Capstone: Exploratory Analysis

Javier Angoy

Jan 8th, 2018

INTRODUCTION

Around the world, people are spending an increasing amount of time on their mobile devices for email, social networking, banking and a whole range of other activities. But typing on mobile devices can be a serious pain. A smart keyboard that makes it easier for people to type on their mobile devices. One cornerstone of the smart keyboards are predictive text models. In this capstone we will work on understanding and building predictive text models.

To start building a predictive model for text it is very important to understand the distribution and relationship between the words, tokens, and phrases in the text. The goal of this task is to understand the basic relationships in the data and prepare to build the first linguistic models.

The second goal is to build a simple model for the relationship between words. This is the first step in building a predictive text mining application.

This report describes in plain language, plots a preliminary exploratory analysis of the capstone data set.

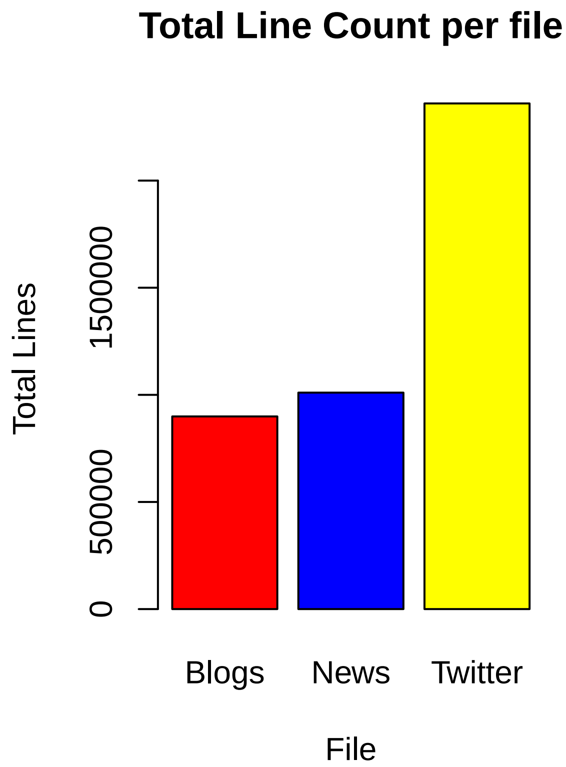
THE DATA

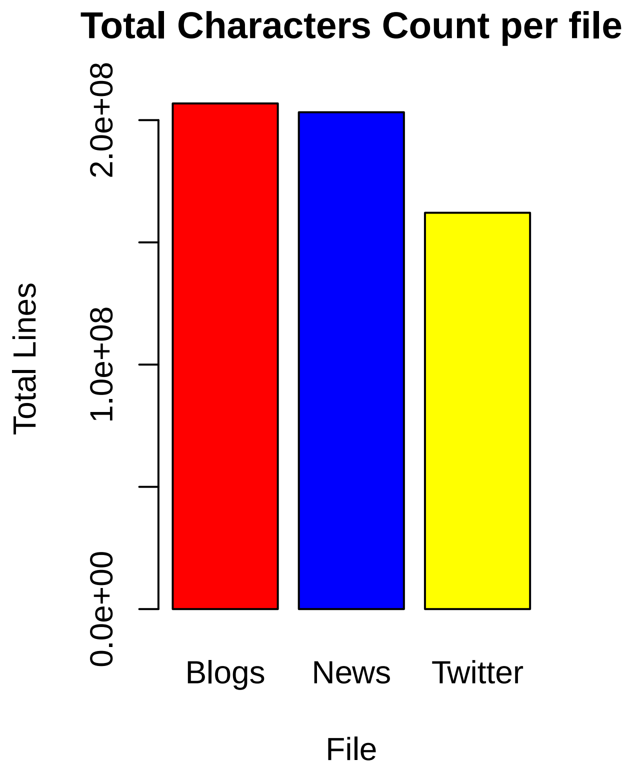
Text comes in three separated files: blog, news and twitter for each language.

