Data Science Capstone - Milestone Report

CChevalier

December 2015

Contents

Introduction **1**

Pre-Processing **2**

The Original Text Dataset 2

Pre-Processing Methodology 2

Basic Summary 2

Exploratory Analysis Methodology **3**

Definitions 3

Sample corpus 3

Methodology 4

Useful TDM Post-Processing functions 4

Exploratory Analysis: Unigrams and stop words **5**

Unigrams in the cleaned sample corpus (excluding removing english stopwords) 5

Unigrams in the cleaned sample corpus (including removing english stopwords) 7

Future Plans **9**

Conclusion **10**

Introduction

This milestone report is written as part of the capstone project of the Data Science Specialization from John Hopkins University / Coursera. The final aim of this project is to produce a text prediction app using different Natural Language Processing (NLP) techniques. The current report is an intermediate report presenting pre-processing and some elementary exploratory analysis off the text dataset provided in the frame of this project.

Pre-Processing

The Original Text Dataset

The provided text dataset is a collection of text from different internet resources (blogs, news feeds and twitter) in four different languages. The current report deals only with the english US version as I have no knowledge of the other three propsed languages. As will be soon later the original dataset is rather large and analysis can only be conducted on a reduced dataset that will be produced during pre-processing.

Pre-Processing Methodology

The pre-processing methodology is the following one:
- Load original dataset (3 en_US files)
- Derive basic summary for each file (file size on disc, number of words and number of lines)
- Resample data to a given fraction of the original one for a more convenient and practical exploratory analysis

Below are reported the most important functions used for the pre-processing part:

```
# getFileStats function
getFileStats <- function(filename) {
  filesize <- file.info(filename)$size / 1024^2
  return(round(filesize))
}

# resample function
resample <- function(data, sampleRatio, sampleFilename) {
  dataSample <- sample(data, round(sampleRatio/100 * length(data)))
  write(dataSample, sampleFilename)
  return(dataSample)
}

# Pre-Process function
preprocess <- function(fileType, dataFolder, LOCALE, dataSampleFolder, sampleRatio) {
  typeFilename <- setDataFilename(fileType, dataFolder, LOCALE)
  typeData <- readDataFile(typeFilename)

  # Basic summary
  print(paste("File:", typeFilename))
  print(paste("     ", getFileStats(typeFilename), "Mb"))
  print(paste("     ", length(unlist(strsplit(typeData, "\\s+", perl = T))), "words"))
  print(paste("     ", length(typeData), "lines"))

  # Resampling for more convenient future exploratory analysis
  typeSampleFilename <- setDataSampleFilename(fileType, sampleRatio, dataSampleFolder, LOCALE)
  typeSampleData <- resample(typeData, sampleRatio, typeSampleFilename)

  return()
}
```

Basic Summary

Thi section presents basic summary for each original file as derived during the pre-processing stage. As we can see the original data files are rather big and cannot really be handled with common personal computer

resources (at least the ones I have at the moment) in a reasonable time.

```
# Current Analysis Settings
LOCALE <- "en_US"
dataFolder <- "./data"
dataSampleFolder <- "./data-samples"
sampleRatio <- 1

# Main loop over selected files
types <- c("blogs", "news", "twitter")
set.seed(12345)
for (fileType in types) {
  preprocess(fileType, dataFolder, LOCALE, dataSampleFolder, sampleRatio )
}
```

```
## [1] "File: ./data/en_US/en_US.blogs.txt"
## [1] "      200 Mb"
## [1] "      37334131 words"
## [1] "      899288 lines"
## [1] "File: ./data/en_US/en_US.news.txt"
## [1] "      196 Mb"
## [1] "      34372530 words"
## [1] "      1010242 lines"
## [1] "File: ./data/en_US/en_US.twitter.txt"
## [1] "      159 Mb"
## [1] "      30373583 words"
## [1] "      2360148 lines"
```

Exploratory Analysis Methodology

Definitions

Before going any further we need to define some terms used hereafter for non data scientist / non NLP enthusiast:
- A (text) corpus, in linguistics, is a large and structured set of texts (https://en.wikipedia.org/wiki/Text_corpus)
- 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 (https://en.wikipedia.org/wiki/N-gram).
- An n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram"; size 3 is a "trigram".
- A term-document matrix (TDM) is a mathematical matrix that describes the frequency of terms (or N-grams for us from now on) that occur in a collection of documents (https://en.wikipedia.org/wiki/Document-term_matrix).
- In computing, stop words are words which are filtered out before or after processing of natural language data text (https://en.wikipedia.org/wiki/Stop_words).

