Modeling and Prediction Algorithm

Ashwin Sai Murali Neelakandan

2025-01-29

Setting up the Environment

Due to large size of the data, I have downloaded the data from the above link, unzipped it and stored it locally in the folder Coursera-SwiftKey to save time downloading and unzipping the data.

```
# Loading the necessary packages

library(knitr)
library(quanteda)
library(tm)
library(dplyr)
library(stringi)
library(stringr)
library(data.table)



# Setting the working directory

setwd("C:/Users/ashwi/OneDrive/Documents/Coursera/Data Science Capstone/Project Data")
```

Document Setup

Reading the Blogs, News, and Twitter Files

Sampling the Corpus Data

Due to large sizes of the data files, to improve algorithm efficiency and reduce computation time, I will be taking a random sampling of from the three files

```
# Setting the seed
set.seed(49284)
# Assigning sample size for each file
s_sizeblog <- 0.04 * length(blogfile)
s_sizetwitter <- 0.04 * length(twitterfile)
s_sizenews <- 0.04 * length(newsfile)

# Sampling the data from each file

sam_blog <- sample(blogfile, s_sizeblog, replace = FALSE)
sam_twitter <- sample(twitterfile, s_sizetwitter, replace = FALSE)
```

```
sam_news <- sample(newsfile, s_sizenews, replace = FALSE)

# Converting all non English characters that may be present in the sample data
sam_blog <- iconv(sam_blog, "latin1", "ASCII", sub = "")
sam_twitter <- iconv(sam_twitter, "latin1", "ASCII", sub = "")
sam_news <- iconv(sam_news, "latin1", "ASCII", sub = "")

# Taking the 25th and 75th quantile of each sample data to help with removing outliers

first_quantB <- quantile(nchar(sam_blog), 0.25)
third_quantB <- quantile(nchar(sam_blog), 0.75)
first_quantT <- quantile(nchar(sam_twitter), 0.25)
third_quantT <- quantile(nchar(sam_twitter), 0.75)
first_quantN <- quantile(nchar(sam_news), 0.25)
third_quantN <- quantile(nchar(sam_news), 0.75)

# Removing outliers from the sample files
sam_blog <- sam_blog[nchar(sam_blog)> first_quantB]
sam_blog <- sam_blog[nchar(sam_blog)< third_quantB]
sam_twitter <- sam_twitter[nchar(sam_twitter)> first_quantT]
sam_twitter <- sam_twitter[nchar(sam_twitter)< third_quantT]
sam_news <- sam_news[nchar(sam_news)> first_quantN]
sam_news <- sam_news[nchar(sam_news)< third_quantN]
```

Combining the three sample files into a single data file

```
# Putting the three sample files into one
sample_data <- c(sam_blog, sam_twitter, sam_news)

# Looking at the number of lines, words in the combined data

samplelines <- length(sample_data)
samplewords <- sum(sapply(strsplit(sample_data, " "), length))
# Load the profanity file
profanitylist <- readLines("http://www.bannedwordlist.com/lists/swearWords.txt", encoding = "UTF-8", sk
```

```
## Warning in readLines("http://www.bannedwordlist.com/lists/swearWords.txt", :
## incomplete final line found on
## 'http://www.bannedwordlist.com/lists/swearWords.txt'
```

```
profanitylist <- iconv(profanitylist, "latin1", "ASCII", sub = "")
```

Cleaning the Data

The text data is cleaned to remove the following: 1. URLs and Twitter handles
2. Non-ASCII characters
3. Numbers and Punctuation
4. Additional Whitespaces

