Assy #2

Sebastian Schmeck

17 Mai 2018

Analysis Report

```
library(SnowballC)

## Warning: package 'SnowballC' was built under R version 3.4.1

library(rJava)

## Warning: package 'rJava' was built under R version 3.4.3

library(stringi)

## Warning: package 'stringi' was built under R version 3.4.4

library(knitr)

## Warning: package 'knitr' was built under R version 3.4.4

library(data.table)

## Warning: package 'data.table' was built under R version 3.4.4

library(RWeka)

## Warning: package 'RWeka' was built under R version 3.4.4

# Load the dataset.
# It will be easy for me (I can read, understand the text and spot
# different language words) to load and take a look at part of the English
dataset.
tweet_data <-("C:/Users/SchmeckS/Documents/3. Training/2017-04 Data
Science/10_capstone project/final/en_US/en_US.twitter.txt")
news_data <- ("C:/Users/SchmeckS/Documents/3. Training/2017-04 Data
Science/10_capstone project/final/en_US/en_US.news.txt")
blog_data <- ("C:/Users/SchmeckS/Documents/3. Training/2017-04 Data
Science/10_capstone project/final/en_US/en_US.blogs.txt")
tweet.en <- readLines(tweet_data, skipNul = TRUE)
news.en <- readLines(news_data, warn = FALSE)
blog.en <- readLines(blog_data, skipNul = TRUE)
# Statistics of the three files
# 1. size of three files
tweetSize <- file.info(tweet_data)$size/(1024*1024)
newsSize <- file.info(news_data)$size / (1024*1024)
blogSize <- file.info(blog_data)$size / (1024*1024)
corpusSize <- c(tweetSize, newsSize, blogSize)
# 2. Line counts of the three documents
```

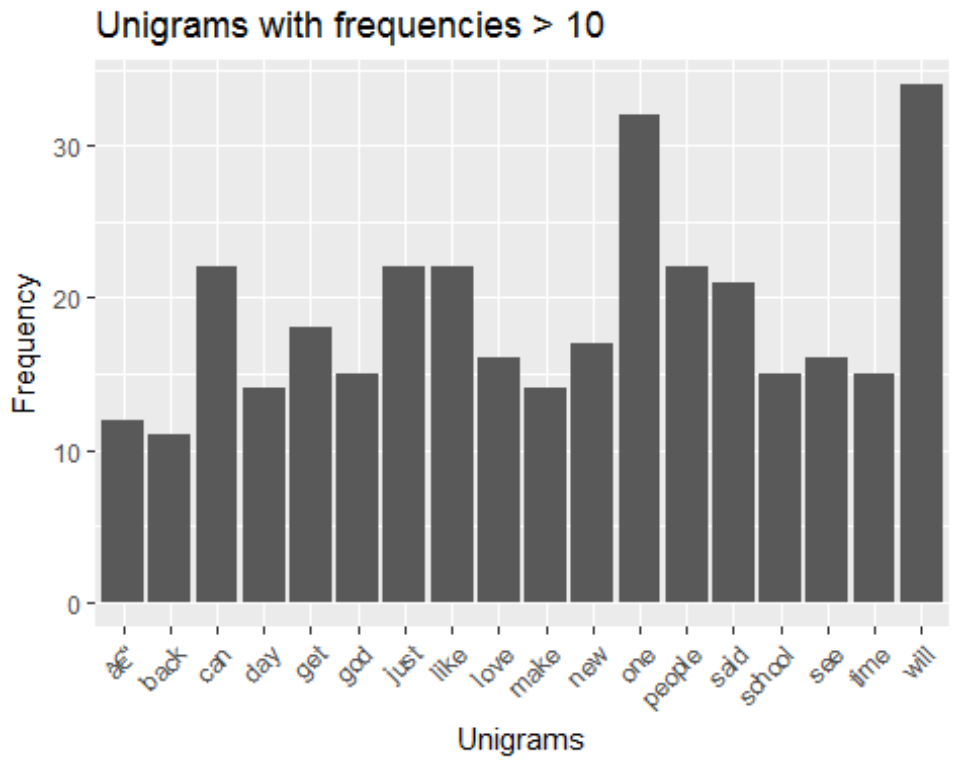
```
tweetLines <- length(tweet_data)
newsLines <- length(news_data)
blogLines <- length(blog_data)
corpusLC <- c(tweetLines, newsLines, blogLines)
#3. Word count
tweetWordCount <- sum(stri_count_words(tweet_data))
newsWordCount <- sum(stri_count_words(news_data))
blogWordCount <- sum(stri_count_words(blog_data))
corpusWC <- c(tweetWordCount, newsWordCount, blogWordCount)
# 4. words per line
tweet.wpl <- as.integer(tweetWordCount/tweetLines)
news.wpl <- as.integer(newsWordCount/newsLines)
blog.wpl <- as.integer(blogWordCount/blogLines)
Corpus.wpl <- c(tweet.wpl, news.wpl, blog.wpl)
# Display document statistics in table format.
corpus.text <- c("tweet.en", "news.en", "blog.en")
basicInfo <- data.table(corpus.text, corpusSize, corpusLC, corpusWC,
Corpus.wpl)
colnames <- c("Corpus Files", "File Size", "Line Count", "Word Count", "Words
per Line" )
kable(basicInfo, format="markdown", caption = "Info on three files")
```

