Predicting Words Milestone 1

WritPen

12/29/2020

Project Description

This project examines the frequencies of common words in sequence, to begin to build an algorithm that will predict or suggest the next word the writer intends to type. This project uses data from SwiftKey (swiftkey.com) from US twitter, blogs, and news sources.

This project endeavors to be reproducible (Peng, 2011, 2016), and follows the John Hopkins Univeristy Biostatistics Capstone project.

Packages needed for this project: - knitr - tm - quanteda - stringr - ggplot2 - caret

Summary Statistics and Loading in the data

The data files are large. The files also need to be cleaned prior to developing a model.

```
con <- file("en_US.twitter.txt", "r")
allLines <- suppressWarnings(readLines(con))
twitter_len <- length(allLines)
writeLines(paste("twitter file length:",twitter_len))
```

```
## twitter file length: 2360148
```

```
longest_line <- 0
for (idx in 1:twitter_len) {
  linelength <- nchar(allLines[idx])
  if (linelength > longest_line)
  {
    longest_line <- linelength
  }
}
writeLines(paste("longest twitter line :",longest_line,"characters"))
```

```
## longest twitter line : 213 characters
```

```
close(con) # close connection to twitter document to save on RAM

con <- file("en_US.blogs.txt", "r")
allLines <- suppressWarnings(readLines(con))
blogs_len <- length(allLines)
writeLines(paste("blogs file length:",blogs_len))
```

```
## blogs file length: 899288

longest_line <- 0
for (idx in 1:blogs_len) {
        linelength <- nchar(allLines[idx])
        if (linelength > longest_line)
        {
                longest_line <- linelength
        }
}
writeLines(paste("longest blogs line:",longest_line,"characters"))

## longest blogs line: 40835 characters

close(con) # close connection to blogs document to save on RAM

con <- file("en_US.news.txt", "r")
allLines <- suppressWarnings(readLines(con))
news_len <- length(allLines)
writeLines(paste("news file length:",news_len))

## news file length: 77259

longest_line <- 0
for (idx in 1:news_len) {
        linelength <- nchar(allLines[idx])
        if (linelength > longest_line)
        {
                longest_line <- linelength
        }
}
writeLines(paste("longest news line:",longest_line,"characters"))

## longest news line: 5760 characters

close(con) # close connection to news document to save on RAM
```

Tokenization and Transformation

To minimize the amount of RAM utilized in processing, the model is built using a sample of the full corpus. This also doubles as creating a training set, where further samples can be pulled to test the effectiveness of the algorithm on additional samples in the corpus. Each sample, as pulled, will need to be preprossed/cleaned prior to running the final algorithms.

