# Capstone Project: Milestone Report

*Tomas Gogorza*

*March 20th, 2016*

## Summary

The goal of this project is to analyze a large corpus of text documents and discover the structure in the data and how words are put together. This milestone report covers the steps of loading, cleaning and analyzing text data, in the process of getting ready to build a predictive text model.

## Corpus Load and Cleanup

The first step to the creation of our predictive model is to understand the data being used to train our model. In this case the dataset is composed of 3 collections of text pieces, containig blog entries, news and tweets. To create the corpus, we'll be loading 3 files:

```
+ en_US.blogs.txt     (200mb)
+ en_US.news.txt      (196mb)
+ en_US.twitter.txt   (159mb)
```

In order to compose the corpus we need to load the text collections and explore some basic features of the data. Due to the size of the data set and processing and memory constraints, a sample of 10% of each file will be used to create the corpus for this report. After loading the text files, let's take a look of the first lines of each of them:

First 5 blog entries:

```
## [1] "<U+0093>Hell is larger today than it was yesterday, because many of us have failed to pray.<U+0
## [2] "Roughly a week later, my sister and her ever-expanding family drove in from Alaska and picked m
## [3] "\"If I could keep you little,"
## [4] "I first came across these beautiful patterns by artist Paula Vaughan in the mid-1990s. Traditio
## [5] "to do this in remembrance of Him. Luke 22:19"
```

First 5 news:

```
## [1] "The conference is reportedly working toward a new television deal with ESPN, and Neinas was pus
## [2] "Barrett acknowledges now that he might have been too brusque at times."
## [3] "In light of Alex Gonzalez's 413-foot homer on his previous at-bat, Lohse didn't seem to mind wa
## [4] "Light, simple, and relying heavily on stove-top and cold preparation, crab might make a nice ea
## [5] "Another strong month of hiring makes it less likely that the Federal Reserve will take additiona
```

First 5 tweets:

```
## [1] "Beauty Brainstorming in the Alchemy office with and Sally Walker!"
## [2] "<U+0093>: \"The tragedy of life is not that it ends so soon, but that we wait so long to begin
## [3] "More skating! Come by the check out a movie, eat a great dinner and top it off with great times
## [4] "watch your mailbox! : )"
## [5] "Small market baseball. You, know...for the 99%."
```

Now in order to perform some deeper analysis of the corpus, we need to clean up the text, extract unwanted symbols and characters and split the text into tokens. After splitting the corpus, we get the following information:

| Source | Lines | Words |
|---|---|---|
| Blogs | 89210 | 3669936 |
| News | 100755 | 3409951 |
| Tweets | 236824 | 2984313 |
| Total | 426789 | 10064200 |

The above table gives a general sense of the size of the corpus and number of words in each type of data set, as well as in total.

## Exploratory Data Analysis

With the complete list of tokens created, let's proceed to exploring the corpus and get a wider understanding of word relationships. First we need to create a few n-gram collections and explore word frequencies, so we'll create a collection of bigrams and a collection of trigrams (groups of 2 and 3 contiguous words respectively).

Now we can obtain repetition frequencies for tokens, bigrams and trigrams, and we can see how frequently they show up in the corpus.

Tokens:

```
##    token  count freq length
## 1    the 473249 4.70      3
## 2     to 273205 2.71      2
## 3    and 238834 2.37      3
## 4      a 236260 2.35      1
## 5     of 199322 1.98      2
## 6     in 164056 1.63      2
## 7      i 163394 1.62      1
## 8    for 109751 1.09      3
## 9     is 106868 1.06      2
## 10  that 103135 1.02      4
```

Bigrams:

```
##        gram word1 word2 count freq
## 1    of_the    of   the 42586 0.42
## 2    in_the    in   the 40951 0.41
## 3    to_the    to   the 21230 0.21
## 4   for_the   for   the 20172 0.20
## 5    on_the    on   the 19693 0.20
## 6     to_be    to    be 16283 0.16
## 7    at_the    at   the 14144 0.14
## 8   and_the   and   the 12411 0.12
## 9      in_a    in     a 11834 0.12
## 10 with_the  with   the 10550 0.10
```

Trigrams:

```
##              gram  word1 word2 word3 count freq
## 1      one_of_the    one    of   the  3453 0.03
## 2        a_lot_of      a   lot    of  2981 0.03
## 3  thanks_for_the thanks   for   the  2391 0.02
## 4        to_be_a     to    be     a  1877 0.02
## 5     going_to_be  going    to    be  1810 0.02
## 6      the_end_of    the   end    of  1501 0.01
## 7        it_was_a     it   was     a  1470 0.01
## 8      out_of_the    out    of   the  1465 0.01
## 9       i_want_to      i  want    to  1457 0.01
## 10     as_well_as     as  well    as  1413 0.01
```

