Peer-graded Assignment: Milestone Report

Author: Isaac G Veras

October 5, 2023

/ Instructions

The goal of this project is to publish on R Pubs a report that explains the exploratory analysis and its goals for the application and algorithm. This document should be concise and explain only the key features of the data you identified and briefly summarize the plans for creating the Shiny prediction algorithm and application in a way that is understandable to a non-Data Scientist manager. Present tables and graphs to illustrate important summaries of the data set. The motivation for this project, therefore, is:

1. Demonstrate that the data download was imported and loaded successfully.
2. Create a basic summary statistics report on the datasets.
3. Report all interesting discoveries to date.
4. Get feedback on plans to create a Shiny prediction algorithm and app.

Libraries to be loaded for Data Analysis

```
if (!require("pacman")) install.packages("pacman")
pacman::p_load(
        pacman,
        knitr,
        tidyverse,
        NLP,
        openNLP,
        qdapDictionaries,
        qdapRegex,
        qdapTools,
        slam,
        tools,
        RWeka,
        ngram,
```

```
        stringr,
        RColorBrewer,
        SnowballC,
        wordcloud,
        wordcloud2
)
```

Final Product

The final product of the project will be an algorithm that predicts the next word in text provided with inputs from the test dataset, similar to the text prediction functions found in today's modern smartphones.

/ Data Set

HC Corpora's `Swiftkey` dataset is comprised of output from several News, Blogs and Twitter sites. The dataset contains 3 files in four languages (Russian, Finnish, German and English). This project will focus on English (en) language datasets:

1. en_US.blogs.txt
2. en_US.twitter.txt
3. en_US.news.txt

`NOTE:`
`and these will be referred to as "Blogs", "Twitter" and "News" in the remainder of this repo`

Tasks performed on data

1. Explore the training dataset;
2. Profanity filtering - removal of inappropriate terms;;
3. Tokenization – identifying appropriate tokens such as words, punctuation and numbers

// Importing the Data:

The training datasets for this project were downloaded via this link

```
library(dplyr)
library(kableExtra)

blogsC        <- file("./swiftkeys/en_US.blogs.txt"  , "r")
newsC         <- file("./swiftkeys/en_US.news.txt"   , "r")
twitterC      <- file("./swiftkeys/en_US.twitter.txt", "r")

blogs         <- readLines(blogsC,   n = -1, encoding = "UTF-8")
news          <- readLines(newsC,    n = -1, encoding = "UTF-8")
twitter       <- readLines(twitterC, n = -1, encoding = "UTF-8")

close(blogsC)
close(newsC)
close(twitterC)

nCharBlogs    <- sum(nchar(blogs))
nCharNews     <- sum(nchar(news))
nCharTwitter  <- sum(nchar(twitter))
```

```
lenBlogs      <- length(blogs)
lenNews       <- length(news)
lentwitter    <- length(twitter)

nWordsBlogs   <- sum(sapply(strsplit(blogs, " "), length))
nWordsNews    <- sum(sapply(strsplit(news, " "), length))
nWordsTwitter <- sum(sapply(strsplit(twitter, " "), length))

outputTable   <- data.frame(
    c("blogs", "news", "twitter"),
    c(nCharBlogs, nCharNews, nCharTwitter),
    c(lenBlogs, lenNews, lentwitter),
    c(nWordsBlogs, nWordsNews, nWordsTwitter)
)
colnames(outputTable) <- c("FileType", "Characters", "Lines", "Words")

rm(blogs, news, twitter)
```

/// SwiftKeys (EN-US)

FileType	Characters	Lines	Words
blogs	206824505	899288	37334131
news	15639408	77259	2643969
twitter	162096031	2360148	30373543

Steps taken:

- Code to download the file and unzip is hidden and its with the assumption that files are available in the current working directory;
- Open the file connection;
- Close connection;
- Compute n°. of characters;
- Compute n°. of lines;
- Compute n°. of words;
- Build a Dataset Summary output Table;
- Release object memory;
- Summary of Data Analysis.

