Capstone Milestone Report

Carlos Alberto Guevara Diez

March 2016

Summary

The goal of this project is just to display that we've gotten used to working with the Swiftkey data and that we're on track to create our prediction algorithm. The motivation for this project is to load data from text sources, create basic reports of summary statistics, report interesting findings and get some insights of the next steps to follow in order to create a shiny app that predicts text.

1. Loading the data

The first thing I need to do is to load the data into R, once that is in memory I´m going to perform some counting on the objects, size of each one of them and frequency analysis, with this I´m goint to undersand the information that I´m handling. For demostrating purposes I´m going to show only pieces of code that performs the operations behind the data, nevertheless if the reader wants to dig into de code, the .Rmd file can be found in my Github page

```
#Read blogs file
cname<-file.path("~", "Data Science Specialization", "10 Data Science Capstone", "Capstone", "Dataset",
con <- file(paste(cname,"/en_US.blogs.txt",sep=""), open = "rb")
blogs <- readLines(con, encoding="latin1")
close(con)
#Read news file
con <- file(paste(cname,"/en_US.news.txt",sep=""), open = "rb")
news <- readLines(con, encoding="latin1")
close(con)
#Read twitter file
con <- file(paste(cname,"/en_US.twitter.txt",sep=""), open = "rb")
tweets <- readLines(con, encoding="latin1")
close(con)
```

2. Analyzing the size of files and number of words

Once the data have been loaded into R it´s time to build a basic statistical report that shows the main characteristics of the data.

File	Object.size..MB.	Number.of.lines	Number.of.characters
blogs	260.56	899,288	208,361,438
news	261.76	1,010,242	203,791,405
tweets	316.04	2,360,148	162,384,825

Now its time to look at the number of words and unique words in every file

File	Number.of.words	Number.of.Unique.Words
blogs	38,153,767	342,482

File	Number.of.words	Number.of.Unique.Words
news	35,016,742	294,261
tweets	30,195,678	380,765

With this information I can say that the largest dataset it´s the one that comes from twitter, in size represents the 37 % of the entire data set, nevertheless is the one which have less words.

The entire data set (the 3 files) contains more than 4 million of lines, 574 million of characters and 103 million of words, that is a big amount of data to handle.

3. Creating sample of the data, cleaning and profanity filtering

As it can be seen in the last section, the dataset is too big and also a large amount of computational resources are needed to process it, thats why to make further analysis easier and faster, I've created a random sample of the data by selecting 5000 rows from each file. With this samples I´m creating a new dataset that contains random information (using binomial distribution) that will be used as the corpus for the rest of the analysis. A corpus (plural corpora) or text corpus is a large and structured set of texts (nowadays usually electronically stored and processed). They are used to do statistical analysis and hypothesis testing, checking occurrences or validating linguistic rules within a specific language territory.

```
set.seed(10)
sample=5000

# Creating sample files of 5000 lines for each dataset
blogsSample = blogs[rbinom(sample, length(blogs),0.5)]
newsSample = news[rbinom(sample, length(news),0.5)]
tweetSample = tweets[rbinom(sample, length(tweets),0.5)]

# Merging the sample datasets into one
sampleDS = c(blogsSample, newsSample, tweetSample)
```

Now it´s time to clean the sampled dataset and remove profanities, the cleaning proces includes:

- Remove special characters
- Remove punctuations
- Remove numbers
- Remove extra whitespace
- Convert to lowercase
- Remove stop words
- Remove profanity words.

The list for profanity filtering was obtained from http://www.cs.cmu.edu/~biglou/resources/bad-words.txt that contaions around 1400 bad words. Finally the new corpus with the cleaned and filtered data is created.

```
#Function created to transform and remove undesired features from the dataset.
cleaningFunction = function(x) {
  x <- gsub("/|\\||@|#", "", x)  # Remove special characters
  x <- removePunctuation(x)      # Remove puntuations
  x <- removeNumbers(x)          # Remove numbers
  x <- stripWhitespace(x)        # Remove extra whitespace
```

```
  x <- tolower(x)                 # Convert to lowercase
  x <- removeWords(x,tm::stopwords(kind="en"))  # Remove stop words
  x <- removeWords(x,badWords)          # Remove profanity words
  x <- stemDocument(x)            # Removing commond word endings
  return(unlist(x))
}

sampleDS <- cleaningFunction(sampleDS)
#Print a summary of the transformed dataset
data.table("Lines" = length(sampleDS), "Words" = sum(nchar(sampleDS)))
```

```
##    Lines   Words
## 1: 15000 1850359
```

```
#Corpus created for further analysis
cleanCrps <- corpus(sampleDS)
```