Summary of source files

File Name	N. Lines	N. Characters	Size
en_US.blogs.txt	899288	206824505	200.4 MiB
en_US.news.txt	1010242	203223159	196.3 MiB
en_US.twitter.txt	2360148	162096241	159.4 MiB

```
##    user  system elapsed
##  16.888   0.288  31.893
```





So we see that blogs have the least number of lines while the twitter data have the most. On the other side, blogs have more characters (and blog file size is bigger too).

THE NEW CORPUS

Being the original files too bulky for our analysis, due to memory and processing time constrains, we will take a 10% sample from each file and gather the docs into a corpus. As our data analysis will be done through the English text, we choose only the "en_US" files and proceed. After that, a summary with the 10 first documents (lines) of the corpus is obtained.

```
## Corpus consisting of 1067419 documents, showing 10 documents:
##
##    Text Types Tokens Sentences
##   text1    60     79         1
##   text2    31     40         1
##   text3    22     26         1
##   text4    41     51         1
##   text5    71     95         1
##   text6    27     32         1
##   text7   121    200         1
##   text8     3      3         1
##   text9    28     30         1
##  text10    23     23         1
##
## Source:  /media/Windows/Users/kakis/Documents/DOCS/Dropbox/Data Scientist Coursera/Projects/Capstone/
## Created: Fri Feb  9 12:29:58 2018
## Notes:
```

```
##    user  system elapsed
##  30.016   0.708  47.517
```

List of top 25 ngrams (unigrams, duograms and trigrams)

the	477,065	of the	43,368	one of the	3,548
to	276,818	in the	41,227	a lot of	3,049
and	241,589	to the	21,308	thanks for the	2,335
a	239,134	for the	19,901	to be a	1,788
of	201,333	on the	19,602	going to be	1,695
in	165,627	to be	16,193	the end of	1,535
i	164,437	at the	14,159	i want to	1,517
for	110,075	and the	12,656	out of the	1,511
is	107,647	in a	12,008	as well as	1,447
that	104,747	with the	10,492	it was a	1,360
you	93,934	is a	10,106	some of the	1,357
it	92,478	it was	9,432	be able to	1,328
on	82,636	for a	9,407	part of the	1,253
with	71,618	from the	8,769	the rest of	1,171
was	62,177	i have	8,642	i have a	1,165
my	60,587	i was	8,514	looking forward to	1,127
at	57,234	it is	8,282	i have to	1,106
be	55,154	with a	8,212	i don't know	1,092
this	54,338	will be	8,110	thank you for	1,085
have	53,405	of a	8,072	the first time	1,019
are	49,145	and i	8,001	is going to	997
as	48,861	going to	7,889	a couple of	984
but	48,694	i am	7,646	this is a	963
he	43,500	have a	7,435	i'm going to	955
we	41,836	is the	7,377	you want to	949

So we have a corpus with 1067419 documents (lines).

Data cleaning

Unlike other NLP methods that avoid some words like offensive words and empty words, our aim is to get the richest possible model. Therefore, we assume that the results will have more accuracy if the model takes into account all kind of vocabulary. To tidy data we will remove numbers (standalone), punctuation marks and other symbols, separators, twitter special chars (@ and #), hyphens and url addresses, as we will focus on words. Foreign language words or phrases will not be handled in any way.