The data will be easier to run in analysis if the words are clearly written following rules for plain English (https://en.wikipedia.org/wiki/Plain_English). The following code: 1. removes special letters (e.g. those with accents), 2. makes all text lines lowercase, 3. removes non-alphabetic characters, 4. deletes extra spaces 5. replaces common contractions with full words

```
## Package version: 2.1.2

## Parallel computing: 2 of 4 threads used.

## See https://quanteda.io for tutorials and examples.

##
## Attaching package: 'quanteda'

## The following object is masked from 'package:utils':
##
##     View
```

```r
fix.syntax <- function(text, wordCount = 350000) {
                # Remove special-characters words (e.g. 'f---', 'f***',
'f#@%')
                text <- gsub(text, "\\b[a-z]+[#$%&()*+/:<=>@[\\^_`|~-]+")
                return(text)
        }
        convertText <- function (text) {
                # Replace common contractions
                # auxiliary + negation
                text <- ConvertTo(text, "\\bcan't\\b", "cannot")
                text <- ConvertTo(text, "\\bwon't\\b", "will not")
                text <- ConvertTo(text, "\\bshan't\\b", "shall not")
                text <- ConvertTo(text, "\\bain't\\b", "am not")
                # verb + negation (isn't, aren't, wasn't, etc.)
                text <- ConvertTo(text, "n't\\b", " not")
                # miscellaneous forms
                text <- ConvertTo(text, "\\blet's\\b", "let us")
                text <- ConvertTo(text, "\\bc'mon\\b", "come on")
                text <- ConvertTo(text, "'n\\b", " and")
                # pronoun + verb
                text <- ConvertTo(text, "\\bi'm\\b", "i am")
                text <- ConvertTo(text, "'re\\b", " are")
                text <- ConvertTo(text, "'s\\b", " is")
                text <- ConvertTo(text, "'d\\b", " would")
                text <- ConvertTo(text, "'ll\\b", " will")
                text <- ConvertTo(text, "'ve\\b", " have")
                # Replace contractions with full words
                text <- ConvertTo(text, "\\bb\\b", "be")
                text <- ConvertTo(text, "\\bc\\b", "see")
                text <- ConvertTo(text, "\\bm\\b", "am")
                text <- ConvertTo(text, "\\bn\\b", "and")
                text <- ConvertTo(text, "\\bo\\b", "oh")
                text <- ConvertTo(text, "\\br\\b", "are")
                text <- ConvertTo(text, "\\bu\\b", "you")
                text <- ConvertTo(text, "\\by\\b", "why")
                text <- ConvertTo(text, "\\b1\\b", "one")
                text <- ConvertTo(text, "\\b2\\b", "to")
                text <- ConvertTo(text, "\\b4\\b", "for")
                text <- ConvertTo(text, "\\b8\\b", "ate")
                text <- ConvertTo(text, "\\b2b\\b", "to be")
                text <- ConvertTo(text, "\\b2day\\b", "today")
                text <- ConvertTo(text, "\\b2moro\\b", "tomorrow")
                text <- ConvertTo(text, "\\b2morow\\b", "tomorrow")
                text <- ConvertTo(text, "\\b2nite\\b", "tonight")
                text <- ConvertTo(text, "\\bl8r\\b", "later")
                text <- ConvertTo(text, "\\b4vr\\b", "forever")
                text <- ConvertTo(text, "\\b4eva\\b", "forever")
                text <- ConvertTo(text, "\\b4ever\\b", "forever")
                text <- ConvertTo(text, "\\bb4\\b", "before")
                text <- ConvertTo(text, "\\bcu\\b", "see you")
                text <- ConvertTo(text, "\\bcuz\\b", "because")
                text <- ConvertTo(text, "\\btnx\\b", "thanks")
                text <- ConvertTo(text, "\\btks\\b", "thanks")
                text <- ConvertTo(text, "\\bthks\\b", "thanks")
                text <- ConvertTo(text, "\\bthanx\\b", "thanks")
                text <- ConvertTo(text, "\\bu2\\b", "you too")
                text <- ConvertTo(text, "\\bur\\b", "your")
                text <- ConvertTo(text, "\\bgr8\\b", "great")
                return(text)
        }

        cleanText <- function(text) {
                # remove punctuation and other special characters
                text <- removePunctuation(text,
                                          preserve_intra_word_dashes = TRUE)

                # gsub hashtags
                text <- gsub("#", pattern, "#\\w+")

                # gsub "rt"
                text <- gsub("rt", pattern, "\\brt\\b")

                # gsub email addresses
                text <-
                        gsub("@", pattern, "( \\S+\\@\\S+\\..{1,3}(\\s)? )")

                # gsub things starting with http
                text <- gsub("http", pattern, "http[^[:space:]]*")

                # gsub slashes
                text <- ConvertTo(text, "/|@|\\|", " ")

                return(text)

        # Truncate text to the last wordCount words
        text <- substring(text, 350000)
```

```
        # Convert to lowercase
        text <- tolower(text)

        # Convert contractions
        text <- convertText(text)

        # Truncate text to the last wordCount words
        text <- substring(text, 350000)

        # remove profanity
        text <- removeProfanity(text)

        # Clean
        text <- cleanText(text)

        # remove numbers
        text <- gsub(text, "[^a-z ]")

        # Strip out extra whitespace
        text <- str_trim(stripWhitespace(text), side = 'both')

        if (length(text) == 0)
                return("")
        else
                return(text)
}

write.fix.file <- function(corpus_sample, new_file){
   my_text <- read_lines(corpus)
   new_text <- fix.letters(my_text)
   clean_text <- fix.syntax(new_text)

   write_lines(new_text, file=new_file)
}
new_file <- 'corpus_fix.txt'
corpus_fix <- paste0(corpus_sample, new_file)
# if (!file.exists(dir, new_file[1]))
str(corpus_fix)

##  chr [1:77259] "He wasn't home alone, apparently.corpus_fix.txt" ...
```