4. Performing basic words frequency analysis on Sampled Corpus

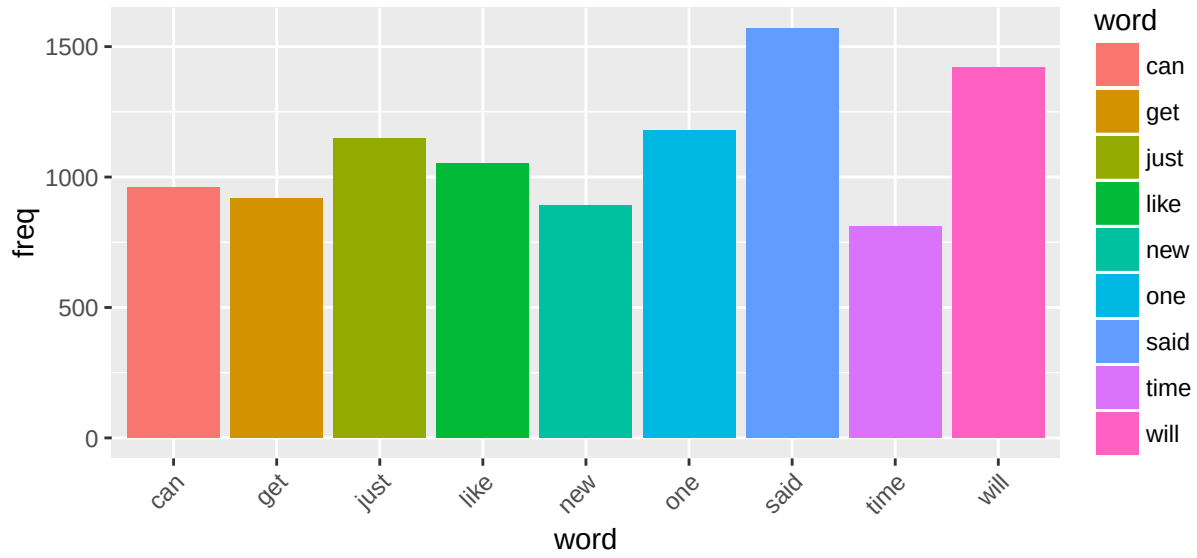
Now that I have a clean and filtered sampled corpus it´s time to do some frequency analysis to understand its structure, to do that I'm using a Document Term Matrix which is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms.

```
cleanCrps2 <- Corpus(VectorSource(sampleDS))
dtm <- DocumentTermMatrix(cleanCrps2)
```

The first analysis is to find the 20 most frequently used words:

```
##  [1] "also"   "back"   "can"    "day"    "first"  "get"    "going"
##  [8] "good"   "just"   "know"   "last"   "like"   "make"   "much"
## [15] "new"    "now"    "one"    "people" "said"   "think"
```

Now I'm showing a plot for words which frequencies in the corpus are bigger than 800



With this I´m ending the basic exploratory analyses and start with the real core of text processing.

