RPubs by RStudio Sign in Register

Peer-graded Assignment: Milestone Report

Sushant Tripathi

August 05, 2020

Synopsis

This report provides a short overview of the exploratory analysis of the text data to be used for the Capstone project for the Data Science Specialization along with a description of plans for the word prediction algorithm.

As outlined on the Capstone Project website (https://www.coursera.org/learn/data-science-project/peer/BRX21/milestone-report), the motivation for this project is to:

- Demonstrate that the student have downloaded the data and have successfully loaded it in;
- Create a basic report of summary statistics about the data sets;
- Report any interesting findings that you amassed so far;
- Get feedback on your plans for creating a prediction algorithm and Shiny app.

Data loading and analysis

0. Install the R packages necessary for running the analysis (if not already installed).

```
list.of.packages <- c("stringi", "tm", "wordcloud", "RColorBrewer")
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)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(stringi)
```

1. Load the data

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

The data consist of text from 3 different sources: blogs, news, and twitter feeds and are provided in 4 different languages: German, English (US), Finnish, and Russian. For the remainder of this project, we will use only the the English (US) data sets.

2. Summary of the English (US) data

```
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 = "")

data.summary <- matrix(0, nrow = 3, ncol = 3, dimnames = list(c("blogs", "news", "twitter"),c("file size, Mb", "lines", "wor
ds")))
for (i in 1:3) {
  con <- file(file.list[i], "rb")
  text[[i]] <- readLines(con, encoding = "UTF-8",skipNul = TRUE)
  close(con)
  data.summary[i,1] <- round(file.info(file.list[i])$size / 1024^2, 2)
  data.summary[i,2] <- length(text[[i]])
  data.summary[i,3] <- sum(stri_count_words(text[[i]]))
}
```

The data is summarized in the table below.

```
library(knitr)
kable(data.summary)
```

	file size, Mb	lines	words
blogs	200.42	899288	37546246
news	196.28	1010242	34762395
twitter	159.36	2360148	30093410

These datasets are rather large, and since the goal is to provide a proof of concept for the data analysis, for the remainder of the report we will sample a smaller fraction of the data (1 %) to perform the analysis. The three parts will be combine into a single file and used to generate the corpus.

```
set.seed(123)
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))
sampled_data <- c(blogs_sample, news_sample, twitter_sample)
sum <- sum(stri_count_words(sampled_data))
```

The new data set consists of (1024351) words.

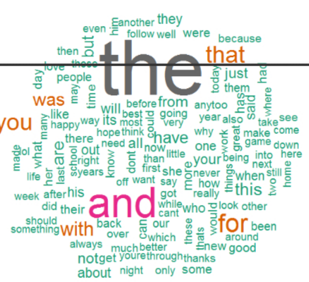
Build the corpus

```
library(tm)
library(wordcloud)
library(RColorBrewer)
# remove emoticons
sampled_data <- iconv(sampled_data, 'UTF-8', 'ASCII')
corpus <- Corpus(VectorSource(as.data.frame(sampled_data, stringsAsFactors = FALSE)))
corpus <- corpus %>%
  tm_map(tolower) %>%
  tm_map(PlainTextDocument) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(stripWhitespace)
```

```
term.doc.matrix <- TermDocumentMatrix(corpus)
term.doc.matrix <- as.matrix(term.doc.matrix)
word.freqs <- sort(rowSums(term.doc.matrix), decreasing=TRUE)
dm <- data.frame(word=names(word.freqs), freq=word.freqs)
```

Word cloud plot of the most common words in the corpus

```
wordcloud(dm$word, dm$freq, min.freq= 500, random.order=TRUE, rot.per=.25, colors=brewer.pal(8, "Dark2"))
```



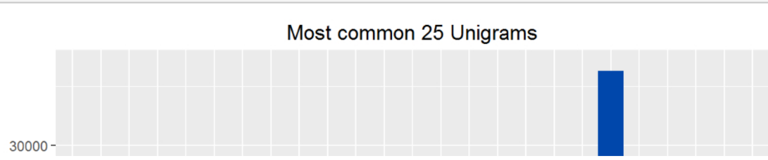
Tokenization

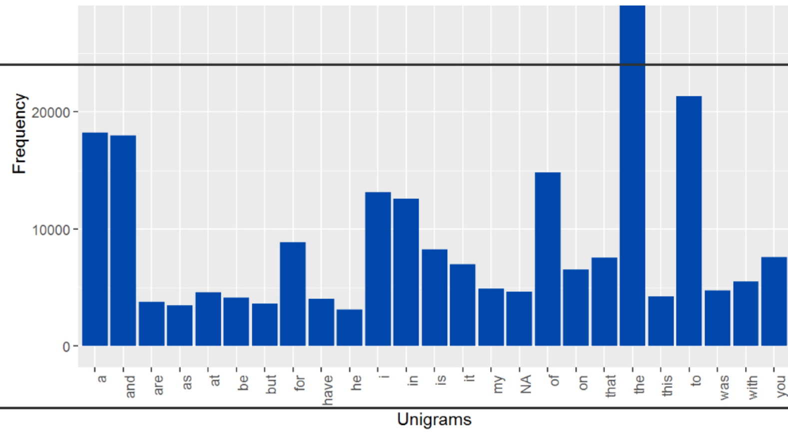
```
library(RWeka)
unigram <- NGramTokenizer(corpus, Weka_control(min = 1, max = 1))
bigram <- NGramTokenizer(corpus, Weka_control(min = 2, max = 2)) #, delimiters = " \\r\\n\\t.,;:\"()?!"))
trigram <- NGramTokenizer(corpus, Weka_control(min = 3, max = 3)) #, delimiters = " \\r\\n\\t.,;:\"()?!"))
```

