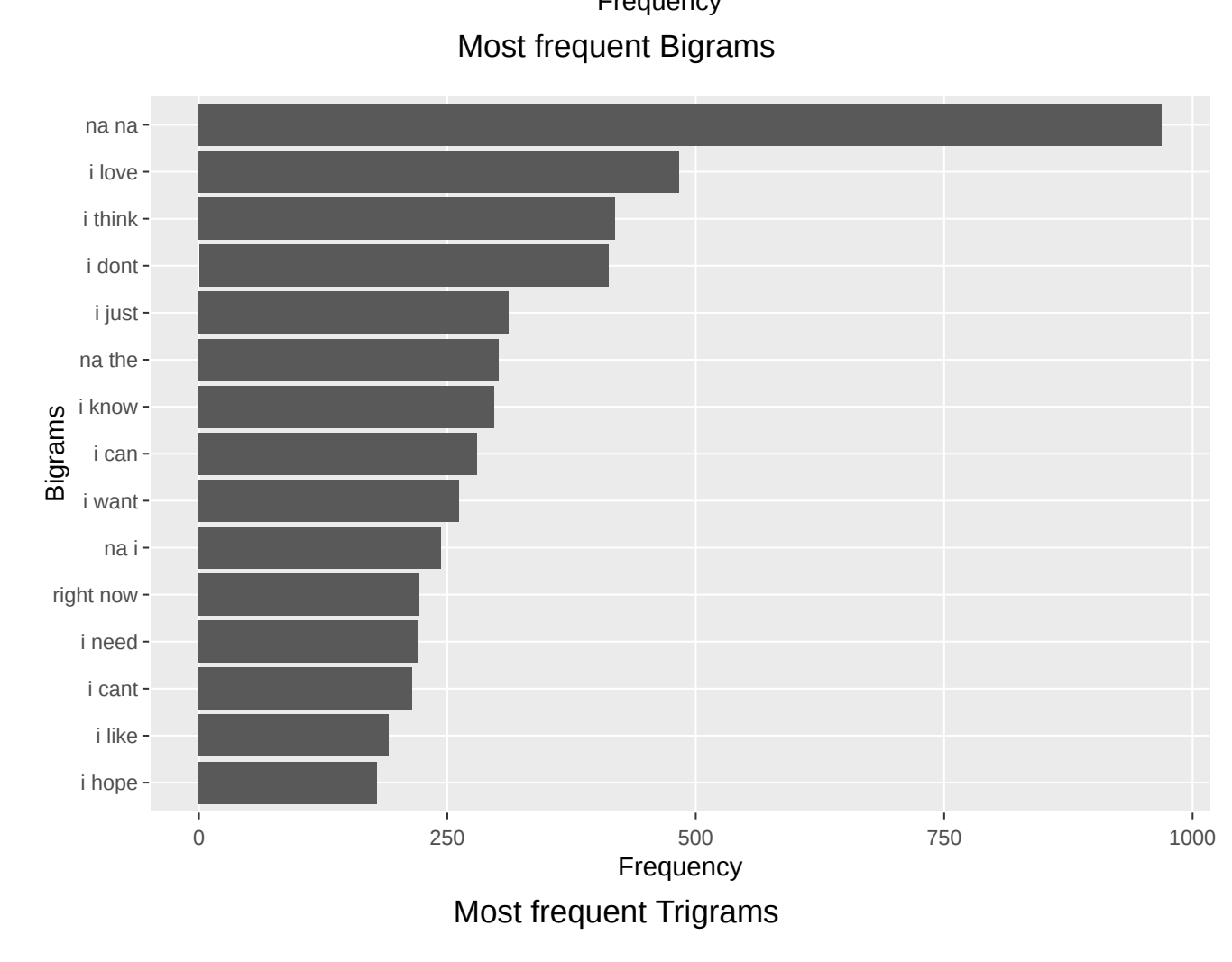
Trigram Word Cloud

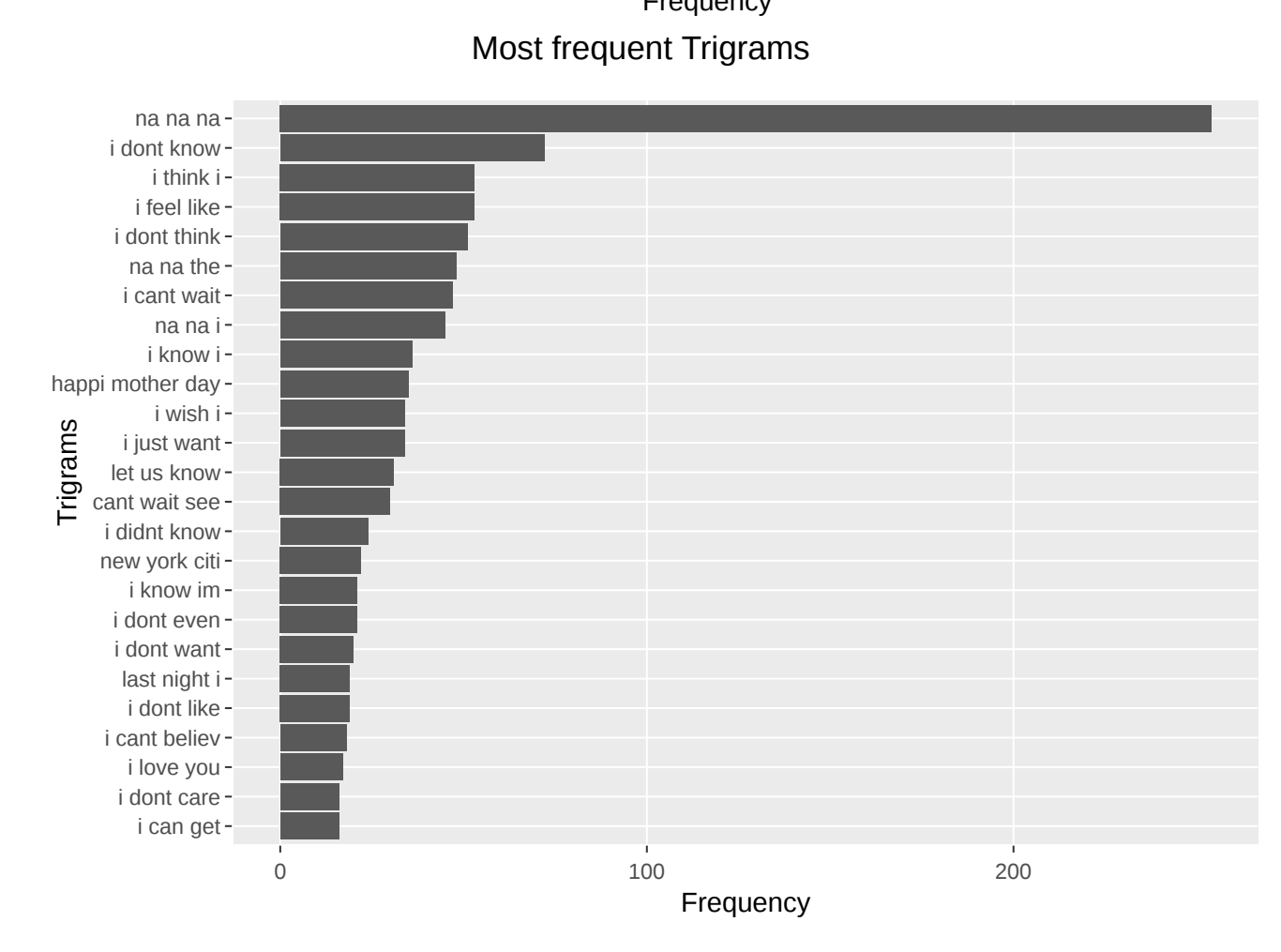


N-Gram Frequency

We conduct ngram analysis. The analysis is limited to the top twenty (20) ngrams in the corpus.