Basic Statistics to summarize features

We perform a summary of the main characteristics we can

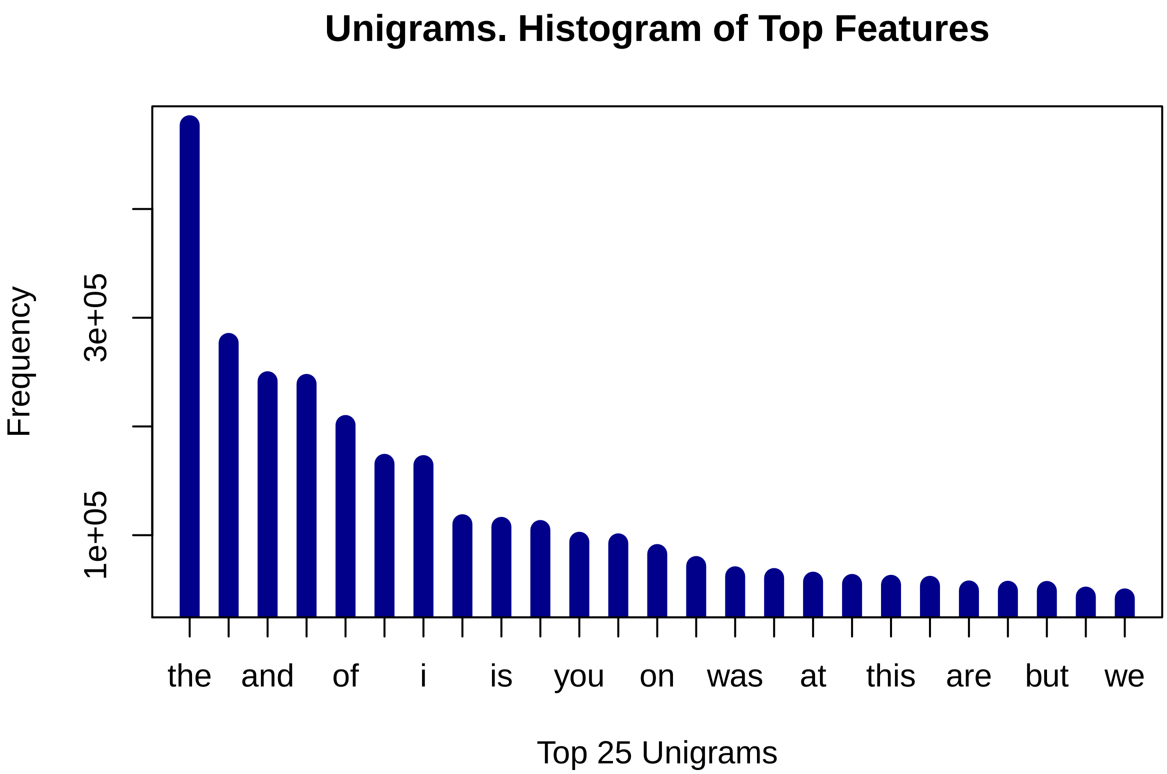
Feature Statistics

unigrams	bigrams	trigrams
178796	2622252	6239837

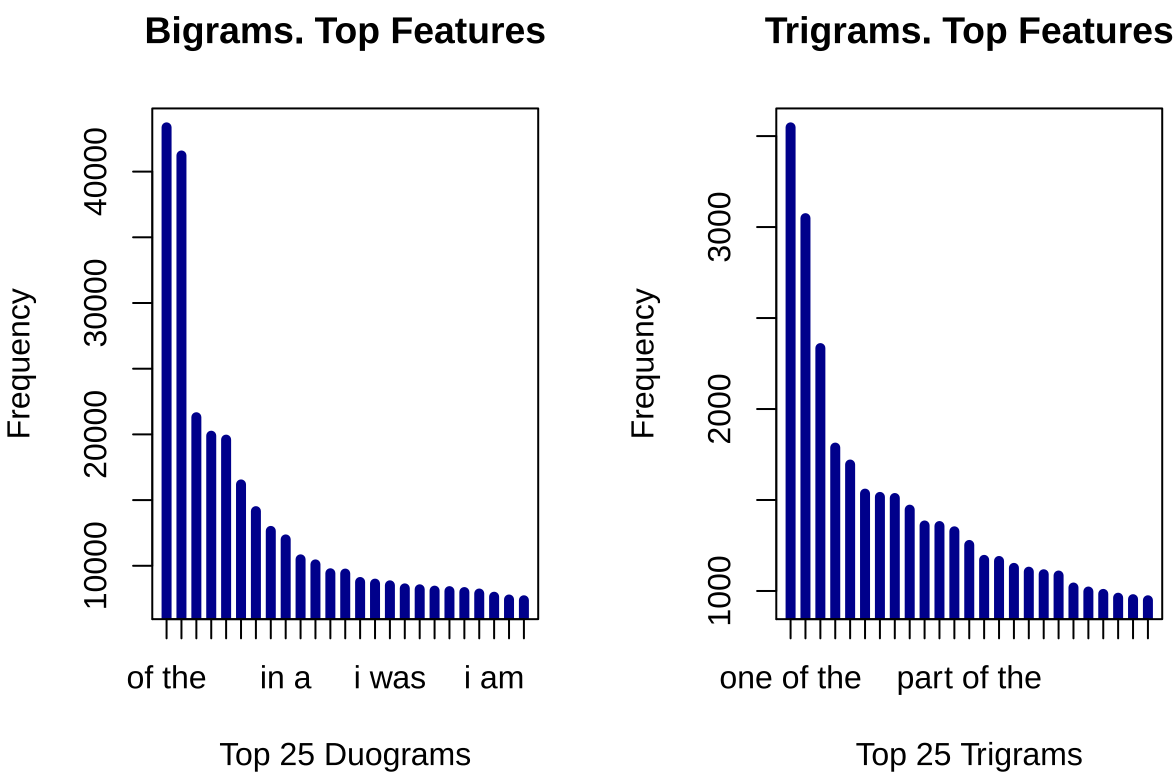
PLOTS

Top Features

Distributions of word frequencies



Frequencies of 2-grams and 3-grams in the dataset



Coverage 50% - 90%

Now we estimate the number of unique words that we need in our frequency sorted dictionary to cover 50% of all word instances in the language.

```
## [1] 143
```

And for 90%

```
## [1] 7423
```

So we see that we need 143 unique words to get an accuracy of 50%, and that 7423 will give to us a 90% accuracy.

CONCLUSIONS

As we see a small number of words represents the bigger part of the corpus. Therefore, we can say that our model will not take much advantage of the most seldom features. Ignoring those less common features might make our model a bit resource efficient.

NEXT STEPS

Next we will try to build a predictive model that explores the relationship between words in the more effective way we can. So we could try to do the same analysis up to bigger n-grams, may be 5. Memory usage and processing speed will be important tradeoffs however.

APPENDIX. THE CODE

Getting the data

```
# Download zip file and decompress
if(!file.exists("./Project/Coursera-SwiftKey.zip")){
        download.file(url = "https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey
                      destfile= "./Coursera-SwiftKey.zip",
                      method = "curl")
    unzip("./Project/Coursera-SwiftKey.zip", overwrite = FALSE, exdir = "./Project")