These are removed because they are irrelevant to general text prediction and/or would lead to incorrect results.

```
# Converting words to lowercase
sample_data <- tolower(sample_data)
# Removing punctuation
sample_data <- removePunctuation(sample_data)
# Removing numbers
sample_data <- removeNumbers(sample_data)
# Removing Whitespaces
sample_data <- stripWhitespace(sample_data)
# Removing URLs, Email addresses,  Twitter handles, and hashtags
sample_data <- gsub("(f|ht)tp(s?)://(.*)[.][a-z]+", "", sample_data,
                    ignore.case = FALSE, perl = TRUE)
sample_data <- gsub("\\S+[@]\\S+", "", sample_data,
                    ignore.case = FALSE, perl = TRUE)
sample_data <- gsub("@[^\\s]+", "", sample_data,
                    ignore.case = FALSE, perl = TRUE)
sample_data <- gsub("#[^\\s]+", "", sample_data,
                    ignore.case = FALSE, perl = TRUE)
# Removing profanity
sample_data <- removeWords(sample_data, profanitylist)
```

Building a corpus using the sample_data

```
EN_corpus <- corpus(sample_data)
```

Creating N Grams

This is done to understand the word distribution and relationship between phrases, tokens, and words

```
TopThree <- function(corpus) {
  first <- !duplicated(corpus$token)
  balance <- corpus[!first,]
  first <- corpus[first,]
  second <- !duplicated(balance$token)
  balance2 <- balance[!second,]
  second <- balance[second,]
  third <- !duplicated(balance2$token)
  third <- balance2[third,]
  return(rbind(first, second, third))
}
```

Observing at the frequencies of these, to see which unigrams, bigrams, trigrams, and quadgrams are most common

```
tokenFrequency <- function(corpus, n = 1, rem_stopw = NULL) {
    EN_corpus <- dfm(tokens(EN_corpus), ngrams = n)
    EN_corpus <- colSums(EN_corpus)
    total <- sum(EN_corpus)
    EN_corpus <- data.frame(names(EN_corpus),
                         EN_corpus,
                         row.names = NULL,
                         check.rows = FALSE,
                         check.names = FALSE,
                         stringsAsFactors = FALSE
    )
    colnames(EN_corpus) <- c("token", "n")
    EN_corpus <- mutate(EN_corpus, token = gsub("_", " ", token))
    EN_corpus <- mutate(EN_corpus, percent = EN_corpus$n / total)
    if (n > 1) {
        EN_corpus$outcome <- word(EN_corpus$token, -1)
        EN_corpus$token <- word(string = EN_corpus$token,
                                start = 1, end = n - 1, sep = fixed(" "))
    }
    setorder(EN_corpus, -n)
    EN_corpus <- TopThree(EN_corpus)
    return(EN_corpus)
}
```

Building the Algorithm

Finding the top 3 words to initiate the word prediction app

```
# Getting first word for each document

first_word <- word(EN_corpus$documents$texts, 1)

# Determining most popular words to start the sentence in the corpus

first_word <- tokenFrequency(first_word, n = 1, NULL)
```

```
## Warning: ngrams argument is not used.
```

```
# Selecting the top 3 words

firstthree <- first_word$token[1:3]

# Constructing bigrams, trigrams, and quadgrams
bigram <- tokenFrequency(corpus, n = 2, NULL)
```

```
## Warning: ngrams argument is not used.
```

```
trigram <- tokenFrequency(corpus, n = 3, NULL)
```

```
## Warning: ngrams argument is not used.
```

```
trigram <- trigram %>% filter(n > 1)

quadgram <- tokenFrequency(corpus, n = 4, NULL)
```

```
## Warning: ngrams argument is not used.
```

```
quadgram <- quadgram %>% filter(n > 1)
```

Saving the files to local disk to be used later in building the shiny app

```
saveRDS(quadgram, "~/Coursera/Data Science Capstone/Project Data/Word_Prediction_app/Quadrigram.RDS")

saveRDS(trigram, "~/Coursera/Data Science Capstone/Project Data/Word_Prediction_app/Trigram.RDS")

saveRDS(bigram, "~/Coursera/Data Science Capstone/Project Data/Word_Prediction_app/Bigram.RDS")

saveRDS(firstthree, "~/Coursera/Data Science Capstone/Project Data/Word_Prediction_app/Firstthree.RDS")
```

Removing the ngram frequency data from the environment after saving them to disk

```
remove(bigram)
remove(trigram)
remove(quadgram)
```