Generating the togens with quanteda

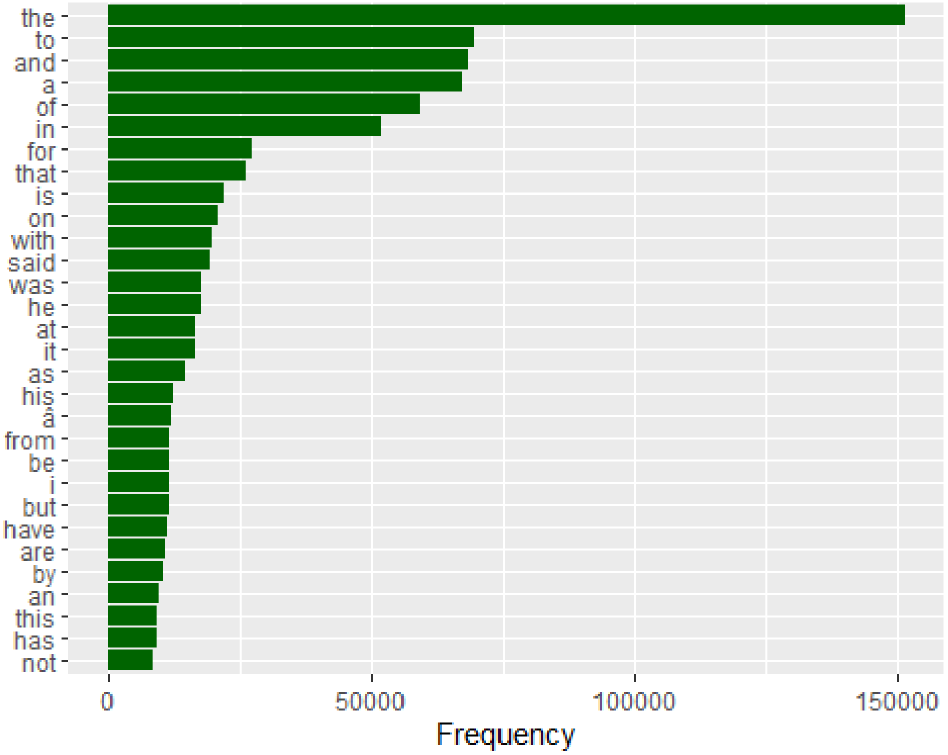
In this project a token is a word, a string with an identified meaning.

```
library(quanteda)
my_tokens <- tokens(corpus_sample, what='word', remove_punct=TRUE,
split_hyphens=TRUE, remove_numbers=TRUE, remove_symbols=TRUE)
```

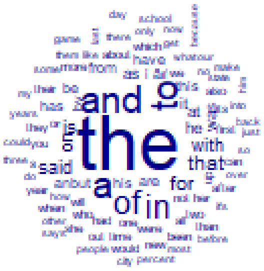
Exploratory ngrams

This section explores word counts. This analysis counts single, pairs (bigrams), three somes (trigrams). Then the plot of the frequencies is plotted. This illustrates the balance between accuracy and computer processing needs. The small dataset allows for fairly quick accuracy quickly within the sample, but with out of sample error potentially increasing as the sample's use grows.

```
library(ggplot2)
tokens_ngram_1 <- dfm(my_tokens)
text_freq_1 <- textstat_frequency(tokens_ngram_1, n=30)
ggplot(text_freq_1, aes(x=reorder(feature, frequency), y=frequency)) +
   geom_bar(stat = "identity", fill = I("dark green")) + coord_flip() +
   labs(x = NULL, y = "Frequency")
```