    }
```

Exploring the files

```
ptm <- proc.time()

dirFiles <- paste0(getwd(),'/Project/final/en_US/')

# List all files in directory and extracts information
readF <- function(dirOpen){
    co <- 0 #just a counter
    finfo=data.frame(fname=character(), flen=integer(),
                     fchars=integer(),fsize=character(),stringsAsFactors = FALSE)
    listf <- list.files(path = dirOpen, full.names = T)
    #loop list of files, extract info
    parallel::mclapply(listf,function(fileOpen) {
        co <- 1 + co
        con <- file(fileOpen, "r", blocking = FALSE)
        lines <- readLines(con, skipNul = T)
        finfo[co,1] <- as.character(basename(fileOpen))
        finfo[co,2] <- length(lines)
        finfo[co,3] <- sum(nchar(lines))
        finfo[co,4] <- gdata::humanReadable(file.info(fileOpen)$size)
        close(con)
        return(finfo)
        })
}
info_table <- readF(dirFiles)
info_table <- rbind(info_table[[1]],info_table[[2]],info_table[[3]])
names(info_table)<- c("File Name","N. Lines","N. Characters","Size")
knitr::kable(info_table, caption = "Summary of source files")
proc.time() - ptm
```

```
par(mfrow = c(1, 2))
barplot(info_table$`N. Lines`,
        names.arg = c("Blogs","News","Twitter"),
        col=c("red","blue","yellow"),
        xlab = 'File', ylab = 'Total Lines',
        main = "Total Line Count per file")
barplot(info_table$`N. Characters`,
        names.arg = c("Blogs","News","Twitter"),
        col=c("red","blue","yellow"),
```

```
        xlab = 'File', ylab = 'Total Lines',
        main = "Total Characters Count per file")
```