Plan for the Prediction Algorithm and Shiny app

Resulting from exploratory analysis and the n-gram analysis, it is clear that sampling produced a reasonable representation of the text given that plotting the N-Grams for two and three word combinations mostly produced grammatically correct sentence portions. This backs up the proposed plan for for the prediction algorithm to utilize a frequency look up using the n-gram models.

In developing and testing the prediction algorithm, the following shall specifically be done:

1. Additional data cleaning
2. Better sample size selection technique for the training and testing data sets
3. Build the prediction model basing on the n-gram frequancy on test data
4. Test the performance of the algorithm on test data using some of the following metrics: accuracy, precision, confusion matrix, recall and f1 score
5. Build a shiny app which is interactive and allows users to enter text for prediction of the next word(s)
6. Experiment to optimize the model to handle unseen n-grams, increasing model performance.
7. Get feedback ion the plan for creating a prediction algorithm and Shiny app

Appendices

Appendix - Setup

```r
knitr::opts_chunk$set(echo = FALSE)

list.of.packages <- c("dplyr", "ggplot2", "stringi", "knitr", "tm", "pwr", "wordcloud", "RColorBrewer", "kableExt
ra", "survey", "RSQLite", "SnowballC", "RWeka", "plotly", "devtools", "slam", "tokenizers", "quanteda", "ngram")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages, repos="http://cran.rstudio.com/")


library(dplyr)
library(ggplot2)
library(stringi)
library(tm)
library(wordcloud)
library(RColorBrewer)
library(SnowballC)
library(RWeka)
library(knitr)
library(kableExtra)
library(pwr)
library(survey)
library(RSQLite)
library(plotly)
library(devtools)
library(slam)
library(tokenizers)
library(quanteda)
library(ngram)
library(NLP)
# devtools::install_github("ropensci/plotly")
```

Appendix - Data Source

```r
setwd("P:/NITA/New/Personal/Training/Data Science/John Hopkins/Course 10 - Capstone Project/Week 2/Peer-Graded")

sourceUrl <-"https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip"
if (!file.exists("Coursera-SwiftKey.zip")){
  download.file(sourceUrl, destfile = "Coursera-SwiftKey.zip")
  unzip("Coursera-SwiftKey.zip")
}
```

Appendix - Read Data

```r
file.list = c("final/en_US/en_US.blogs.txt", "final/en_US/en_US.news.txt", "final/en_US/en_US.twitter.txt")
text <- list(blogs = "", news = "", twitter = "")

en_US.summary <- matrix(0, nrow = 3, ncol = 3, dimnames = list(c("Blogs", "News", "Twitter"),c("File size, Mb",
"No. of Lines", "No. of Words")))
for (i in 1:3) {
  con <- file(file.list[i], "rb")
  text[[i]] <- readLines(con, encoding = "UTF-8",skipNul = TRUE)
  close(con)
  en_US.summary[i,1] <- round(file.info(file.list[i])$size / 1024^2, 2)
  en_US.summary[i,2] <- length(text[[i]])
  en_US.summary[i,3] <- sum(stri_count_words(text[[i]]))
}

kable(en_US.summary, "html", caption = "Engilsh Data Sets Summary") %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

Appendix - Sample Data

```r
set.seed(12345)

## Get population sizes (lines)
blogs_pop <- length(text$blogs)
news_pop <- length(text$news)
twitter_pop <- length(text$twitter)

## Create the sample
blogs_sample <- sample(text$blogs, 0.01*length(text$blogs))
news_sample <- sample(text$news, 0.01*length(text$news))
twitter_sample <- sample(text$twitter, 0.01*length(text$twitter))
sample_corpus_data <- c(blogs_sample, news_sample, twitter_sample)

sumLines <- length(sample_corpus_data)
sumWords <- sum(stri_count_words(sample_corpus_data))
paste("The computed and combined sample size (lines) is", sumLines, "with", sumWords, "words.")