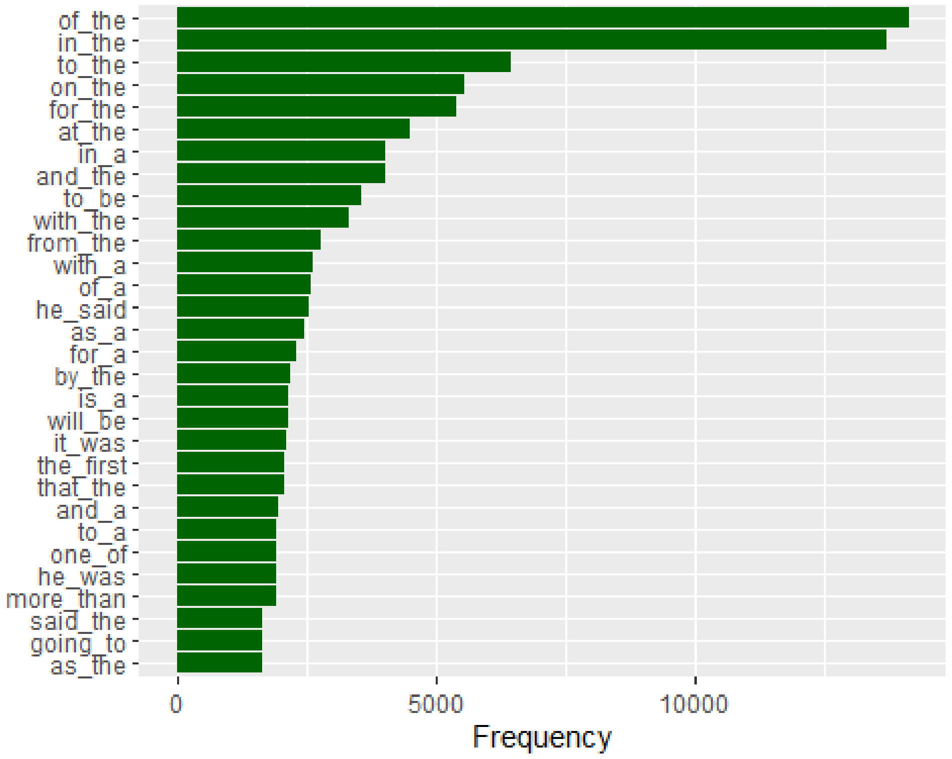


```
set.seed(40)
textplot_wordcloud(tokens_ngram_1, max_words=100)
```

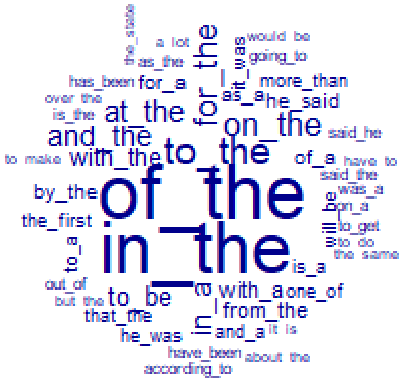


# Bigrams frequency

```
tokens_ngram_2 <- dfm(tokens_ngrams(my_tokens, n=2))
text_freq_2 <- textstat_frequency(tokens_ngram_2, n=30)
ggplot(text_freq_2, aes(x=reorder(feature, frequency), y=frequency)) +
   geom_bar(stat = "identity", fill = I("dark green")) + coord_flip() +
   labs(x = NULL, y = "Frequency")
```

