Milestone Report

Arturk Mammadli

1/28/2021

##Introduction

This milestone report is based on exploratory data analysis of the SwifKey data provided in the context of the Coursera Data Science Capstone. The data consist of 3 text files containing text from three different sources (blogs, news & twitter). Some of the code is hidden to preserve space, but can be accessed by looking at the Raw .Rmd, which can be found in my GitHub repository https://github.com/erturkmemmedli/Milestone-Report

##Assumptions

It is assumed that the data has been downloaded, unzipped and placed into the active R directory, maintaining the folder structure. Another assumption is that the command `wc` is available in the target system. This command can be used for obtaining text stats and is available on every Unix based system. If you are running windows, you can download the GnuWin32 utility set from http://gnuwin32.sourceforge.net/.

##Raw Data Summary

Below you can find a summary of the three input files. The numbers have been calculated by using the `wc` command.

```
##         Word Count Line Count Longest Line
## blogs       899288   37334435        40835
## news       1010242   34372596        11384
## twitter    2360148   30373830          192
```

##Load Data & Sample

Next, we need to load the data into R so we can start manipulating. We use readLines to load blogs and twitter, but we load news in binomial mode as it contains special characters.

```
# Set the correct working directory
setwd("C:/Users/Erturk Memmedli/Documents//Coursera-Swiftkey//final//en_US")

# Read the blogs and twitter files
source.blogs <- readLines("en_US.blogs.txt", encoding="UTF-8")
source.twitter <- readLines("en_US.twitter.txt", encoding="UTF-8")

# Read the news file. using binary mode as there are special characters in the text
con <- file("en_US.news.txt", open="rb")
source.news <- readLines(con, encoding="UTF-8")
close(con)
rm(con)
```

Now that we have loaded the raw data, we will take a sub sample of each file, because running the calculations using the raw files will be really slow. To take a sample we use a binomial function. Essentially, we flip a coin to decide which lines we should include. We decide to include 1% of each text file.

```
setwd("C:/Users/Erturk Memmedli/Documents//Coursera-Swiftkey//final//en_US")

# Binomial sampling of the data and create the relevant files
sample.fun <- function(data, percent)
{
  return(data[as.logical(rbinom(length(data),1,percent))])
}

# Remove all non english characters as they cause issues down the road
source.blogs <- iconv(source.blogs, "latin1", "ASCII", sub="")
source.news <- iconv(source.news, "latin1", "ASCII", sub="")
source.twitter <- iconv(source.twitter, "latin1", "ASCII", sub="")

# Set the desired sample percentage
percentage <- 0.1

sample.blogs   <- sample.fun(source.blogs, percentage)
sample.news   <- sample.fun(source.news, percentage)
sample.twitter   <- sample.fun(source.twitter, percentage)

dir.create("sample", showWarnings = FALSE)

write(sample.blogs, "sample/sample.blogs.txt")
write(sample.news, "sample/sample.news.txt")
write(sample.twitter, "sample/sample.twitter.txt")

remove(source.blogs)
remove(source.news)
remove(source.twitter)
```

##Sample Summary

A summary for the sample can be seen on the table below.

```
##         Word Count Line Count Longest Line
## blogs        90262    3731192         5054
## news        101220    3426574         5110
## twitter     236221    3035930          140
```

Create & Clean a Corpus

In order to be able to clean and manipulate our data, we will create a corpus, which will consist of the three sample text files

```
library(tm)
```

```
## Loading required package: NLP
```

```
library(RWeka)
library(SnowballC)
sample.corpus <- c(sample.blogs,sample.news,sample.twitter)
my.corpus <- Corpus(VectorSource(list(sample.corpus)))
```

Now that we have our corpus item, we need to clean it. In order to do that, we will transform all characters to lowercase, we will remove the punctuation, remove the numbers and the common english stopwords (and, the, or etc..)

```
my.corpus <- tm_map(my.corpus, content_transformer(tolower))
my.corpus <- tm_map(my.corpus, removePunctuation)
my.corpus <- tm_map(my.corpus, removeNumbers)
my.corpus <- tm_map(my.corpus, removeWords, stopwords("english"))
```

We also need to remove profanity. To do that we will use the google badwords database.

```
setwd("C:/Users/Erturk Memmedli/Documents//Coursera-Swiftkey//final//en_US")
googlebadwords <- read.delim("google_bad_words.txt",sep = ":",header = FALSE)
googlebadwords <- googlebadwords[,1]
my.corpus <- tm_map(my.corpus, removeWords, googlebadwords)
```

