Milestone Report for Data Science Capstone Project

Thawatchai Phakwithoonchai

4/13/2020

1. Objective

The objective of this report is to show the summary result of exploratory data analysis prior to build the subsequent app. The major features of the data are also identified and briefly summarize in order to create the prediction algorithm and Shiny app. **Please be note that all relevant coding sessions are shown in the appendix**, while all object are stored and loaded from the specific .RData environment.

2. Loading, Getting, and Basic Summary of the data

2.1 Load the relevant library package in order to perform the entire operation for data analysis.

```
library(ggplot2)  # Data visualization
library(NLP)  # Basic classes and methods for natural language processing
library(tm)  # Text mining applications
```

2.2 Download the **Swifkey** dataset and then unzip the data. It is found that there are datasets, which consist of 4 folders that indicate the different languages, while each folders have 4 files: blogs, news, and twitter. This project study will be, therefore, focus on English language files, that are: [1] en_US.blogs.txt, [2] en_US.news.txt, and [3] en_US.twitter.txt

```
fileURL <- "https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip"
download.file(fileURL, "./Coursera-SwiftKey.zip")
unzip("./Coursera-SwiftKey.zip")
```

2.3 Load all of en_US data file from the specific folder location for information gathering. The information from the original dataset can be shown as following table:

```
##                   Blogs     News      Twitter
## File Size:        210160014 205811889 167105338
## Object Size:      267758632 20729472  334484736
## Numbers of Lines: 899288    77259     2360148
## Numbers of Words: 38625966  2734672   32207334
```

3. Data Preparation

3.1 Sample the dataset since the files are very large (> 150 MB), the model does not needed to load and use all of the dataset. Therefore, only **5%** of each dataset are randomly selected. Function *rbinom* is used to generate the random sample.

3.2 Create the corpus from multiple sample datafiles. All non-word characters e.g. emoji, latin, etc. are also removed in this step.

3.3 Clean the dataset, whic in **tm**, all of function is subsumed into the concept of a *transformation*. Transformations are done via the **tm_map()** function which applies (maps) a function to all elements of the corpus. All transformation steps are consist of:

1. Convert all letters to lower case

2. Remove all of URL
3. Remove all of the numbers

4. Remove all of punctuation except hyphens between words and apostrophes in contractions e.g. don't, can't, etc.

5. Remove all of stopwords (i.e. "and", "or", "not", "is", etc.)

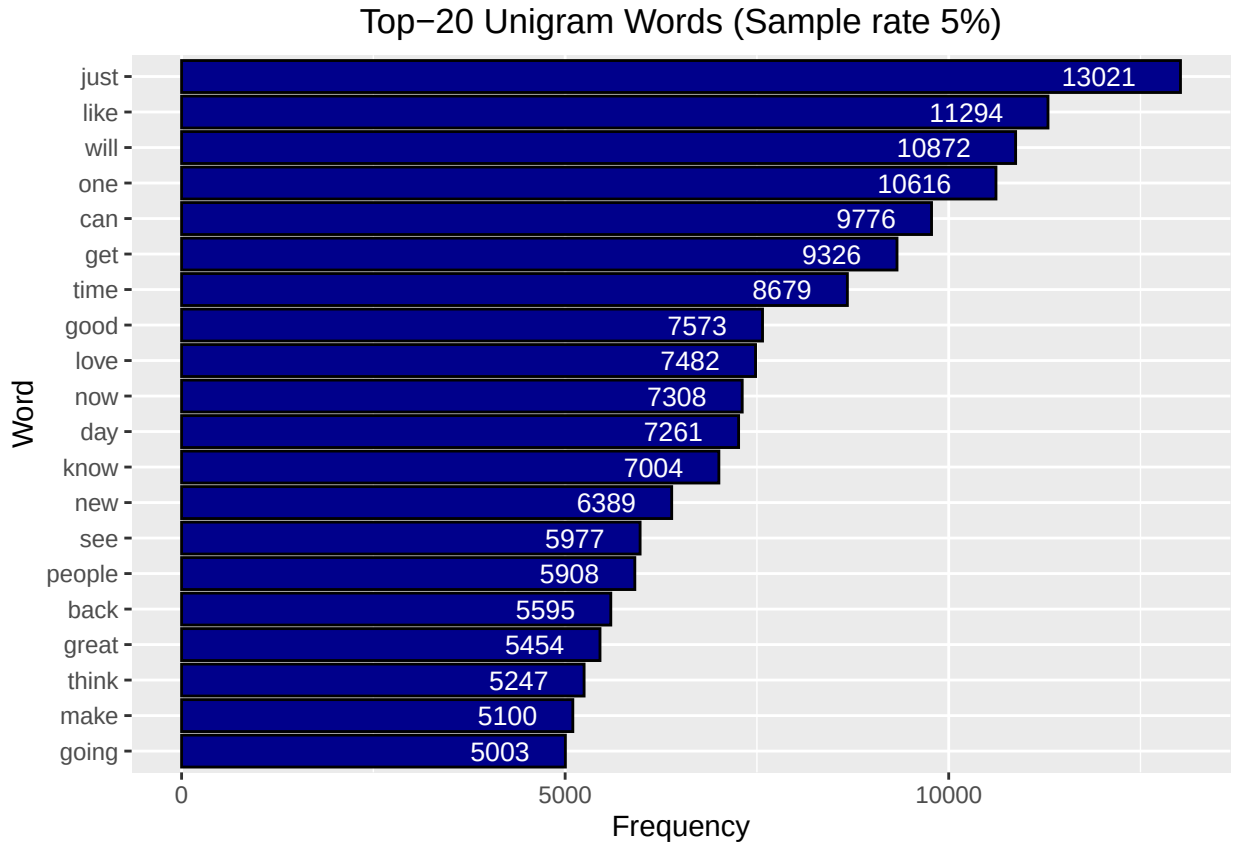
6. Remove all of profanity

7. Remove all of white spaces

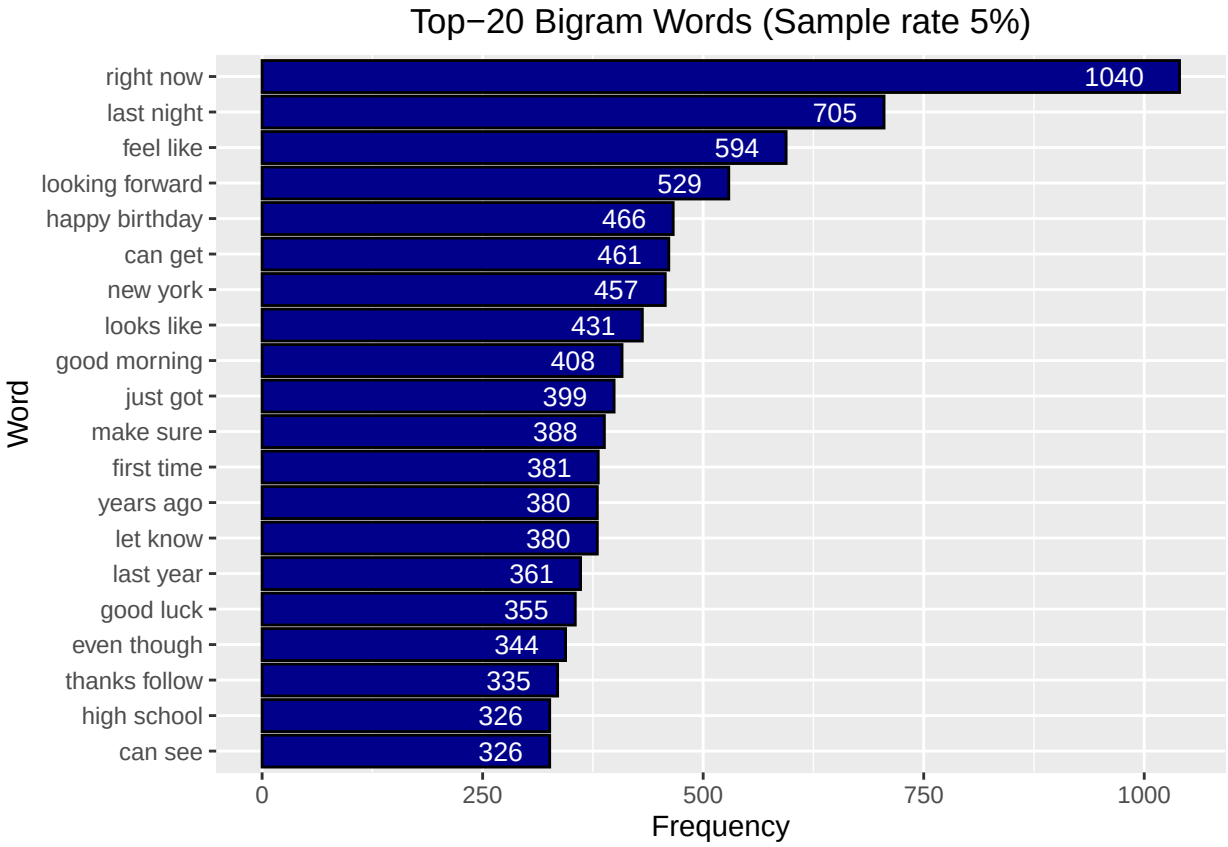
4. Exploratory Data Analysis

For exploratory data analysis, it is necessary to understanding the distribution of words and relationship between the words in the corpora. Word tokenization and following up process with (n-gram) words histogram plot can help to understand variation in the frequencies of words and word groups (n-gram) in the corpora.