Sample corpus

For this exploratory analysis we will use the reduced / resampled dataset (called sample corpus hereafter) produced during the pre-processing phases. Common NLP R packages are also used for this analysis (see code below)

```
library(NLP)
library(tm)
library(RWeka)
library(RColorBrewer)
library(wordcloud)

corpus_sample <- Corpus(DirSource(file.path(".", dataSampleFolder, LOCALE)),
                        readerControl=list(reader=readPlain, language="en_US"))
```

Methodology

On a long run I aim at finding at most frequent N-grams in the sample corpus (N = 1, 2 and 3) in order to build a text prediction application. But time constraints (and a busy end of year followed by the Christmas break with family to be fully honest) has reduced the scope I had in mind. This report therefore presents most frequent unigrams in the derived sample corpus. Likewise the corpus cleaning process is fairly simple (removing punctuation and converting all characters to lower case). The influence of removing or not the english stop words is analysed on most frequent unigrams.

In both case, the methodology can therefore be summarize by the following steps:
- Perform a fairly simple cleaning of the sample corpus (optional: removing english stop words)
- compute the TDM of unigrams
- Plot most frequent unigrams in above TDM using a cloud of words / a barplot

Useful TDM Post-Processing functions

Below are two useful (and universal) plot functions for illustrating most frequent N-grams of a TDM:
- as a cloud of N-grams
- as an barplot of top X most frequent N-grams

```
# plotWorldCloud function
#     adapted from:
#       Word Cloud in R
#       http://www.r-bloggers.com/word-cloud-in-r/

plotWordCloud <- function(tdm, user_scale=c(3,.3)) {

  m <- as.matrix(tdm)
  v <- sort(rowSums(m), decreasing=TRUE)
  d <- data.frame(word = names(v), freq=v)

  pal <- brewer.pal(8, "Dark2")
  pal <- pal[-(1:2)]

  wordcloud(d$word, d$freq,
            scale=user_scale,
            min.freq=2,
            max.words=50,
            random.order=F,
            rot.per=.15,
            colors=pal,
            vfont=c("sans serif","plain"))
}
```

```r
# plotTopWords function
plotTopWords <- function(tdm, N) {

  m <- as.matrix(tdm)
  v <- sort(rowSums(m), decreasing=TRUE)
  d <- data.frame(word = names(v), freq=v)

  barplot(d[1:N,2], names.arg = d[1:N,1],
          horiz = T,
          cex.names = .6,
          las=1,
          legend.text  = paste("Top", N, "words"))
}
```

Exploratory Analysis: Unigrams and stop words

This section presents the most frequent unigrams (words) in a cleaned sample corpus with optional removing of english stop words. It is pretty clear from the results presented below (see associated plots) that removing has a great impact (as it is expected).

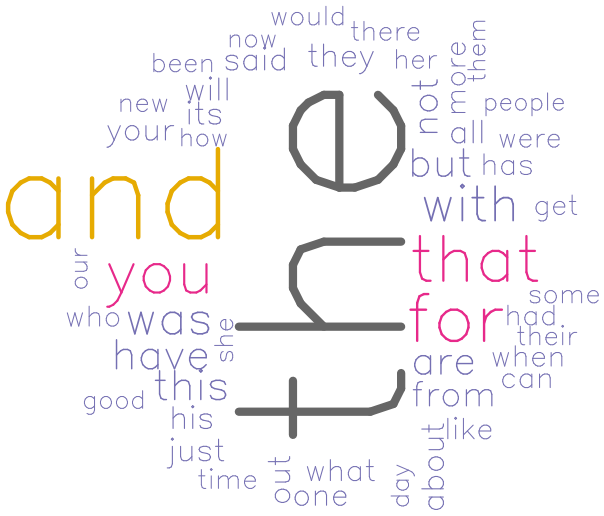
Now the question is wheter or not to include this stage in the cleaning process of a corpus. The answer depends on the final aim of the analysis conducted after. For instance if one is performing a "sentiment analysis" of a corpus then rmeoving the stop words (like "the" in english) is mandatory. But for some other purposes (like a N-grams analysis with $N > 1$) then removing the stop words should be prohibited according to me.

Unigrams in the cleaned sample corpus (excluding removing english stopwords)

