Data Science Capstone - Milestone Report

Teppei Miyazaki

12/19/2021

Executive Summary

- This report is a part of the Data Science Capstone course of Coursera.
- The final goal of this course is to build an app which predicts the next word after taking a phrase as input.
- In this milestone report, we explain exploratory analysis of the data set and our goals for the eventual app and algorithm.

Loading Data

- The data is obtained from the Coursera site (link) (https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip)
- This data is from a corpus called HC Corpora and includes large text files from blogs, news, and twitter.
- The file paths including the file name are shown below. Please note that the R code for this project is attached in the appendix.

```
## [1] "./Word-Prediction/en_US.blogs.txt"   "./Word-Prediction/en_US.news.txt"
## [3] "./Word-Prediction/en_US.twitter.txt"
```

Exploratory Analysis

- First, we look at the number of words/lines of the text files.
- As you can see, these files have large amount of texts.

Word count of each file

```
## $blogs
## [1] 37334131
##
## $news
## [1] 34372530
##
## $twitter
## [1] 30373583
```

Line count of each file

```
## $blogs
## [1] 899288
##
## $news
## [1] 1010242
##
## $twitter
## [1] 2360148
```

Tokenization

- Given the large size of data, we tokenize the 10% sample of data.
- Overview of the token is shown below.

```
## Tokens consisting of 426,967 documents.
## text1 :
##  [1] "It"         "wasn't"     "a"          "rebellion"   "The"
##  [6] "Metis"      "were"       "not"        "insurgents" "They"
## [11] "never"      "were"
## [ ... and 122 more ]
##
## text2 :
## [1] "04"      "Toot"    "Toot"    "Tootsie" "Styne"   "Green"   "Cahn"
## [8] "04"      "22"
##
## text3 :
##  [1] "Somehow"  "I"       "knew"    "Millar"  "would"   "through" "Cowboy"
##  [8] "Up"       "in"      "there"
##
## text4 :
## [1] "I'm"      "watch"    "Caged"    "Are"      "you"      "watching" "it"
## [8] "to"
##
## text5 :
##  [1] "Also"   "it"      "would"   "appear" "that"   "Tetley" "will"   "no"
##  [9] "longer" "be"      "sold"    "at"
## [ ... and 76 more ]
##
## text6 :
##  [1] "I"        "must"    "be"      "tired"   "I"        "just"    "carried"
##  [8] "my"       "cup"     "of"      "#coffee" "with"
## [ ... and 4 more ]
##
## [ reached max_ndoc ... 426,961 more documents ]
```