// Exploratory Analysis of Data

Since the training data set size is huge, a sample of each file is extracted and explored for further study

/// Sample Data:

FileType	Characters	Lines	Words
blogsSample	2277384	10000	410620

FileType	Characters	Lines	Words
newsSample	2035687	10000	343929
twitterSample	681544	10000	127674

Steps taken:

- Set the line count information for extraction;
- Open the file connection;
- Close the file connection;
- Compute n°. of characters;
- Compute n°. of lines;
- Compute n°. of words;
- Build a Dataset Summary output Table

/ Data Cleansing & Corpora building

The text sample extracted from each of the files is transformed step-by-step for Predictive Model bulding

Steps taken:

`NOTE`
`Optional steps - collate Sample Text Data and create a reduced raw data file before processi`

```
beforeData <- c(blogsS, newsS, twitterS)
```

- Remove retweets from Twitter Data Sample;
- Remove @people names from Twitter;
- Collate text data from different file samples;
- Replace abbreviation so that the sentences are not split at incorrect places;

`NOTE:` `Optional steps - convert paragraphs to sentences`

```
endNotations <- c("?", ".", ",","!", "|", ":", "\n", "\r\n")<
sampleData <- sent_detect(
                        sampleData,
                        endmarks   = endNotations,
                        rm.bracket = FALSE
                        )
```

- Collate Sample Text Data and create a reduced raw data file;
- Create Text Corpus for processing;
- Release object memory;
- Convert the text to lower case;
- Remove URL from Corpora, symbols;
- Replace emails and such but space, websites and file systems;
- Edit out most non-alphabetical character, text must be lower case first;
- Remove the punctuations after trimming leading & trailing white spaces;
- Remove numbers from the text;

- Build profane word list;
- Remove profane words from the corpora;
- Remove all the white space that was created by the removals;
- Sample Text Data after processing.

Please find below the first few lines of text data before and after processing in the Corpora:

```
## Before Text Processing
sampleTxtBP
```

```
##  [1] "In the years thereafter, most of the Oil fields and platforms were"
##  [2] "named after pagan “gods”."
##  [3] "We love you Mr. Brown."
##  [4] "Chad has been awesome with the kids and holding down the fort while I"
##  [5] "work later than usual! The kids have been busy together playing"
##  [6] "Skylander on the XBox together, after Kyan cashed in his $$$ from his"
##  [7] "piggy bank. He wanted that game so bad and used his gift card from his"
##  [8] "birthday he has been saving and the money to get it (he never taps into"
##  [9] "that thing either, that is how we know he wanted it so bad). We made"
## [10] "him count all of his money to make sure that he had enough! It was very"
## [11] "cute to watch his reaction when he realized he did! He also does a very"
## [12] "good job of letting Lola feel like she is playing too, by letting her"
## [13] "switch out the characters! She loves it almost as much as him."
## [14] "so anyways, i am going to share some home decor inspiration that i have"
## [15] "been storing in my folder on the puter. i have all these amazing images"
## [16] "stored away ready to come to life when we get our home."
## [17] "With graduation season right around the corner, Nancy has whipped up a"
## [18] "fun set to help you out with not only your graduation cards and gifts,"
## [19] "but any occasion that brings on a change in one's life. I stamped the"
## [20] "images in Memento Tuxedo Black and cut them out with circle"
## [21] "Nestabilities. I embossed the kraft and red cardstock with TE's new"
## [22] "Stars Impressions Plate, which is double sided and gives you 2"
## [23] "fantastic patterns. You can see how to use the Impressions Plates in"
## [24] "this tutorial Taylor created. Just one pass through your die cut"
## [25] "machine using the Embossing Pad Kit is all you need to do - super easy!"
```

```
## After Text Processing
sampleTxtAP
```

```
##  [1] "in the years thereafter most of the oil fields and platforms were named"
##  [2] "after pagan gods"
##  [3] "we love you mister brown"
##  [4] "chad has been awesome with the kids and holding down the fort while i"
##  [5] "work later than usual the kids have been busy together playing"
##  [6] "skylander on the xbox together after kyan cashed in his from his piggy"
##  [7] "bank he wanted that game so bad and used his gift card from his"
##  [8] "birthday he has been saving and the money to get it he never taps into"
##  [9] "that thing either that is how we know he wanted it so bad we made him"
## [10] "count all of his money to make sure that he had enough it was very cute"
## [11] "to watch his reaction when he realized he did he also does a very good"
## [12] "job of letting lola feel like she is playing too by letting her switch"
```

```
## [13] "out the characters she loves it almost as much as him"
## [14] "so anyways i am going to share some home decor inspiration that i have"
## [15] "been storing in my folder on the puter i have all these amazing images"
## [16] "stored away ready to come to life when we get our home"
## [17] "with graduation season right around the corner nancy has whipped up a"
## [18] "fun set to help you out with not only your graduation cards and gifts"
## [19] "but any occasion that brings on a change in ones life i stamped the"
## [20] "images in memento tuxedo and cut them out with circle nestabilities i"
## [21] "embossed the kraft and red cardstock with tes new stars impressions"
## [22] "plate which is double sided and gives you fantastic patterns you can"
## [23] "see how to use the impressions plates in this tutorial taylor created"
## [24] "just one pass through your cut machine using the embossing pad kit is"
## [25] "all you need to do super easy"
```