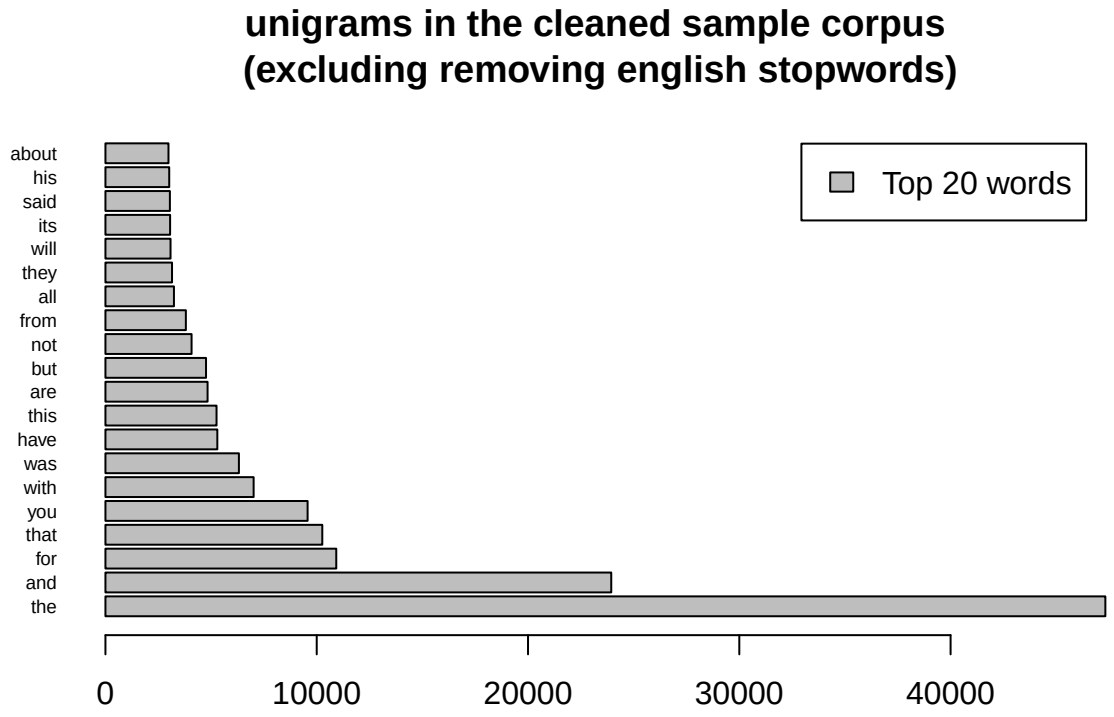
```r
# Corpus cleaning, version 1
corpus_sample_cleaned_1 <- tm_map(corpus_sample, removePunctuation)
corpus_sample_cleaned_1 <- tm_map(corpus_sample_cleaned_1, tolower)
corpus_sample_cleaned_1 <- tm_map(corpus_sample_cleaned_1, PlainTextDocument)

# Associated TDM
tdm_1 <- TermDocumentMatrix(corpus_sample_cleaned_1)
```

```r
plotWordCloud(tdm_1, c(8,.6))
title(main = "unigrams in the cleaned sample corpus \n(excluding removing english stopwords)")
```

unigrams in the cleaned sample corpus
(excluding removing english stopwords)




```
plotTopWords(tdm_1, 20)
title(main = "unigrams in the cleaned sample corpus \n(excluding removing english stopwords)")
```

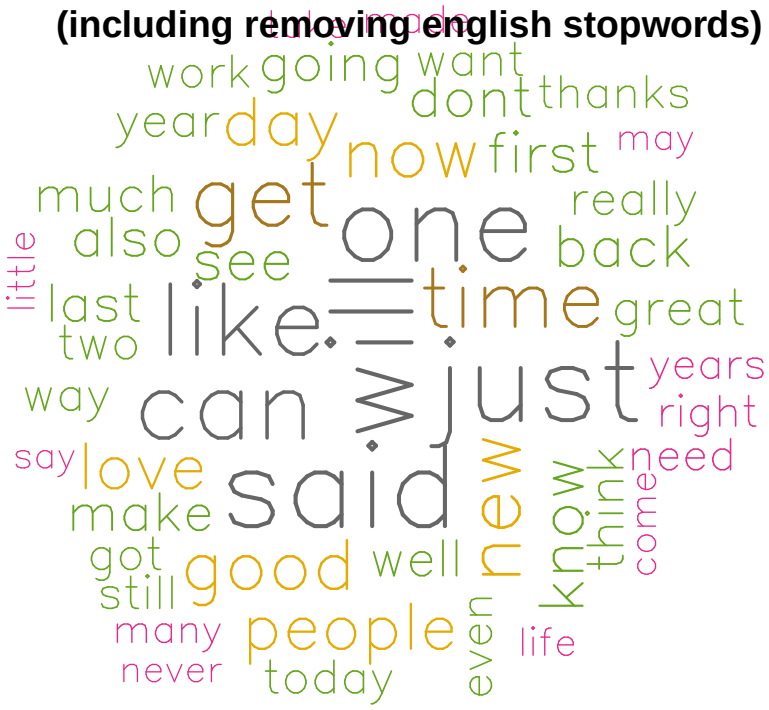


As expected the most frequent words are the stop words and therefore the value of such an analysis would be limited in most cases (unless you are aiming at finding these stop words...)