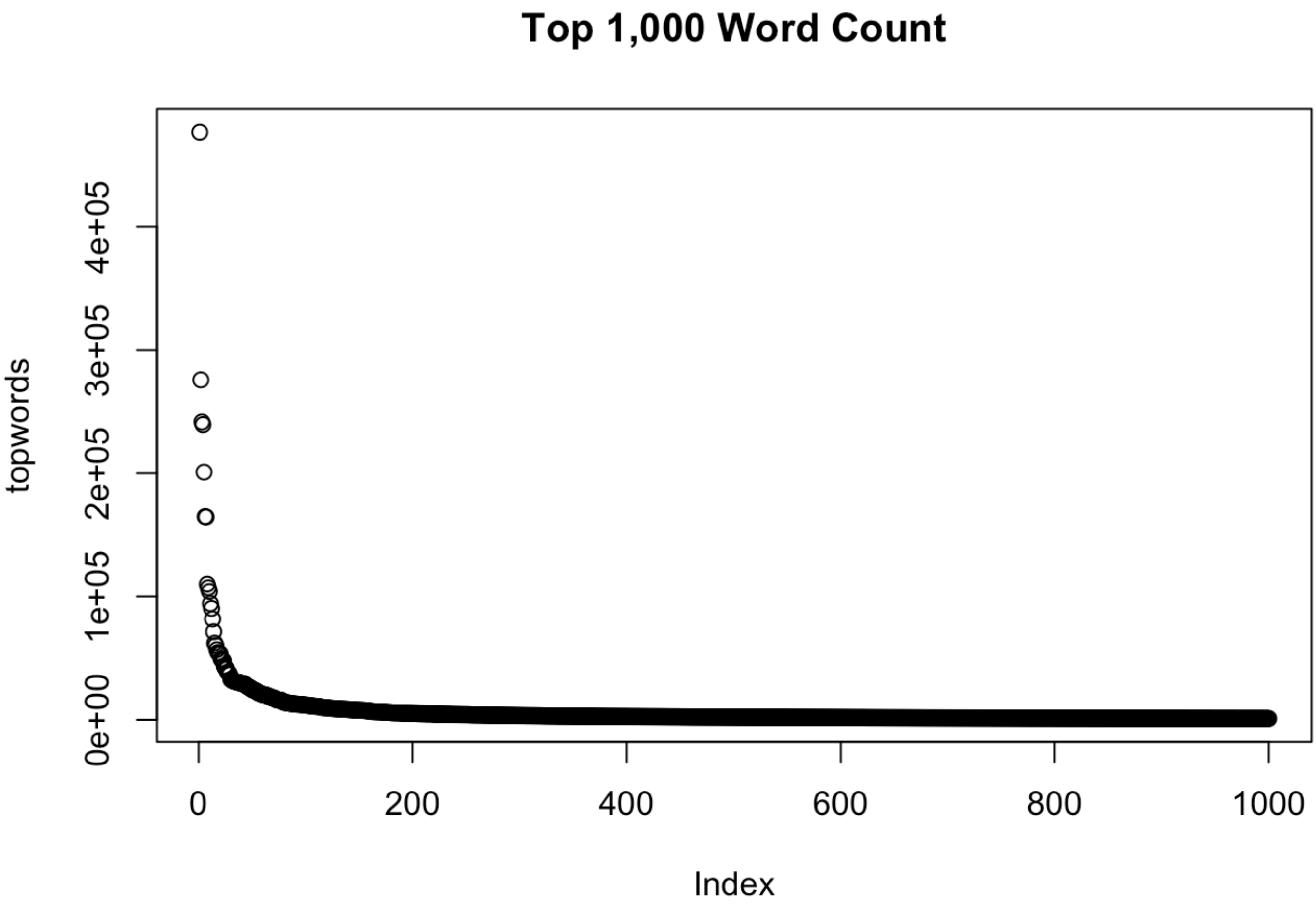
Document-feature matrix

- Then, we construct a document-feature matrix.
- Here, we show the first 6 words and the last 6 words of the top 1000 words.
- As you can see, the top 1,000 words cover approximately 70% of all words instances.

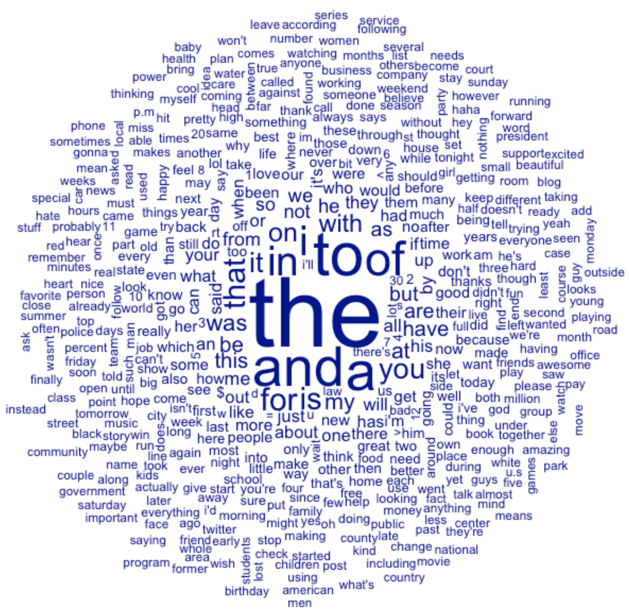
- We plot frequency of words below and you can see that top words accounts for large proportion of data.
- We also plot a word cloud of the data using textplot_wordcloud() function.

```
##     topwords proportion    cum_sum
## the   476527 0.04686830 0.04686830
## to    275724 0.02711854 0.07398683
## and   241513 0.02375375 0.09774058
## a     239381 0.02354406 0.12128464
## of    200902 0.01975950 0.14104414
## i     164829 0.01621158 0.15725572
```

```
##              topwords   proportion   cum_sum
## chicken          1087 0.0001069107 0.6990524
## development      1086 0.0001068124 0.6991592
## deep             1086 0.0001068124 0.6992660
## photos           1085 0.0001067140 0.6993727
## plus             1083 0.0001065173 0.6994792
## restaurant       1083 0.0001065173 0.6995857
```