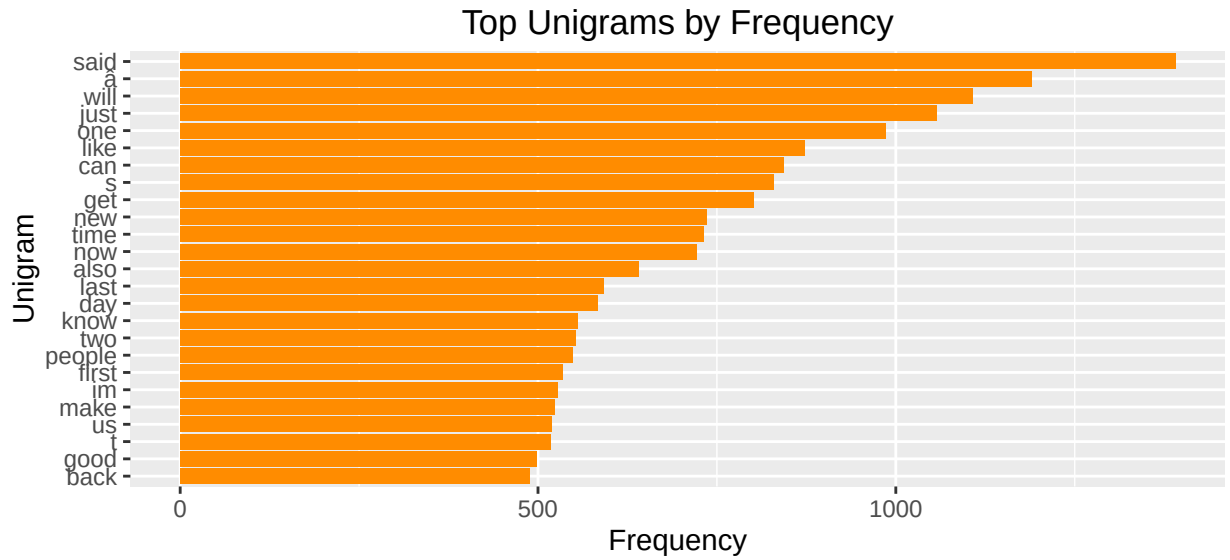
5. N-Grams, Tokenization and Plots

One of the most important concepts in Natural Language Processing (NLP) are the use of N-Gram and tokenization to create prediction text models, in the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus. When the items are words, n-grams may also be called shingles. An n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram" (or, less commonly, a "digram"); size 3 is a "trigram". Larger sizes are sometimes referred to by the value of n, e.g., "four-gram", "five-gram", and so on.

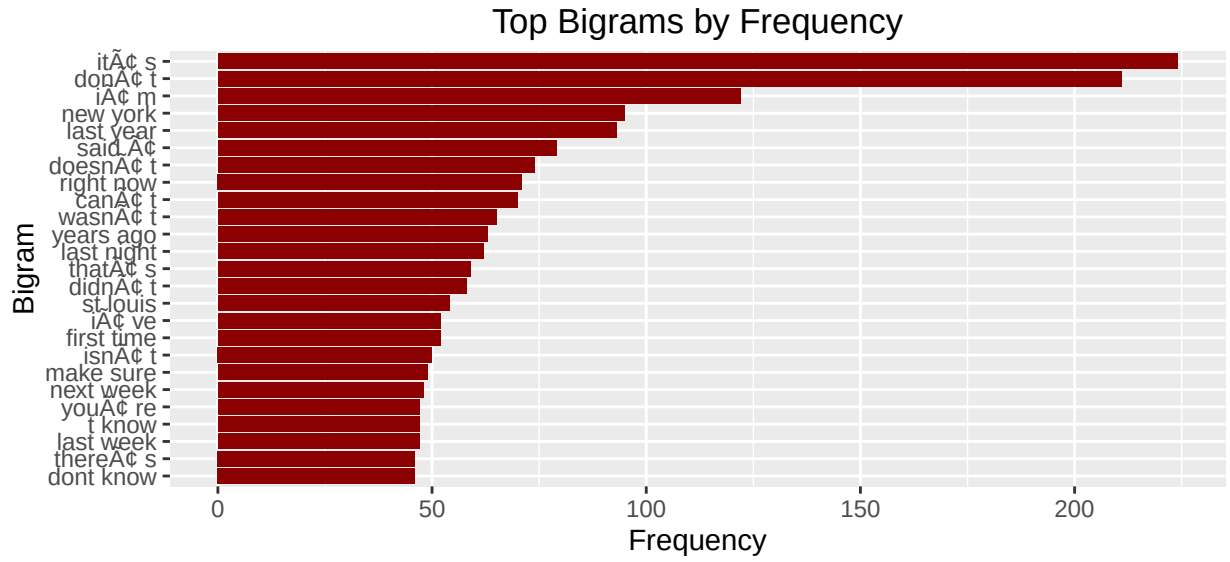
Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing such as parsing or text mining. Tokenization is useful both in linguistics (where it is a form of text segmentation), and in computer science, where it forms part of lexical analysis.