## Write the file to disc
writeLines(sample_corpus_data, "sample_corpus_data.txt")
```

Appendix - Clean Corpus

```r
sample_corpus_data <- iconv(sample_corpus_data, 'UTF-8', 'ASCII')
corpusNew <- Corpus(VectorSource(as.data.frame(sample_corpus_data, na.rm=TRUE, stringsAsFactors = FALSE)))

## Pre-process the corpus
corpusNew <- corpusNew %>%
  tm_map(removePunctuation) %>% # remove punctuation
  tm_map(removeNumbers) %>% # Remove numbers
  tm_map(removeWords, stopwords("english")) %>% # remove stopwords
  tm_map(stemDocument) %>% # Stem words
  tm_map(tolower) %>%  # convert to lower case
  tm_map(stripWhitespace)%>%    # Strip white spaces
  tm_map(PlainTextDocument) # Convert to plain text
```

Appendix - Tokenise and Word Cloud

```r
knitr::opts_chunk$set(echo = TRUE)

unigram <- NGramTokenizer(corpusNew, Weka_control(min = 1, max = 1))
bigram <- NGramTokenizer(corpusNew, Weka_control(min = 2, max = 2))
trigram <- NGramTokenizer(corpusNew, Weka_control(min = 3, max = 3))

## Unigram
unigram.df <- data.frame(table(unigram))
unigram.df <- unigram.df[order(unigram.df$Freq, decreasing = TRUE),]

## Bigram
bigram.df <- data.frame(table(bigram))
bigram.df <- bigram.df[order(bigram.df$Freq, decreasing = TRUE),]


## Trigram
trigram.df <- data.frame(table(trigram))
trigram.df <- trigram.df[order(trigram.df$Freq, decreasing = TRUE),]

## Generate Word Clouds
col.palette <- brewer.pal(8, "Dark2") # Colour Palette to be used for all Word Clouds

# Unigram Word Cloud
layout(matrix(c(1, 2), nrow=2), heights=c(1, 4))
par(mar=rep(0, 4))

plot.new()
text(x=0.5, y=0.5, "Unigram Word Cloud")
wordcloud(unigram.df$unigram, unigram.df$Freq, max.words=90, random.order = F, colors=col.palette, main="Title")

# Bigram Word Cloud
layout(matrix(c(1, 2), nrow=2), heights=c(1, 4))
par(mar=rep(0, 4))

plot.new()
text(x=0.5, y=0.5, "Bigram Word Cloud")

wordcloud(bigram.df$bigram, bigram.df$Freq, max.words=90, random.order = F, colors=col.palette, main="Title")

## Trigram Word Cloud
layout(matrix(c(1, 2), nrow=2), heights=c(1, 4))
par(mar=rep(0, 4))

plot.new()
text(x=0.5, y=0.5, "Trigram Word Cloud")
wordcloud(trigram.df$trigram, trigram.df$Freq, max.words=90, random.order = F, colors=col.palette, main="Title")
```

Appendix - Tokenise and Word Cloud

```r
#set up the plot area
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))

## Unigram Frequency
ggUni <- ggplot(head(unigram.df,25), aes(reorder(unigram,Freq), Freq)) +
  geom_bar(stat="identity") + coord_flip() +
  xlab("Unigrams") + ylab("Frequency") +
  ggtitle("Most frequent Unigrams")

ggplotly(ggUni)

# Bigram Frequency
ggBi <- ggplot(head(bigram.df,15), aes(reorder(bigram,Freq), Freq)) +
  geom_bar(stat="identity") + coord_flip() +
  xlab("Bigrams") + ylab("Frequency") +
  ggtitle("Most frequent Bigrams")

ggplotly(ggBi)

## Tigram Frequency
ggTri <- ggplot(head(trigram.df,25), aes(reorder(trigram,Freq), Freq)) +
  geom_bar(stat="identity") + coord_flip() +
  xlab("Trigrams") + ylab("Frequency") +
  ggtitle("Most frequent Trigrams")

ggplotly(ggTri)
```