corpus.text	corpusSize	corpusLC	corpusWC	Corpus.wpl
tweet.en	159.3641	1	15	15
news.en	196.2775	1	15	15
blog.en	200.4242	1	15	15

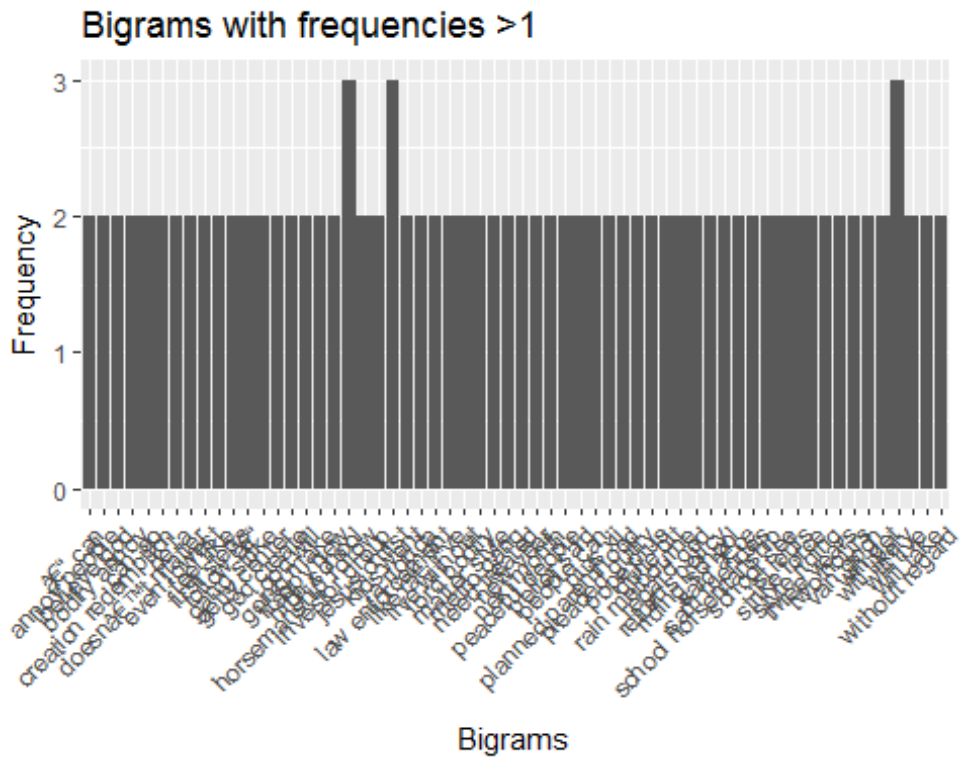
```
# Subsetting data.
# Since the files are too big, it would be difficult to use them for creating
an n-gram.
# Taking a small Sample allow us to select a representative subset of the
dataset which we can work with.
set.seed(85)
tweet_subset <- tweet.en[sample(1:length(tweet.en), 90)]
news_subset <- news.en[sample(1:length(news.en), 90)]
blog_subset <- blog.en[sample(1:length(blog.en), 90)]
# Data Cleaning.
# First, I combined my data subsets into one data. Then remove unwanted
charaters such as non english
# characters we do not want to predict
subsetData <- c(tweet_subset,news_subset,blog_subset) # combines data subsets
options(mc.cores = 1)
library(tm)

## Warning: package 'tm' was built under R version 3.4.4

## Loading required package: NLP

## Warning: package 'NLP' was built under R version 3.4.1

corpus <- VCorpus(VectorSource(subsetData))
toSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))
```

```
corpus <- tm_map(corpus, toSpace, "(f|ht)tp(s?)://(.*)[.][a-z]+")
corpus <- tm_map(corpus, toSpace, "@[^\\s]+")
corpus <- tm_map(corpus, tolower)
corpus <- tm_map(corpus, removeWords, stopwords("en"))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, PlainTextDocument)

# Creating N-Grams.
Tokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 1, max = 1))
unigram <- DocumentTermMatrix(corpus,
                              control = list(tokenize = Tokenizer))
BigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max =
2))
bigram <- DocumentTermMatrix(corpus,
                             control = list(tokenize = BigramTokenizer))
TrigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 3, max =
3))
trigram <- DocumentTermMatrix(corpus,
                              control = list(tokenize = TrigramTokenizer))
QuatgramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 4, max
= 4))
quatgram <- DocumentTermMatrix(corpus,
                               control = list(tokenize = QuatgramTokenizer))

# Examining NGrams frequencies
ufu <- sort(colSums(as.matrix(unigram)), decreasing=TRUE)
ufu.wordF <- data.frame(word=names(ufu), freq=ufu)

bfb <- sort(colSums(as.matrix(bigram)), decreasing=TRUE)
bfb.wordF <- data.frame(word=names(bfb), freq=bfb)

tft <- sort(colSums(as.matrix(trigram)), decreasing=TRUE)
tft.wordF <- data.frame(word=names(tft), freq=tft)

qfq <- sort(colSums(as.matrix(quatgram)), decreasing=TRUE)
qfq.wordF <- data.frame(word=names(qfq), freq= qfq)



library(ggplot2)

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':
##
##     annotate
```