With this two main concepts, I have built unigrams, bigrams and trigrams that will be used in future stages of this project to build the prediction model and later the shiny application that is going to be used as the interface with the user, next I'm showing exploratory graphics on the n-grams.

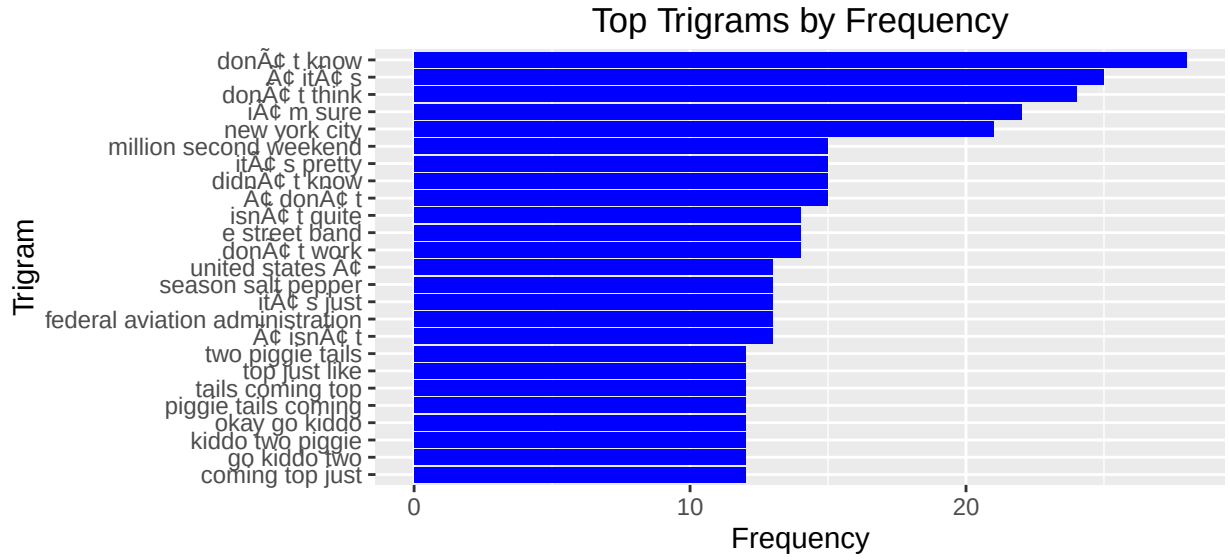
Top Unigrams



Top Bigrams



Top Trigrams



6. Conclusions and future steps

Text analysis is fascinating, the theories and concepts involved in it are amazing and R provides a good set of libraries to deal with it, nevertheless, when you perform this type of analysis you have to be careful because the computational resources involved are huge.

Thru this analysis I have demonstrated the size of the objects handled, even I have to get a subset of the original data to move on more complicated tasks as the creation of n-grams. Despite of the sampling my machine had problems to deal with the calculations, thats why when you decide to create this kind of applications you have to be careful in the selection of data, keep monitored your computational resources and even think in compressing techniques.

I really have enyojed building this report and I hope that you enjoy reading it too, also I have understand that I need to explore more on the techniques to get to the final result, the following steps to follow are:

- Dig in on papers about Natural Language Processing (NLP), creation of N-Grams and Markov Chains, a deeper study of the tm, quantena and Rweka packages to optimize my previous analyses.
- Create an initial prediction model that helps me to understand deeper the concepts involved in text prediction.
- Create a Shiny application that will consume the NLP model and a user friendly presentation of its use.

I really understand that what I´ve presented is only the beginning and there´s a lot of work coming,thank you for reading!.