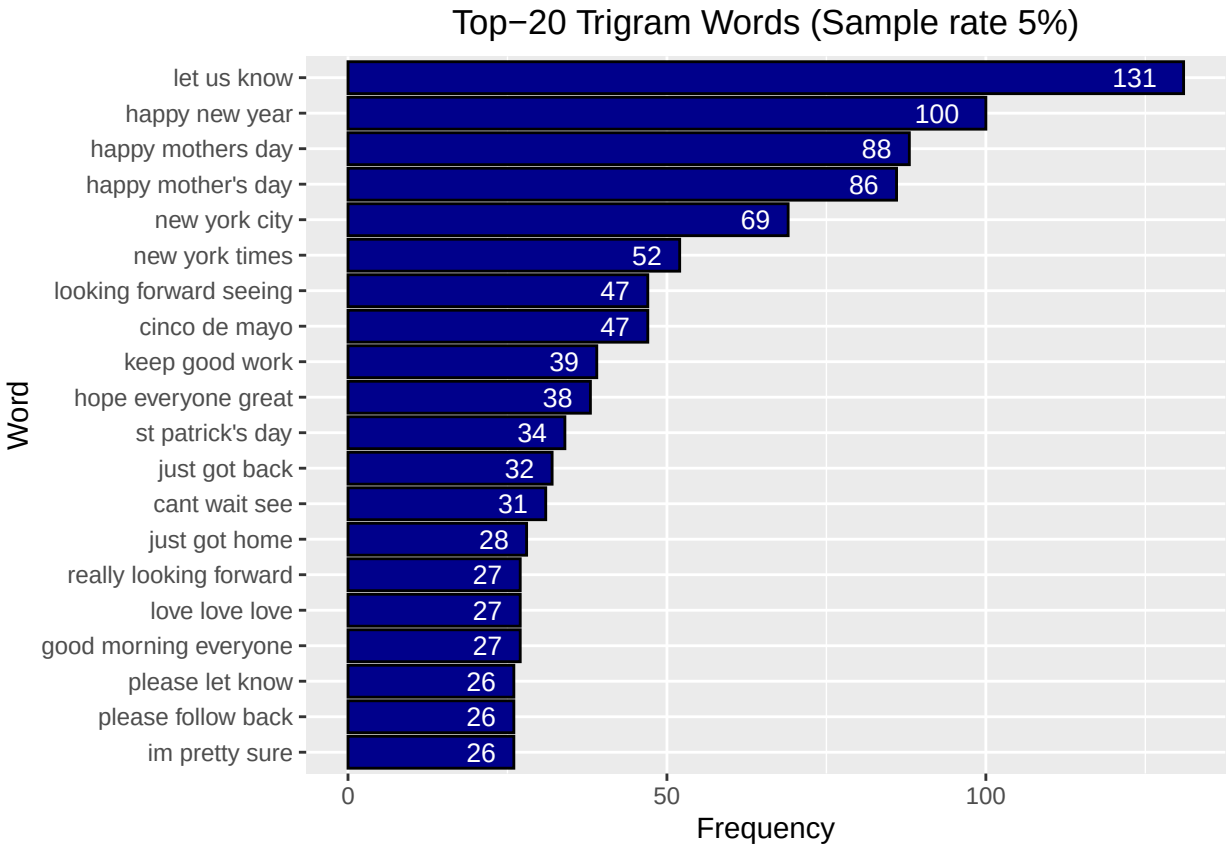
4.1 Unigram


Top−20 Unigram Words (Sample rate 5%)
Word
just
like
will
one
can
get
time
good
love
now
day
know
new
see
people
back
great
think
make
going
13021
11294
10872
10616
9776
9326
8679
7573
7482
7308
7261
7004
6389
5977
5908
5595
5454
5247
5100
5003
0
5000
10000
Frequency

4.2 Bigram


Top−20 Bigram Words (Sample rate 5%)
Word
right now
last night
feel like
looking forward
happy birthday
can get
new york
looks like
good morning
just got
make sure
first time
years ago
let know
last year
good luck
even though
thanks follow
high school
can see
1040
705
594
529
466
461
457
431
408
399
388
381
380
380
361
355
344
335
326
326
0
250
500
750
1000
Frequency

4.3 Trigram



5. Findings

5.1 One of the most concern aspect in building the prediction model is the size and runtime. Different sample rate 1%, 5%, and 10%, which cause the object (corpus) size was major difference (137MB, 690 MB, and 1.3 GB, respectively), are selected from each of datasets and proceed for the data exploratory analysis in order to verify the variation of result. The result indicate that it is similar in the n-gram tokenization between 5% and 10% sample rate but there are major difference in between 1% and the others. Therefore, it is reasonable to select 5% sample rate as baseline for the corpus dataset in order to build the subsequent prediction model.