Now a few plots will give us a better sense of the words distribution

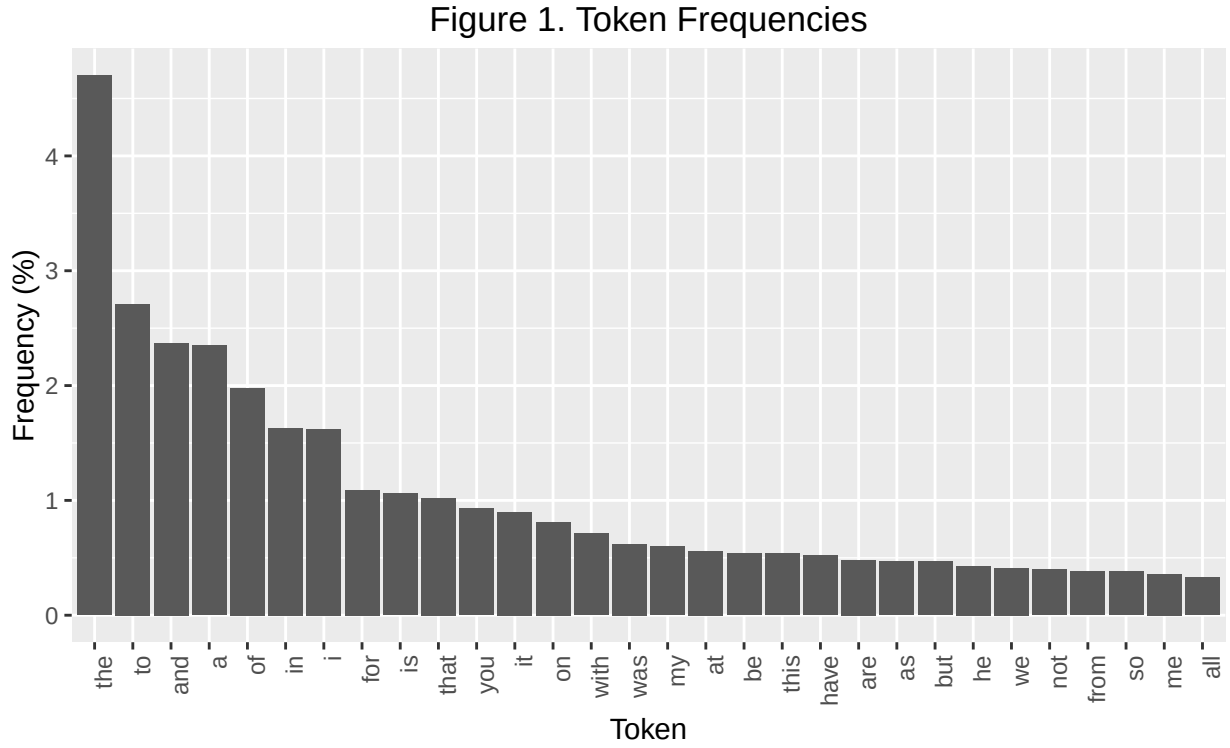


Figure 1. Word count for the 30 most common tokens in the corpus. Bar labels show word frequencies.