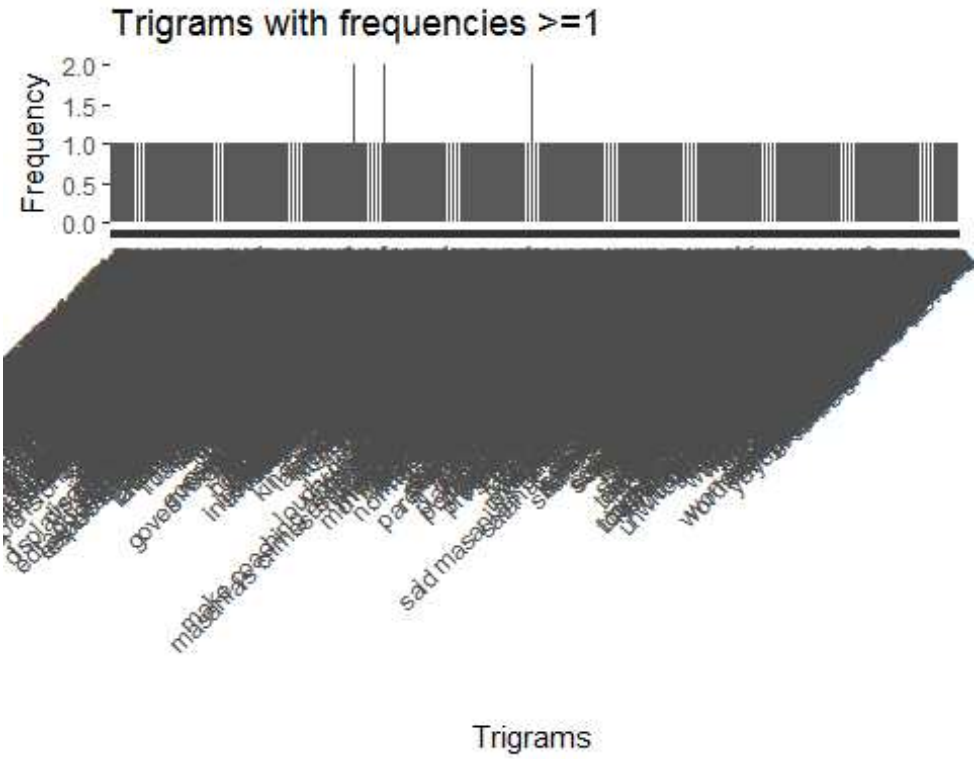
```
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:NLP':
##
##     annotate

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.4.3

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:data.table':
##
##     between, first, last
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

ufu.wordF %>%
    filter(freq > 10) %>%
    ggplot(aes(word,freq)) +
    geom_bar(stat="identity") +
    ggtitle("Unigrams with frequencies > 10") +
    xlab("Unigrams") + ylab("Frequency") +
    theme(axis.text.x=element_text(angle=45, hjust=1))

## Warning: package 'bindrcpp' was built under R version 3.4.3
```