Top 1,000 Word Count
topwords
0e+00
1e+05
2e+05
3e+05
4e+05
0
200
400
600
800
1000
Index



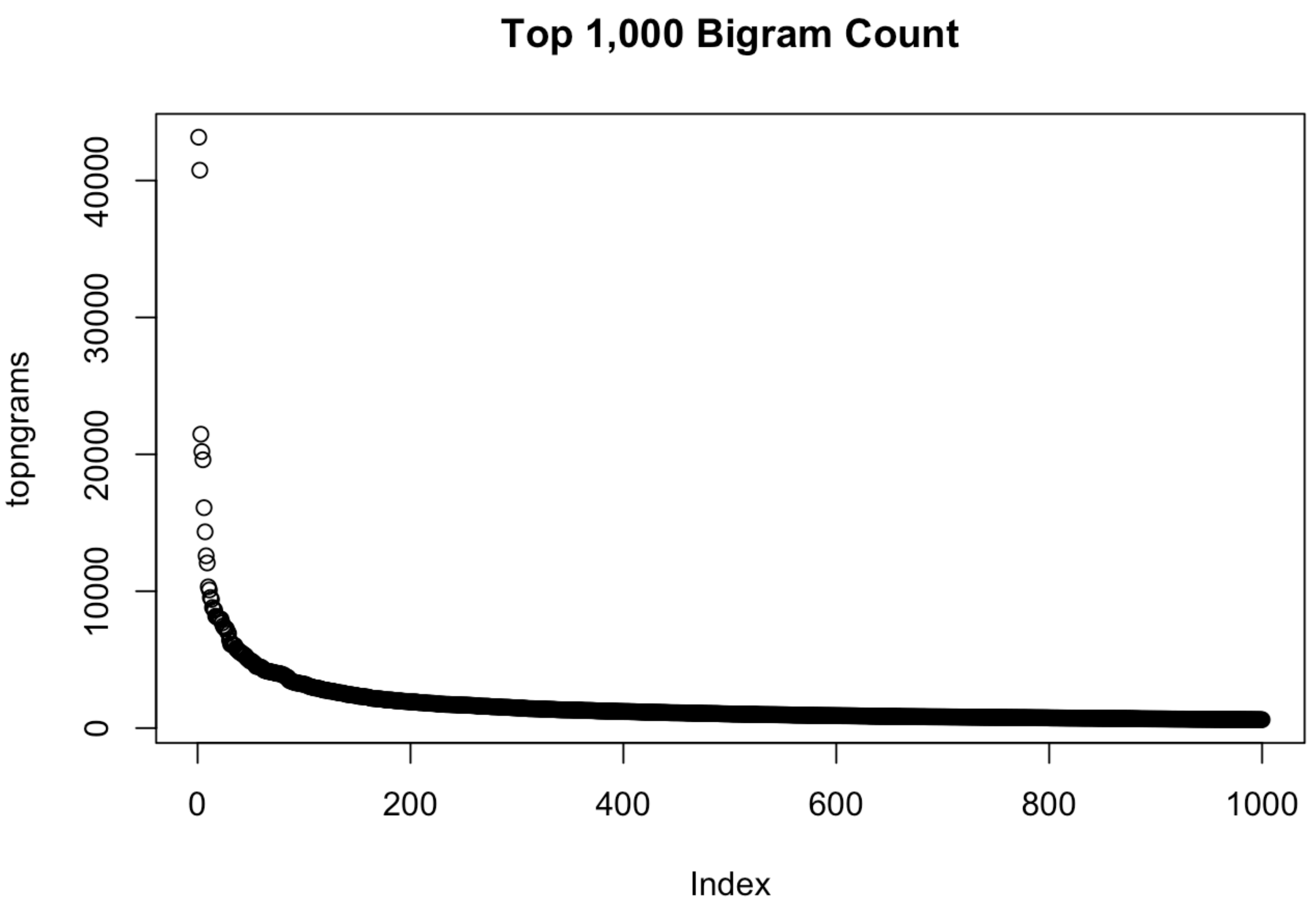
the
and
a
to
of
in
i
for
is
on
that
it
with
you
was
have

n-grams (n = 2)