5.2 A lot of sparse data is observed in the corpus. It is found that unigram has 99% sparsity. While bigram and trigram show even more sparsity that are 99.9 and 99.99% sparsity, respectively.

6. Feedback on the plan

6.1 Manage the sparsity (data) and unkown word that may not appear in the corpus that cause any n-gram word frequency equal to zero by applying the various smoothing method e.g. laplace (add-one), interploation, backoff, etc.

6.2 Calculate the probabilities in each n-gram model as transition that allow the model to predict the next word given the current word based on Markov Chains work.

6.3 Build the prediction model based on the dataset, which will be splitted into 3 parts: training (70%), validation (15%), and testing (15%)

6.4 Evaluate the performance and perplexity of prediction model

6.5 Build the prediction model on Shiny app. User will input the word and the model will predicts the next word that user might enter. Options will be provided to the user, similar to the smart keyboard used on mobile devices.

Appendix

```
# Get the data file
fileURL <- "https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip"
download.file(fileURL, "./Coursera-SwiftKey.zip")
unzip("./Coursera-SwiftKey.zip")

# Define the file path
DataFolder <- "D:/Coursera/Data Science - Specialization (by Johns Hopkins University)/10-Data Science C
Prefix <- "en_US."
BlogsName <- "blogs.txt"
NewsName <- "news.txt"
TwittName <- "twitter.txt"

# Define the function for gathering the file information
NumWord <- vector()
FileInfo <- function(FileName) {
    FileSize <- file.info(FileName)$size
    Con <- file(FileName, open = "r")
    TextObj <- readLines(Con, warn = FALSE, encoding = "UTF-8")
    MemUsed <- object.size(TextObj)
    NumLines <- length(TextObj)
    for (i in 1:NumLines) {
        NumWord[i] <- lengths(gregexpr("[A-z]\\W+", TextObj[i])) + 1L
    }
    SumWords <- sum(NumWord)
    close(Con)
    # create list of info
    list(FileSize = FileSize, MemoryUsed = MemUsed, NumLines = NumLines,
        NumWords = SumWords)
}

# Run the function for each files
BlogsInfo <- FileInfo(paste0(DataFolder, Prefix, BlogsName))
NewsInfo <- FileInfo(paste0(DataFolder, Prefix, NewsName))
TwittInfo <- FileInfo(paste0(DataFolder, Prefix, TwittName))

# Create information matrix
infoMat <- matrix(c(BlogsInfo[1], NewsInfo[1], TwittInfo[1], as.numeric(format(c(BlogsInfo[2],
    NewsInfo[2], TwittInfo[2]))), BlogsInfo[3], NewsInfo[3], TwittInfo[3],
    BlogsInfo[4], NewsInfo[4], TwittInfo[4]), nrow = 4, ncol = 3, byrow = TRUE,
    dimnames = list(c("File Size:", "Object Size:", "Numbers of Lines:",
        "Numbers of Words:"), c("Blogs", "News", "Twitter")))

# Create the sample data files
sampleRT <- 0.05  # 5% sampling from dataset
samplePrefix <- "Sample"  # Prefix 'Sample' for sample datasets
```

```r
SampleFile <- function(FileLocation, FileName) {
    Con <- file(paste0(FileLocation, FileName), open = "r")
    TextObj <- readLines(Con, warn = FALSE, encoding = "UTF-8")
    NumLines <- length(TextObj)
    set.seed(12345)
    inSample <- rbinom(NumLines, 1, sampleRT) == 1
    Sample <- TextObj[inSample]
    writeLines(Sample, con = file(paste0(DataFolder, samplePrefix, FileName),
        "w"))
    closeAllConnections()
}

# Run the function for each files
SampleFile(paste0(DataFolder, Prefix), BlogsName)
SampleFile(paste0(DataFolder, Prefix), NewsName)
SampleFile(paste0(DataFolder, Prefix), TwittName)

# Create readLines function from multiple file location
SampleRead <- function(FileLocation, FileName) {
    Con <- file(paste0(FileLocation, FileName), open = "r")
    TextObj <- readLines(Con, warn = FALSE, encoding = "UTF-8")