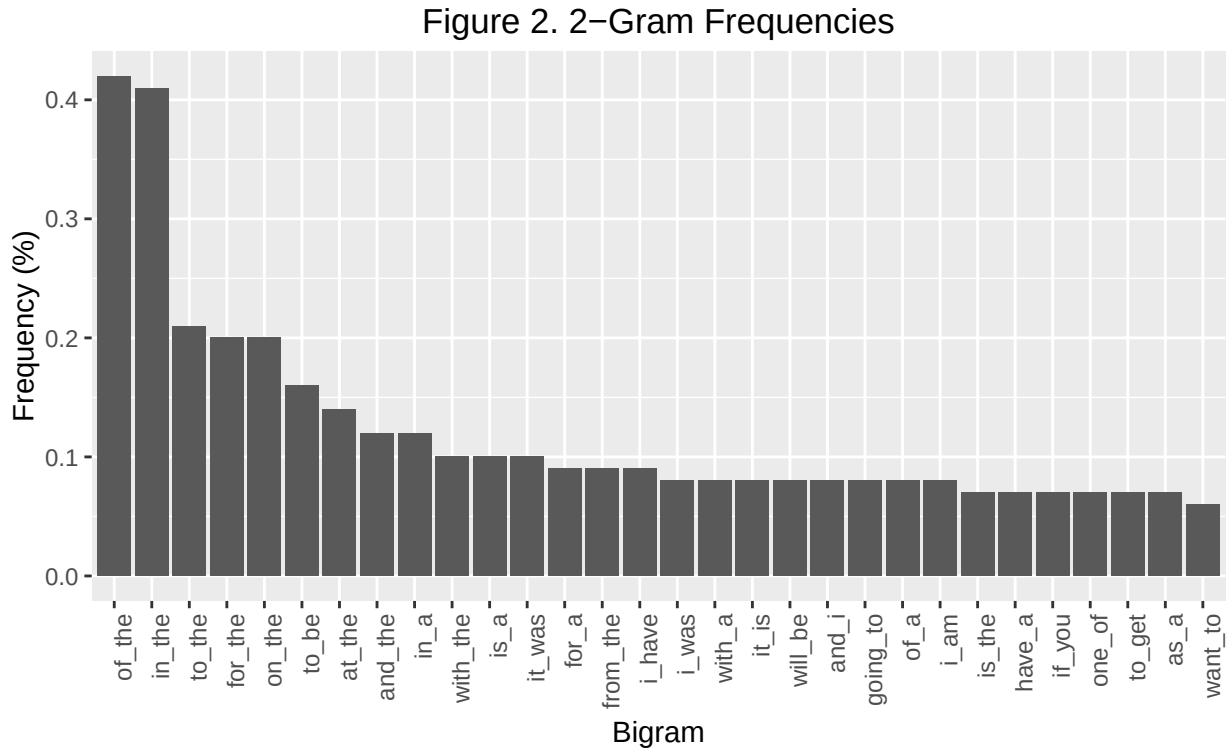


Figure 2. Word count for the 30 most frequent bi-grams in the corpus. Bar labels show word frequencies.

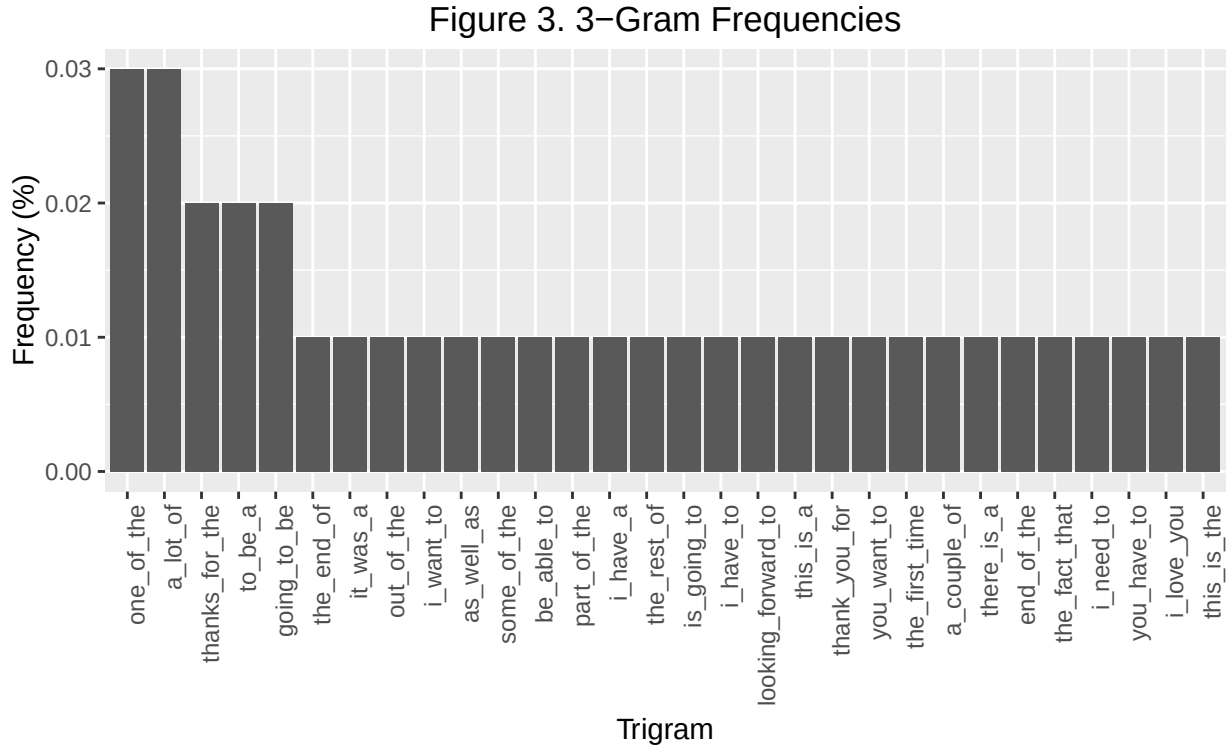


Figure 3. Word count for the 30 most frequent tri-grams in the corpus. Bar labels show word frequencies.

After some initial data analysis, we can see the most frequently typed words are usually short words, and mainly articles, connectors, and prepositions.