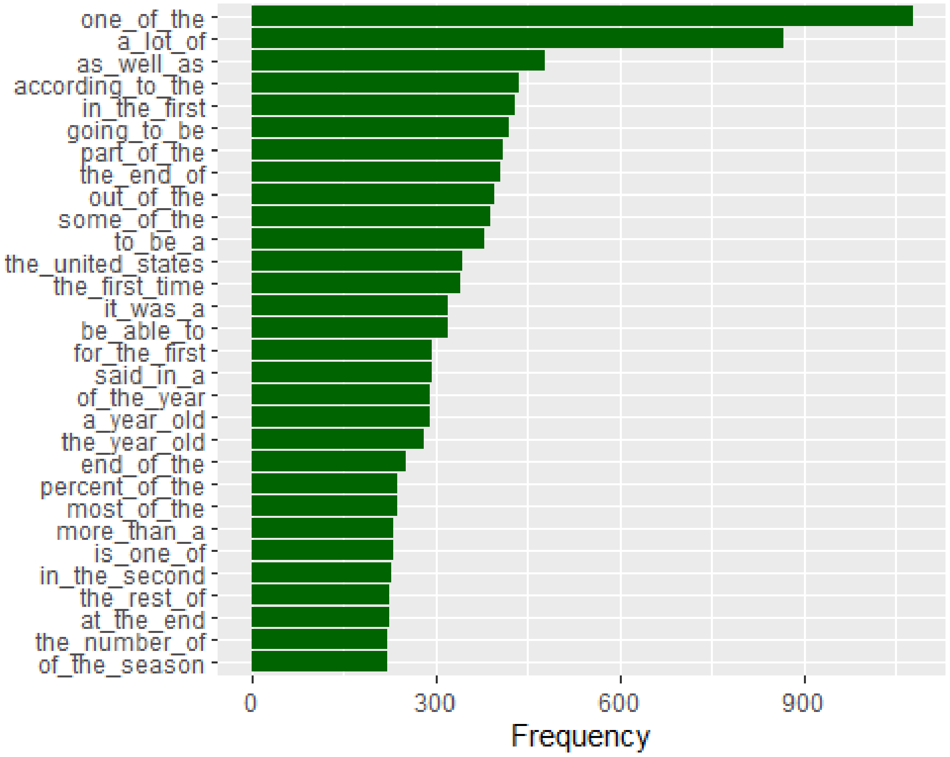


```
set.seed(45)
textplot_wordcloud(tokens_ngram_2, max_words=50)
```

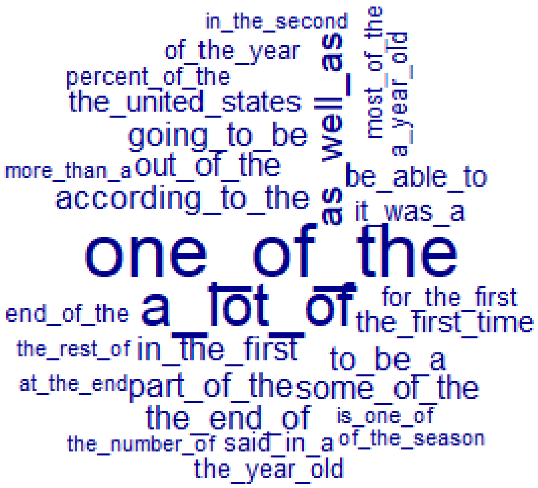


Trigrams Frequency

```
tokens_ngram_3 <- dfm(tokens_ngrams(my_tokens, n=3))
text_freq_3 <- textstat_frequency(tokens_ngram_3, n=30)
ggplot(text_freq_3, aes(x=reorder(feature, frequency), y=frequency)) +
   geom_bar(stat = "identity", fill = I("dark green")) + coord_flip() +
   labs(x = NULL, y = "Frequency")
```