/ Tokenization of text - Creation of N-grams

Perform Tokenization, and thus obtain one (uni-), two (bi-), three (tri-), and four (tetra-) word combinations that appear frequently in the Text Corpus.

```
library(dplyr)

strCorpus <- concatenate(lapply(sampleCorpora,"[",1))
ng1 <- ngram(strCorpus, n=1)
ng2 <- ngram(strCorpus, n=2)
ng3 <- ngram(strCorpus, n=3)
ng4 <- ngram(strCorpus, n=4)
```

/ Inspect first few entries in N-grams generated

// NG1

ngrams	freq	prop
the	41835	0.0466667
“,	25117	0.0280179
to	23813	0.0265633
and	22437	0.0250284
a	21024	0.0234522
of	18805	0.0209769
in	14262	0.0159092
i	11996	0.0133815
that	9492	0.0105883

ngrams	freq	prop
is	9220	0.0102849

// NG2

ngrams	freq	prop
of the	4119	0.0045947
in the	3679	0.0041039
to the	1993	0.0022232
on the	1654	0.0018450
for the	1633	0.0018216
“,”	1532	0.0017089
to be	1448	0.0016152
“,”the	1228	0.0013698
and the	1206	0.0013453
at the	1179	0.0013152

// NG3

ngrams	freq	prop
one of the	300	0.0003346
a lot of	278	0.0003101
the u s	158	0.0001762
to be a	151	0.0001684
as well as	145	0.0001617
the end of	139	0.0001551
going to be	133	0.0001484
out of the	127	0.0001417
i don t	122	0.0001361
some of the	121	0.0001350

// NG4

ngrams	freq	prop
the end of the	75	8.37e-05
at the end of	59	6.58e-05
for the first time	57	6.36e-05
the rest of the	56	6.25e-05
in the middle of	51	5.69e-05
“,”thanks for the	48	5.35e-05
one of the most	47	5.24e-05
is one of the	43	4.80e-05
when it comes to	40	4.46e-05
at the same time	37	4.13e-05

// Make new sentence:

To make new sentence using the n-grams generated, try babble(`ng=ng2, genlen=15, seed= 123445`) which would return a random formed 15 word length sentence.

```
        babble(ng=ng2, genlen=15, seed= 12112344)
```

```
## [1] "you francophiles during june at \", \"fax \", \"indeed the uninsured crisis itself is
```

/ Visual Inspection of tokenized words

Using the corpus of documents, we now construct a Document Term Matrix (DTM). This object is a simple triplet matrix structure (efficient for storing large sparse matrices), that has each document as a row and each n-gram (or term) as a column.