## Predictive Model

The main idea for the predictive model is to use the collection of bigrams and trigrams (and possibly higher order n-grams) to create a Markov Chain which will allow to give next-word probabilities as words are input and a sentence starts to form.

## Appendix: The Code

```
#Load libraries and helper functions
library(dplyr)
library(data.table)
library(knitr)
library(ggplot2)
library(tidyr)
source("loadData.R")
source("cleanData.R")
opts_chunk$set(dev = 'pdf')

#Load the data from txt files
blogs <- fread('Data/en_US.blogs.txt', sep = "\n",header = FALSE, encoding = "UTF-8",
               verbose = FALSE, showProgress = FALSE)$V1
news <- fread('Data/en_US.news.txt', sep = "\n",header = FALSE, encoding = "UTF-8",
              verbose = FALSE, showProgress = FALSE)$V1
tweets <- readFile('Data/en_US.twitter.txt')

#Take random samples representing 30% of each dataset
sample <- rbinom(length(blogs),1,0.1)
blogs <- blogs[sample == 1]
sample <- rbinom(length(news),1,0.1)
news <- news[sample == 1]
sample <- rbinom(length(tweets),1,0.1)
tweets <- tweets[sample == 1]

head(blogs,5)
head(news,5)
head(tweets,5)

#Count number of lines on each collection and totals
bloglines <- length(blogs)
newslines <- length(news)
tweetslines <- length(tweets)
totallines <- bloglines + newslines + tweetslines

# Split collections into sets of words, remove symbols and convert to lower case
blogtokens <- tokenize(blogs, remove_symbols = TRUE,
                       convert_to_lower = TRUE, filter_bad_words = FALSE)
newstokens <- tokenize(news, remove_symbols = TRUE,
                       convert_to_lower = TRUE, filter_bad_words = FALSE)
tweetstokens <- tokenize(tweets, remove_symbols = TRUE,
                         convert_to_lower = TRUE, filter_bad_words = FALSE)
#Group all tokens together
tokens <- c(blogtokens,newstokens,tweetstokens)

#Create sets of 2-grams and 3-grams
bigrams <- ngrams(tokens,2)
trigrams <- ngrams(tokens,3)

#Count total number of tokens
totaltokens <- length(tokens)
```

```
totalbigrams <- length(bigrams)
totaltrigrams <- length(trigrams)

#Create ordered tables of word counts
token_table <- sort(table(tokens), decreasing = TRUE)
bigram_table <- sort(table(bigrams), decreasing = TRUE)
trigram_table <- sort(table(trigrams), decreasing = TRUE)

#Create data frame with tokens frequencies and word length
tokenDF <- data.frame(token = names(token_table), count = as.vector(token_table))
tokenDF$token <- as.vector(tokenDF$token)
tokenDF <- mutate(tokenDF, freq = round((count/totaltokens)*100,2), length = nchar(token))
tokenDF[1:10,]

#Create Bigram data frame with frqeuencies and repetition counts
bigramDF <- data.frame(gram = names(bigram_table), count = as.vector(bigram_table))
bigramDF$gram <- as.vector(bigramDF$gram)
bigramDF <- separate(bigramDF,gram,c("word1","word2"),"_",remove = FALSE)
bigramDF <- mutate(bigramDF, freq = round((count/totalbigrams)*100,2) )
bigramDF[1:10,]

#Create Trigram data frame with frqeuencies and repetition counts
trigramDF <- data.frame(gram = names(trigram_table), count = as.vector(trigram_table))
trigramDF$gram <- as.vector(trigramDF$gram)
trigramDF <- separate(trigramDF,gram,c("word1","word2","word3"),"_",remove = FALSE)
trigramDF <- mutate(trigramDF, freq = round((count/totaltrigrams)*100,2))
trigramDF[1:10,]

#Plot word frequencies
ggplot(tokenDF[1:30,], aes(token,freq)) +
    geom_bar( stat="identity", position="dodge") +
    scale_x_discrete(limits=tokenDF$token[1:30]) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    labs(title="Figure 1. Token Frequencies",x="Token",y="Frequency (%)")

#Plot bigram frequencies
ggplot(bigramDF[1:30,], aes(gram,freq)) +
    geom_bar( stat="identity", position="dodge") +
    scale_x_discrete(limits=bigramDF$gram[1:30]) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    labs(title="Figure 2. 2-Gram Frequencies",x="Bigram",y="Frequency (%)")

#Plot trigram frequencies
ggplot(trigramDF[1:30,], aes(gram,freq)) +
    geom_bar( stat="identity", position="dodge") +
    scale_x_discrete(limits=trigramDF$gram[1:30]) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    labs(title="Figure 3. 3-Gram Frequencies",x="Trigram",y="Frequency (%)")
```