- Finally, we generate n-grams using tokens_ngrams() function.
- Given the large size of data, we generate bigram only.
- The frequency of bigrams look similar to the previous plot and top bigrams account for large portion of data.

```
##          topngrams  proportion     cum_sum
## of_the       43165 0.004431541 0.004431541
## in_the       40756 0.004184220 0.008615761
## to_the       21468 0.002204015 0.010819776
## for_the      20205 0.002074349 0.012894125
## on_the       19602 0.002012442 0.014906567
## to_be        16101 0.001653011 0.016559578
```

```
##             topngrams   proportion   cum_sum
## just_to           617 6.334439e-05 0.1771516
## would_love        617 6.334439e-05 0.1772149
## people_to         616 6.324172e-05 0.1772782
## a_bad             616 6.324172e-05 0.1773414
## we_got            616 6.324172e-05 0.1774046
## was_on            616 6.324172e-05 0.1774679
```



Next Steps

- The final goal of this project is to create the prediction algorithm and Shiny app.
- The application takes a phrase as input, and it predicts the next word.
- As a next step, we would like to work on n-gram model for predicting the next word based on the previous words.

Appendix: R Code

```
# setup
setwd("~/Desktop/Coursera"); set.seed(0)
library(tidyverse); library(quanteda); library(quanteda.textplots)
library(ngram); library(textclean); library(sentimentr)

# loading_data
processFile <- function(path){
        txts <- scan(path, what = character(),
                     sep = "\n", blank.lines.skip = TRUE,
                     skipNul = TRUE, quiet = TRUE)
        return(txts)
}

file_paths <- list.files(path = "./final/en_US", full.names = TRUE)
file_list <- lapply(file_paths, processFile)
file_names <- c("blogs", "news", "twitter")
names(file_list) <- file_names

# file paths
file_paths

# wordcount
wordcountFile <- function(file){
        n <- wordcount(file)
        n_sum <- sum(n, na.rm = TRUE)
        return(n_sum)
}
wordcount_list <- lapply(file_list, wordcountFile)
wordcount_list

# linecount
linecount_list <- lapply(file_list, length)
linecount_list

# 10% sampling and tokenizing
tokenizeFile <- function(files, p = 0.1){
        docs <- unlist(files)
        size <- length(docs) * p
        docs <- sample(docs, size = size)
        docs <- replace_non_ascii(docs)
        corp <- corpus(docs)
        toks <- tokens(corp, remove_punct = TRUE)

        # removing bad words
        pwords <- lexicon::profanity_alvarez
```

```
        toks <- tokens_remove(toks, pattern = pwords)
        return(toks)
}

toks <- tokenizeFile(file_list, p = 0.1)
toks

# constructing a document-feature matrix
dfmat <- dfm(toks)
topwords <- topfeatures(dfmat, 1000)
df1 <- data.frame(topwords) %>%
        mutate(proportion = topwords/sum(dfmat)) %>%
        mutate(cum_sum = cumsum(proportion))
head(df1); tail(df1) # Top 1000 words cover 70% of all words instances
plot(topwords, main = "Top 1,000 Word Count")
textplot_wordcloud(dfmat, min_count = 1000)

# generating n-grams (n = 2)
toks_ngrams <- tokens_ngrams(toks, n = 2)
dfmat_ngrams <- dfm(toks_ngrams)
topngrams <- topfeatures(dfmat_ngrams, 1000)
df2 <- data.frame(topngrams) %>%
        mutate(proportion = topngrams/sum(dfmat_ngrams)) %>%
        mutate(cum_sum = cumsum(proportion))
head(df2); tail(df2)
plot(topngrams, main = "Top 1,000 Bigram Count")
```