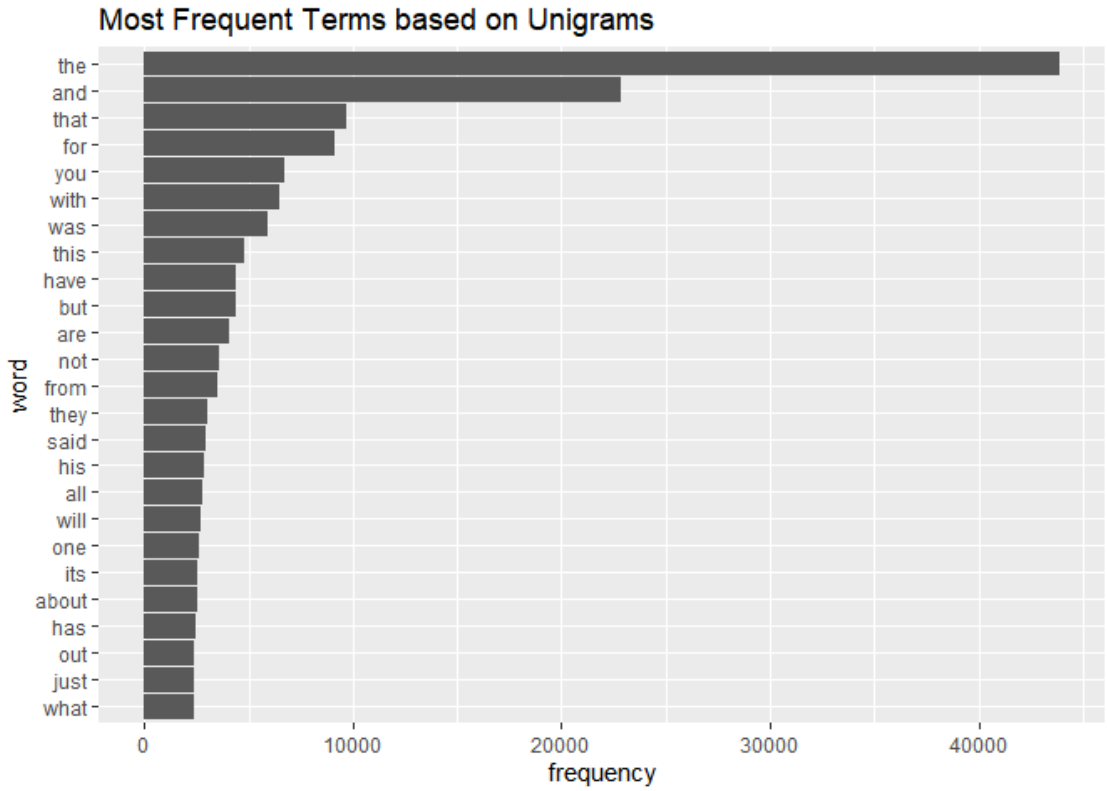
- Build Term document matrix with single tokenizer and words smaller than 3 characters are omitted:

```
        sampleTDM <- tm::TermDocumentMatrix(sampleCorpora, control = list(wordLengths =
c(3,Inf)))

        #Put word count from TDM to data frame
        sampleFreqWords <- data.frame(word = sampleTDM$dimnames$Terms, frequency =
sampleTDM$v)

        # Reorder the word list in descending order
```

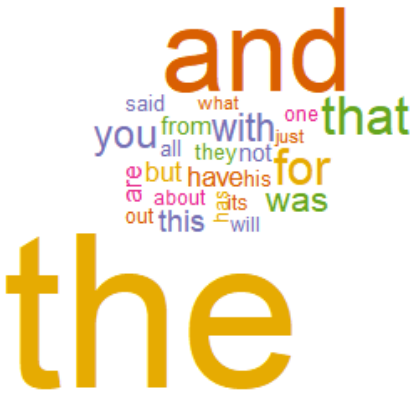
```
        sampleFreqWords <- plyr::arrange(sampleFreqWords, -frequency)

        # Build Most frequent terms
        n <- 25L # variable to set top n words
        # isolate top n words by decreasing frequency
        sampleFreqWords.top <- sampleFreqWords[1:n, ]
        # reorder levels so charts plot in order of frequency
        sampleFreqWords.top$word <- reorder(sampleFreqWords.top$word,
sampleFreqWords.top$frequency)
```

/ Term analysis

// Frequent Terms



// Wordcloud



Next Steps Forward - Prediction Algorithm

Moving forward, the project goal is to develop a natural language prediction algorithm and app. For example, if a user were to type, “I want to go to the .”, the app would suggest the three most likely words that would replace “.”.