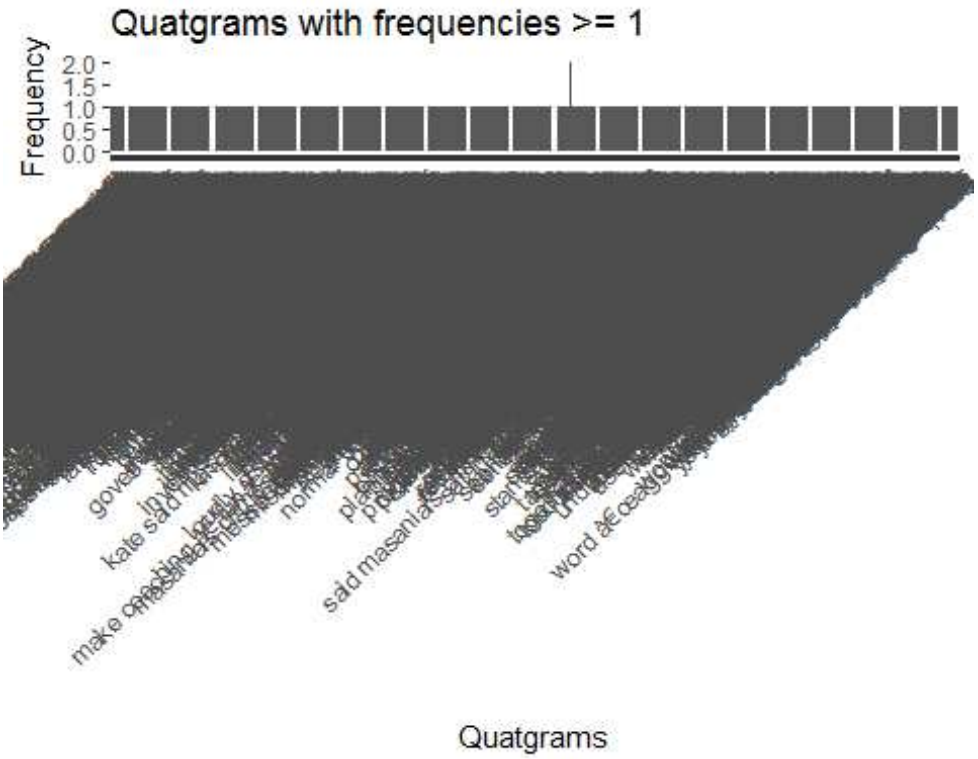

```
bfb.wordF %>%
    filter(freq >1) %>%
    ggplot(aes(word,freq)) +
    geom_bar(stat="identity") +
    ggtitle("Bigrams with frequencies >1 ") +
    xlab("Bigrams") + ylab("Frequency") +
    theme(axis.text.x=element_text(angle=45, hjust=1))
```




```
tft.wordF %>%
    filter(freq >=1) %>%
    ggplot(aes(word,freq)) +
    geom_bar(stat="identity") +
    ggtitle("Trigrams with frequencies >=1") +
    xlab("Trigrams") + ylab("Frequency") +
    theme(axis.text.x=element_text(angle=45, hjust=1))
```




```
qfq.wordF %>%
    filter(freq >=1) %>%
    ggplot(aes(word,freq)) +
    geom_bar(stat="identity") +
    ggtitle("Quatgrams with frequencies >= 1") +
    xlab("Quatgrams") + ylab("Frequency") +
    theme(axis.text.x=element_text(angle=45, hjust=1))
```


Quatgrams with frequencies >= 1
Frequency
2.0
1.5
1.0
0.5
0.0
Quatgrams