    close(Con)
    TextObj
}

# Run the read function
SampleBlogs <- SampleRead(paste0(DataFolder, samplePrefix), BlogsName)
SampleNews <- SampleRead(paste0(DataFolder, samplePrefix), NewsName)
SampleTwitt <- SampleRead(paste0(DataFolder, samplePrefix), TwittName)

# Remove any emoji and latin characters, and then encode to ASCII
# encoding
SampleBlogs <- iconv(SampleBlogs, "latin1", "ASCII", sub = "")
SampleNews <- iconv(SampleNews, "latin1", "ASCII", sub = "")
SampleTwitt <- iconv(SampleTwitt, "latin1", "ASCII", sub = "")

# Create the corpus and remove the existing sample object
corpus <- VCorpus(VectorSource(c(SampleBlogs, SampleNews, SampleTwitt)),
    readerControl = list(readPlain, language = "en", load = TRUE))

# Clean the corpus
corpus <- tm_map(corpus, content_transformer(tolower))  # lower case
removeURL <- content_transformer(function(x) gsub("(f|ht)tp(s?)://\\S+",
    "", x, perl = T))
corpus <- tm_map(corpus, removeURL)  # remove URLs
corpus <- tm_map(corpus, removeNumbers)  # remove numbers
corpus <- tm_map(corpus, removePunctuation, preserve_intra_word_dashes = TRUE,
    preserve_intra_word_contractions = TRUE)  # rm punctuations
corpus <- tm_map(corpus, removeWords, stopwords("english"))  # rm stop words
profanityURL <- url("http://www.bannedwordlist.com/lists/swearWords.txt")
profanity <- readLines(profanityURL, warn = FALSE)
close(profanityURL)
corpus <- tm_map(corpus, removeWords, profanity)  # rm profanity
```

```r
corpus <- tm_map(corpus, stripWhitespace)  # rm white space

# Get the n-gram frequency function
ngramFreq <- function(tdm) {
    freq <- sort(rowSums(as.matrix(tdm)), decreasing = TRUE)
    return(data.frame(word = names(freq), freq = freq))
}

# Term-document matrix
Unigram <- function(x) unlist(lapply(ngrams(words(x), 1), paste, collapse = " "),
    use.names = FALSE)
UniTdm <- TermDocumentMatrix(corpus, control = list(tokenize = Unigram))

Bigram <- function(x) unlist(lapply(ngrams(words(x), 2), paste, collapse = " "),
    use.names = FALSE)
BiTdm <- TermDocumentMatrix(corpus, control = list(tokenize = Bigram))

Trigram <- function(x) unlist(lapply(ngrams(words(x), 3), paste, collapse = " "),
    use.names = FALSE)
TriTdm <- TermDocumentMatrix(corpus, control = list(tokenize = Trigram))

# Sort the n-gram words while remove the sparse data
Unifreq <- ngramFreq(removeSparseTerms(UniTdm, 0.99))
Bifreq <- ngramFreq(removeSparseTerms(BiTdm, 0.999))
Trifreq <- ngramFreq(removeSparseTerms(TriTdm, 0.9999))

# Plot the n-gram words distribution
UniPlot <- ggplot(Unifreq[1:20, ], aes(x = reorder(word, freq), y = freq)) +
    geom_bar(stat = "identity", color = "black", fill = "darkblue") + coord_flip() +
    geom_text(aes(label = freq), hjust = 1.6, color = "white", size = 3.5) +
    labs(title = "Top-20 Unigram Words (Sample rate 5%)", x = "Word", y = "Frequency") +
    theme(plot.title = element_text(hjust = 0.5))

BiPlot <- ggplot(Bifreq[1:20, ], aes(x = reorder(word, freq), y = freq)) +
    geom_bar(stat = "identity", color = "black", fill = "darkblue") + coord_flip() +
    geom_text(aes(label = freq), hjust = 1.6, color = "white", size = 3.5) +
    labs(title = "Top-20 Bigram Words (Sample rate 5%)", x = "Word", y = "Frequency") +
    theme(plot.title = element_text(hjust = 0.5))

TriPlot <- ggplot(Trifreq[1:20, ], aes(x = reorder(word, freq), y = freq)) +
    geom_bar(stat = "identity", color = "black", fill = "darkblue") + coord_flip() +
    geom_text(aes(label = freq), hjust = 1.6, color = "white", size = 3.5) +
    labs(title = "Top-20 Trigram Words (Sample rate 5%)", x = "Word", y = "Frequency") +
    theme(plot.title = element_text(hjust = 0.5))

# End of Code
```