Exploratory Analysis

```
dirFiles <- paste0(getwd(),'/Project/final/')
copyLines <- function(dirOpen){
    dir_US <- paste0(dirOpen,'en_US/')
    fileConn <-paste0(dirOpen,'en_US.txt')
    listf <- list.files(path = dir_US, full.names = T)
    if (file.exists(fileConn)){file.remove(fileConn)}
    #loop list of files, open and copy 25% sample of lines to corpus
    parallel::mclapply(listf,function(fileOpen) {
        con <- file(fileOpen, "r", blocking = FALSE)
        lines <- readLines(con, skipNul = T)
        lines_copy <- sample(lines, size=length(lines)*0.25, replace=F)
        text <- tolower(lines_copy) #all text to lower
        close(con)
        return(text)
        })
}
set.seed(100)
mycorpus <- quanteda::corpus(unlist(copyLines(dirFiles)))
summary(mycorpus,10)
proc.time() - ptm
```

Tokenize the texts, create n-grams

```
tok <- tokens(mycorpus, verbose = TRUE,
       remove_numbers = TRUE, remove_punct = TRUE,
       remove_symbols = TRUE, remove_separators = TRUE,
       remove_twitter = TRUE, remove_hyphens = TRUE,
       remove_url = TRUE)
tokens_1 <- tokens(tok, ngrams = 1, concatenator = " ")
tokens_2 <- tokens(tok, ngrams = 2, concatenator = " ")
tokens_3 <- tokens(tok, ngrams = 3, concatenator = " ")
```

```
unigrams <- dfm(tokens_1, verbose = TRUE)
duograms <- dfm(tokens_2, verbose = TRUE)
trigrams <- dfm(tokens_3, verbose = TRUE)
```

```
df <- cbind(nfeature(unigrams),nfeature(duograms),nfeature(trigrams))
colnames(df) <-c("unigrams","bigrams","trigrams")
knitr::kable(df,style="html", caption = "Feature Statistics")
#summary
```

```
ngram1 <- as.matrix(quanteda::topfeatures(unigrams, 25),dimnames=c("1gram","freq"))
ngram2 <- as.matrix(quanteda::topfeatures(duograms, 25),dimnames=c("2gram","freq"))
ngram3 <- as.matrix(quanteda::topfeatures(trigrams, 25),dimnames=c("3gram","freq"))

knitr::kable(list(ngram1,ngram2,ngram3), caption = "List of top 25 ngrams (unigrams, duograms and trigra
```

```
#, padding = 10
```

Plots

Coverage

```
no_tokens <- sum(colSums(unigrams)) #total tokens
features_desc <- sort(colSums(unigrams), decreasing = TRUE) #sort features +-

rel_freq <- features_desc / no_tokens * 100 #calculates relative frequency vector
acum_rel_freq <- cumsum(rel_freq) #accumulated relative frequency
fcover_50 <- Position(function(x) x >= 50, acum_rel_freq) #features to cover 50%
print(fcover_50)
```

```
fcover_90 <- Position(function(x) x >= 90, acum_rel_freq) #features to cover 90%
print(fcover_90)
```