Milestone Report

Course: Data Science Capstone
by Jeff Leek, PhD, Roger D. Peng, PhD, Brian Caffo, PhD

Author: Eduardo B. Díez — November 2014

Preamble

This concise milestone report is to explain —in a way that would be understandable to a non-data scientist manager— only the major features of the data and just to display that we've gotten used to work with the data from a corpus called HC Corpora and also that we are on track to create a prediction algorithm. The format of the report is html and it is uploaded as a R Pubs document.

It is interesting to get feedback about the plans for creating a prediction algorithm and Shiny app and so, we will explain it after the exploratory analysis of the data, looking forward for those invaluable comments from peers.

Sources files: Lines, Words, Chars and more

Three are the files of interest based on information from twitter, news and blogs. All of them in English.

- en_US.twitter.txt
- en_US.news.txt
- en_US.blogs.txt

The next paragraph —as an example— shows one piece of chunk with the code that load one file of interest to obtains a set of values from this file.

```
# set the data dir

setwd("D:/Cursos/Hopkin/Capstone/Data")

# load one file

con <- file("./twitter/en_US.twitter.txt", "rb")

twitter <- readLines(con, encoding = "UTF-8")

close(con)

# got an array with several information from the
loaded file

library("stringi")

twitter.wn<-stri_stats_latex(twitter)

# also got the number of lines in twitter

twitter.ln<-length(twitter)

names(twitter.ln) <- "Lines"

# calculate the average of words by line

TwitterAvgWordsPerLine <- twitter.wn[4] / twitter.ln

names(TwitterAvgWordsPerLine) <- "Ratio W/L"
```

We can now show the results of the previous code but, we better work the same way as before with the rest of files —this time hiding the chunk codes— and we are going to take up again all the figures in the next table:

Table 1

File	Chars	Words	Lines	Ratio W/L
TWITTER	125570616	30451128	2360148	12.9
NEWS	162227130	34494539	1010242	34.14
BLOGS	162464653	37570839	899288	41.78

Summary of some values of the data sets

The Corpus: Ploting the Words

Once loaded the files is time to clean each one and keep just a fraction of the original in order to fit on the available memory of the computational equipment to work with it and we are going to mean this as our training files.

The next tasks to do are:

- Obtain a training sets
- Clean each one of the training files
- Define a corpus from the training set
- Construct the documents term matrix from the corpus and remove sparse terms
- Extract the word frequencies
- Plot the word frequencies

We've set the chunk option ***echo=FALSE*** in order to avoid boring code in this report and we show below only the results of the hidden code; an histogram and a colorful wordcloud plot from a training set of 1000 document from each of the files of interest.

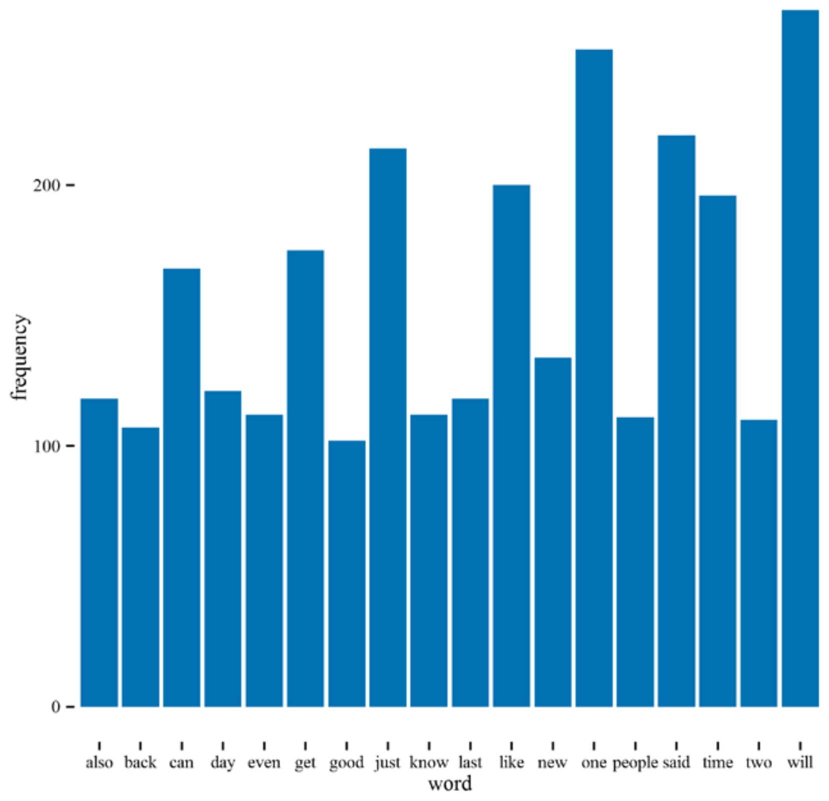


Figure 1. Histogram plot of the words in our training corpus with more than 100 occurrences.

It’s time to show a nowadays fashion plot.

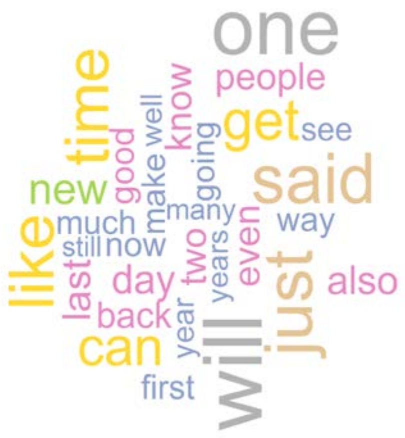


Figure 2. Wordcloud plot with basically the same information than the previous histogram plot, but this time more friendly, where the size and the color of the word are proportional to the occurrence of each one.