Unigram frequency distribution

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

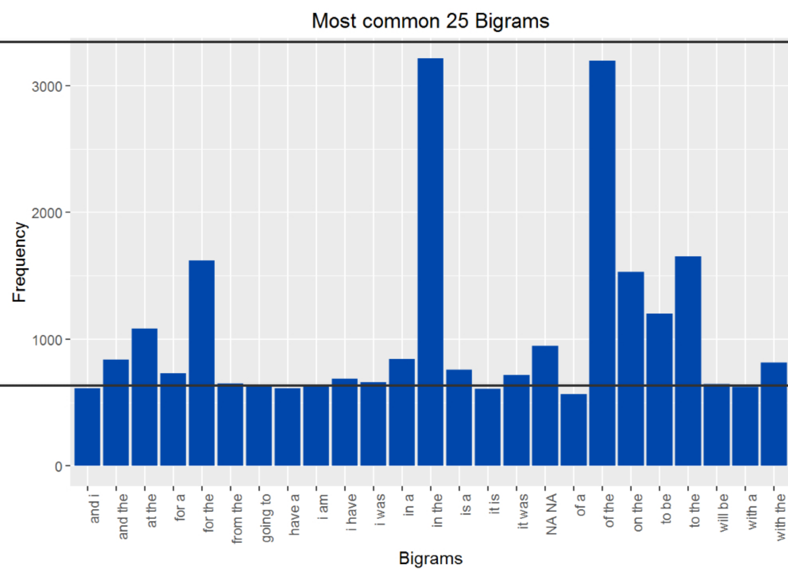
```
ggplot(unigram.df[1:25,], aes(x=unigram, y=Freq)) +
  geom_bar(stat="Identity", fill="#0047AB")+
  xlab("Unigrams") + ylab("Frequency")+
  ggtitle("Most common 25 Unigrams") +
  theme(axis.text.x=element_text(angle=90, hjust=1))
```





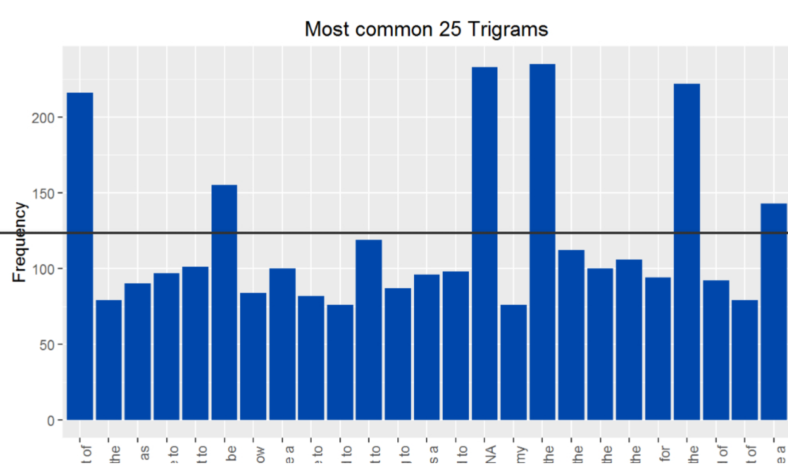
Bigram frequency distribution

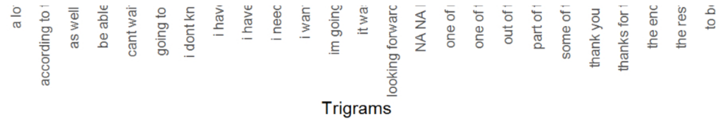
```
bigram.df <- data.frame(table(bigram))
bigram.df <- bigram.df[order(bigram.df$Freq, decreasing = TRUE),]

ggplot(bigram.df[1:25,], aes(x=bigram, y=Freq)) +
  geom_bar(stat="Identity", fill="#0047AB")+
  xlab("Bigrams") + ylab("Frequency")+
  ggtitle("Most common 25 Bigrams") +
  theme(axis.text.x=element_text(angle=90, hjust=1))
```



Trigram frequency distribution

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

ggplot(trigram.df[1:25,], aes(x=trigram, y=Freq)) +
  geom_bar(stat="Identity", fill="#0047AB")+
  xlab("Trigrams") + ylab("Frequency")+
  ggtitle("Most common 25 Trigrams") +
  theme(axis.text.x=element_text(angle=90, hjust=1))
```





Summary

1. the data sets are pretty big and processing them requires time and computing resources;
2. most of the top ranking n-grams contains English stop words
3. using the n-grams we can conceive a crude algorithm to suggest the next words in a text editor; For example, the probability of an untyped word can be estimated from the frequencies in the corpus of the n-grams containing that word in the last position conditioned on the presence the last typed word(s) as the first *n - 1* words in the n-gram. One can use a weighted sum of frequencies, with the weights calculated using machine learning.
4. use a pre-built R algorithm, like one based on Hidden Markov model and the n-grams calculated from the data sets provided in this class.