/ N-gram Dictionary

While the word analysis performed in this document is helpful for initial exploration, the data analyst will need to construct a dictionary of bigrams, trigrams, and tetra-grams, collectively called n-grams. Bigrams are two word phrases, trigrams are three word phrases, and four-grams are four word phrases.

Here is an example of trigrams from the randomly sampled corpus. Recall that stop words had been removed so the phrases may look choppy. In the final dictionary, stop phrases and words of any length will be maintained.

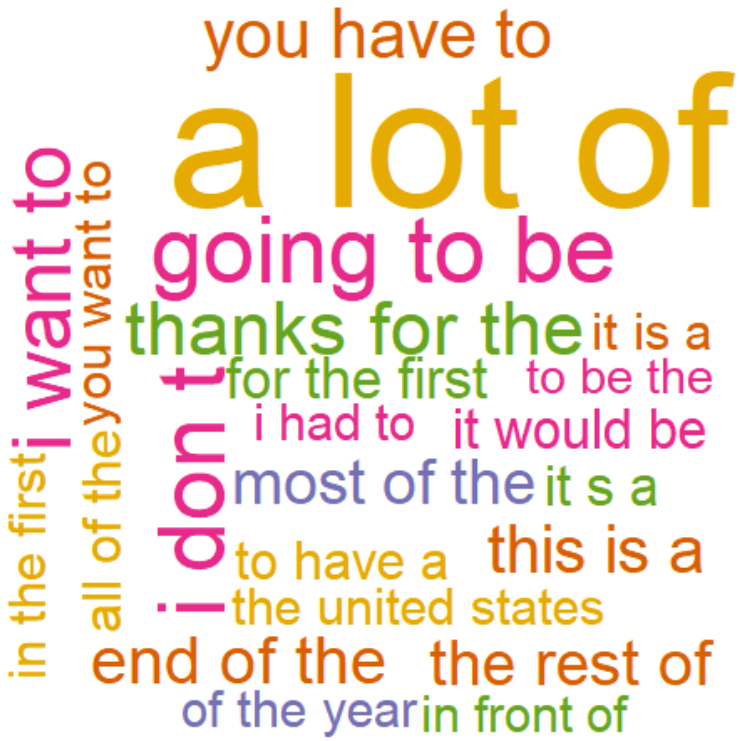
/ Trigrams

// Frequent Terms Trigrams

word	frequency
one of the	330
a lot of	284
the u s	166
to be a	152
as well as	147

word	frequency
the end of	139
going to be	137
i don t	134
out of the	129
some of the	124

// Wordcloud Trigrams



/ Predicting from N-grams

Each n-gram will be split, separating the last word from the previous words in the n-gram.

- bigrams will become unigram/unigram pairs
- trigrams will become bigram/unigram pairs
- four-grams will become trigram/unigram pairs

For each pair, the three most frequent occurrences will be stored in the dictionary. Here are the three most frequent trigrams for a bigram of “cant wait” from the randomly sampled corpus. These eleven trigrams would be split into bigram/unigram pairs and stored in the sample dictionary. Dictionaries will be built for the whole data set

```
freq.trigram %>% filter(str_detect(freq.trigram$word,"^cant wait"))
```

```
##              word frequency
## 1    cant wait to        47
```

```
## 2    cant wait for        16
## 3   cant wait till         7
## 4  cant wait until         3
## 5     cant wait my         2
## 6    cant wait any         1
## 7    cant wait but         1
## 8   cant wait good         1
## 9    cant wait ily         1
## 10    cant wait rt         1
```

// Application Logic

After the dictionaries have been established, an app will be developed allowing the user to enter text. The app will suggest the three most likely words to come next in the text for the text type, based on these rules.

- If the supplied text is greater than 2 words, take the last three words of the text and search the trigram/unigram pairs.
- If the supplied text is 2 words, take the two words and search the bigram/unigram pairs.
- If the supplied text is 1 word, search for that word in the unigram/unigram pairs.
- Suggest the three most frequent unigrams from the n-gram/unigram pair for either 1, 2, or 3 above.

GitHub i544c