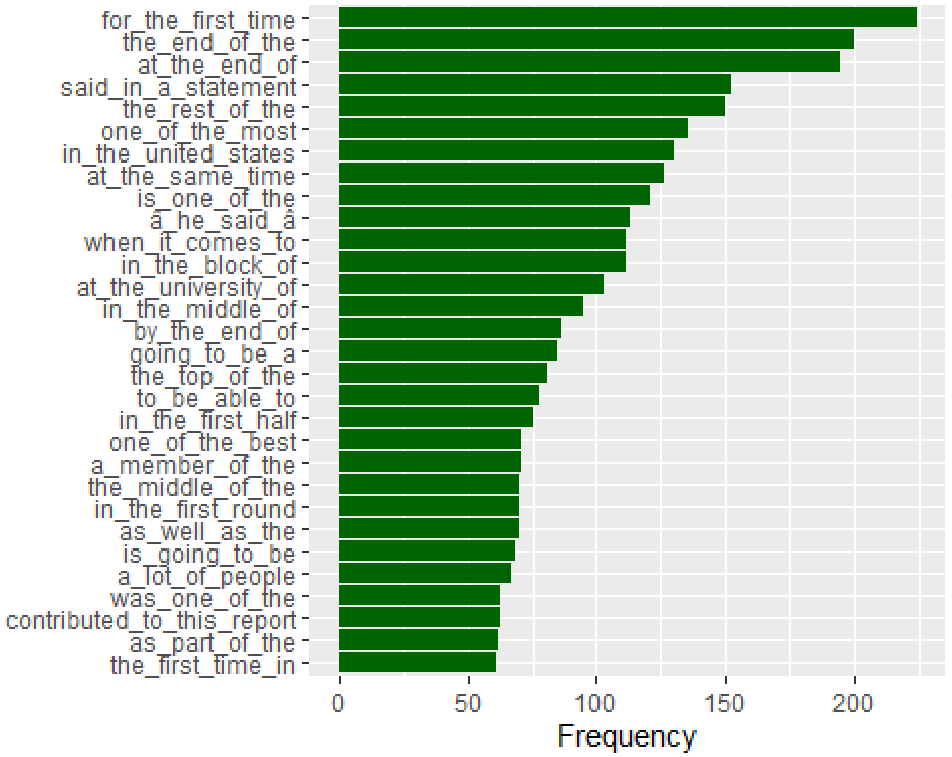


```
set.seed(43)
textplot_wordcloud(tokens_ngram_3, max_words=30)
```



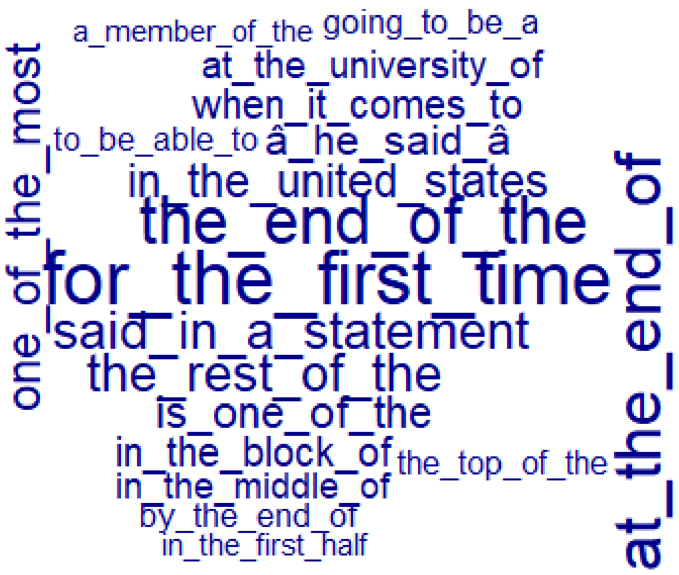
Quadgrams Frequency

```
tokens_ngram_4 <- dfm(tokens_ngrams(my_tokens, n=4))
text_freq_4 <- textstat_frequency(tokens_ngram_4, n=30)
ggplot(text_freq_4, aes(x=reorder(feature, frequency), y=frequency)) +
   geom_bar(stat = "identity", fill = I("dark green")) + coord_flip() +
   labs(x = NULL, y = "Frequency")
```



```
set.seed(86)
textplot_wordcloud(tokens_ngram_4, max_words=20)

## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## at_the_same_time could not be fit on page. It will not be plotted.
```



Considerations for future exploration

1. We might consider the limitations of the Shiny server for processing, and limit the amount of text that feeds into the process for a timely, efficient set of algorithms.
2. Further cleaning of invalid words, such as truncated text vernacular.
3. Further transformation of words where the special characters and contractions complicate the model.