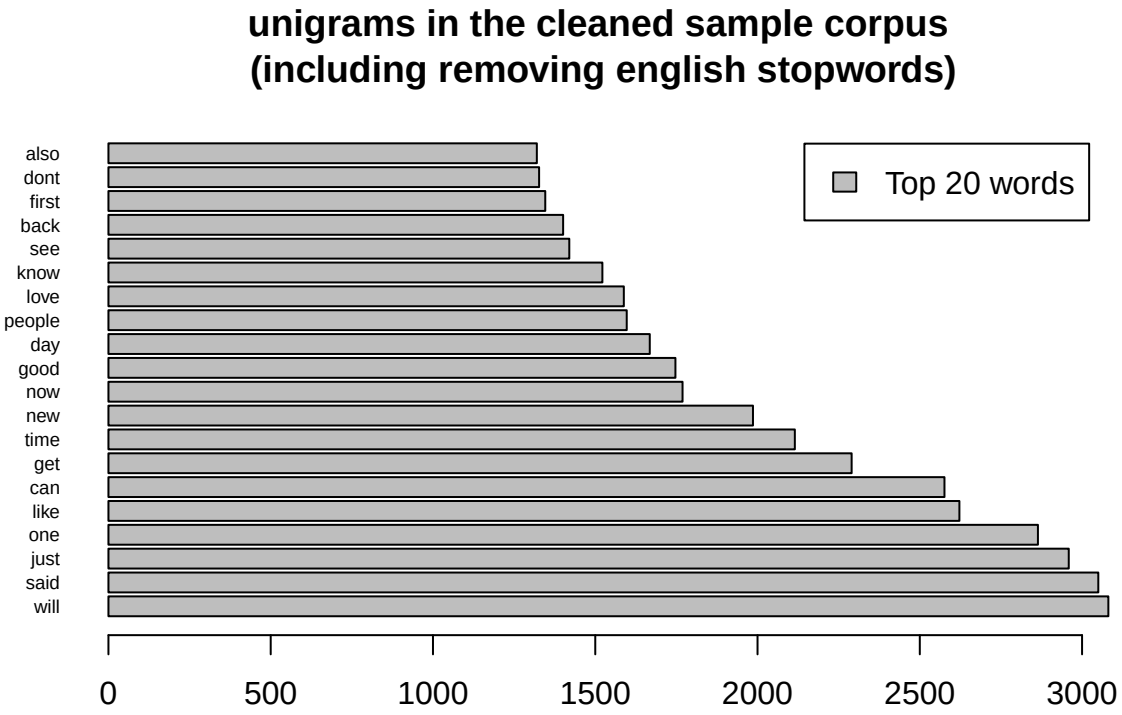
Unigrams in the cleaned sample corpus (including removing english stopwords)

```
# Corpus cleaning, version 2
corpus_sample_cleaned_2 <- tm_map(corpus_sample, removePunctuation)
corpus_sample_cleaned_2 <- tm_map(corpus_sample_cleaned_2, tolower)
corpus_sample_cleaned_2 <- tm_map(corpus_sample_cleaned_2,
                                  function(x) removeWords(x, stopwords("english")))
corpus_sample_cleaned_2 <- tm_map(corpus_sample_cleaned_2, PlainTextDocument)

tdm_2 <- TermDocumentMatrix(corpus_sample_cleaned_2)
```

```
plotWordCloud(tdm_2, c(4,.3))
title(main = "unigrams in the cleaned sample corpus \n(including removing english stopwords)")
```




```
plotTopWords(tdm_2, 20)
title(main = "unigrams in the cleaned sample corpus \n(including removing english stopwords)")
```



The most frequent words after removing the english stop words are still common and rather frequent words but would be useful in "sentiment analysis" I believr (with a corpus less "broad")

Future Plans

Immediate plans for me is to derive and analyze bigrams and trigrams which are more interesting when aiming at building a text prediction app like one can find on smartphones. I have already made some steps towards this goal for the bigrams (see sample code below) but this stage has proven to be too time / CPU demanding to be presented in this report.

```
bigramTokenizer <- function(x) {
  NGramTokenizer(x, Weka_control(min=2, max=2))
}

tdm_bigram <- TermDocumentMatrix(corpus_sample_cleaned, control = list(tokenize=bigramTokenizer))

plotWordCloud(tdm_bigram)
title(main = "Bigrams grams in the cleaned sample corpus \n(excluding removing english stopwords)")
```

Next on my list is to explore more deeply the different cleaning techniques on corpus data and their influence on bigrams and trigrams. It is already clear that removing stopwords as presented in the current report is a bad idea to produce representative N-grams.

As for the final text prediction app I am clearly a bit behind schedule. My initial feeling is to use N-grams but how to build them, on which sample basis seems a crucial issue due to CPU resource required. More bibliographical research on that topic should tell me more I hope.

Conclusion

This report has proven to be difficult to complete due to the Holidays season and some travel back to my home country. But in the end I have found it valuable to realise the amount of the work required to complete the whole project and in that perspective I can say "the sooner the ..."