Recapitulation

So far we have this concise style HTML report uploaded as a R Pubs document describing some exploratory analysis and basic summaries of the three files as word and line counts and basic data tables. Also show a couple of basic plots, such as a histogram and a wordcloud to illustrate features of the data as the frequencies of words.

That is why we are confident we have achieved the goal of this milestone and we just need to expose our planning to implement a predictive algorithm that meets the requirements of the capstone project.

The Future: Predictive Algorithm

The plan is to build a predictive text algorithm and a shiny application based on frequency of occurrence of n-grams. Therefore, our procedure has to be extended to looking at the frequency of occurrence of 2-gram, 3-gram and 4-gram, choosing the prediction from the largest n-gram available in each case.

Let's show a R code example for 3-gram from our three training files —each with 1000 docs only.

```
corpus <- c(twitter, news, blogs)

library("tau")

trigram <- textcnt(corpus, n = 3, method="string")
```

We'll do it for two, three and four grams and show the first five results for each gram in table 2.

Table 2

N-gram	**Term**	**Occurrence**
2-gram		
	high school	22
	new york	20
	last year	18
	last week	16
	years ago	16
3-gram		
	chuck norris means	3
	happy mothers day	3
	just make sure	3
	new york city	3
	sales taxes imposed	3
4-gram		
	christies international real estate	2
	clackamas county chamber commerce	2
	dramatic performances math problems	2
	keri tarr cat detective	2
	let whole world waits	2

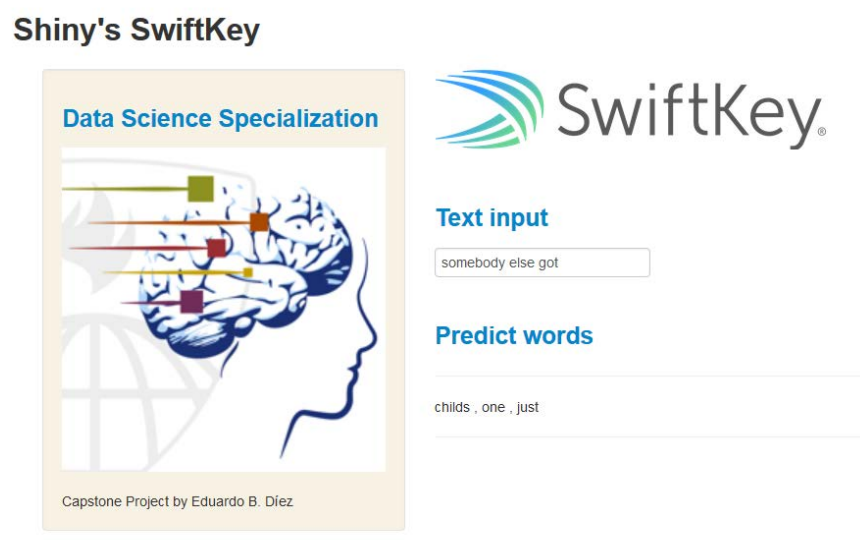
Example of gram from our training data

Also, to reduce bias, the final algorithm will make use of a training sets bigger than 1000 samples like we used in the elaboration of this report. The law of Heaps helps us to guess that we'll need a corpus with 1,000,000 words to get a vocabulary size of 20,000 words, the latter is a good point that we stimate for a lexical saturation of the working corpora. By using the results in table 1 we need roughly 11,500 lines from each file to reach the one million volumen that should give us a vocabulary of around 20,000 different words. This will be a good starting point.

Another inclusions should be a filter of profanity words and the restoration of stop-words like ***I'm***, for this report the latter was used without discrimination.

Finally, the Shiny app should be a simple front-end with a text box input and the output would be an array of the three more likely next words.

The following figure shows one prototype of our proposed interface for the shiny application that is already loaded —but not functional yet— in https://turpial.shinyapps.io/dsscapstone/. The application only needs to incorporate the definitive R code for the model that we are still deciding the final one —surely we'll implement a probabilistic HMM trained with conditional probabilities or Good-Turing.


Shiny's SwiftKey
Data Science Specialization
Capstone Project by Eduardo B. Díez
SwiftKey
Text input
Predict words

somebody else got

childs , one , just

Figure 3. An early model built to test the process of the shiny app.

In order to choose the model to be implemented, we are going to measure how well the app model *fits* a test data corpora sampled randomly from the given corpus and we'll measure on it the perplexity parameter and we look forward to obtain values around (1000, 200, 100) for the unigram, bigram and trigram respectively. But there is a tradeoff, since we want our test set to be as large as possible because a small test set may be accidentally unrepresentative, and on the other hand, we want as much training data as possible. At the minimum, we would want to pick the smallest test set that gives us enough statistical power to measure a statistically significant difference between two potential models.

About this Document

The original markdown file MilestoneReport.Rmd with all the R code can be retrieve in order to reproduce and generate this report or to study the R code that we used to clean, filter and explore the data. The HC Corpora should be obtained separately.

All analyses were performed using *R version 3.1.2 (2014-10-31)* and *RStudio 0.98.1091* as IDE, with the default base packages *splines, grid, stats, graphics, grDevices, utils, datasets, methods, base*. Also have been invoked explicitly *ggplot2, ggthemes, Gmisc, Grmd, knitr, stringi, tau, tm, wordcloud* and additionally were loaded *acepack, cluster, colorspace, digest, evaluate, foreign, formatR, gtable, htmltools, labeling, latticeExtra, MASS, munsell, nnet, parallel, plyr, proto, Rcpp, reshape2, rmarkdown, rpart, scales, slam, SnowballC, sp, stringr, tools, XML, yaml* and finally the R code chunks where formated with Pretty-R