Finally, we will strip the excess white space

```
my.corpus <- tm_map(my.corpus, stripWhitespace)
```

Before moving to the next step, we will save the corpus in a text file so we have it intact for future reference.

```
setwd("C:/Users/Erturk Memmedli/Documents//Coursera-Swiftkey//final//en_US//output")
writeCorpus(my.corpus, filenames="my.corpus.txt")
my.corpus <- readLines("my.corpus.txt")
```

Unigram Analysis

The first analysis we will perform is a unigram analysis. This will show us which words are the most frequent and what their frequency is. To do this, we will use the Ngrams_Tokenizer that Maciej Szymkiewicz kindly made public. We will pass the argumemnt 1 to get the unigrams. This will create a unigram Dataframe, which we will then manipulate so we can chart the frequencies using ggplot.

```
setwd("C:/Users/Erturk Memmedli/Documents/Milestone")
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
##
## The following object is masked from 'package:NLP':
##
##     annotate
```

```
source("Ngrams_Tokenizer.R")
unigram.tokenizer <- ngram_tokenizer(1)
wordlist <- unigram.tokenizer(my.corpus)
unigram.df <- data.frame(V1 = as.vector(names(table(unlist(wordlist)))), V2 = as.numeric(table(unlist(w
names(unigram.df) <- c("word","freq")
unigram.df <- unigram.df[with(unigram.df, order(-unigram.df$freq)),]
row.names(unigram.df) <- NULL
save(unigram.df, file="unigram.Rda")
```

```
ggplot(head(unigram.df,15), aes(x=reorder(word,-freq), y=freq)) +
  geom_bar(stat="Identity", fill="blue") +
  geom_text(aes(label=freq), vjust = -0.5) +
  ggtitle("Unigrams frequency") +
  ylab("Frequency") +
  xlab("Term")
```

```
library(imager)
```

```
## Loading required package: magrittr
```

```
##
## Attaching package: 'imager'
```

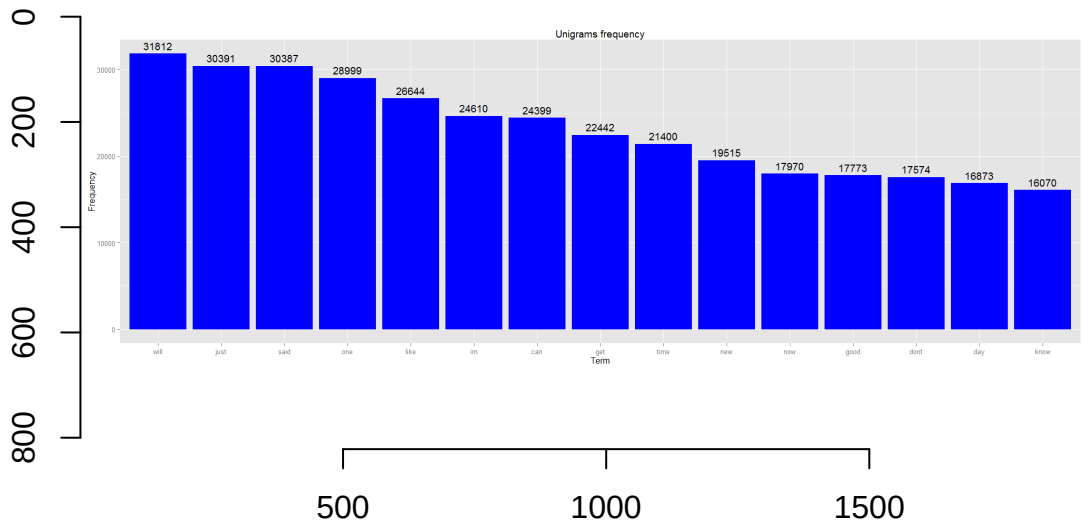
```
## The following object is masked from 'package:magrittr':
##
##     add
```

```
## The following objects are masked from 'package:stats':
##
##     convolve, spectrum
```

```
## The following object is masked from 'package:graphics':
##
##     frame
```

```
## The following object is masked from 'package:base':
##
##     save.image
```

```
plot(load.image("C:/Users/Erturk Memmedli/Documents/Milestone/1.png"))
```



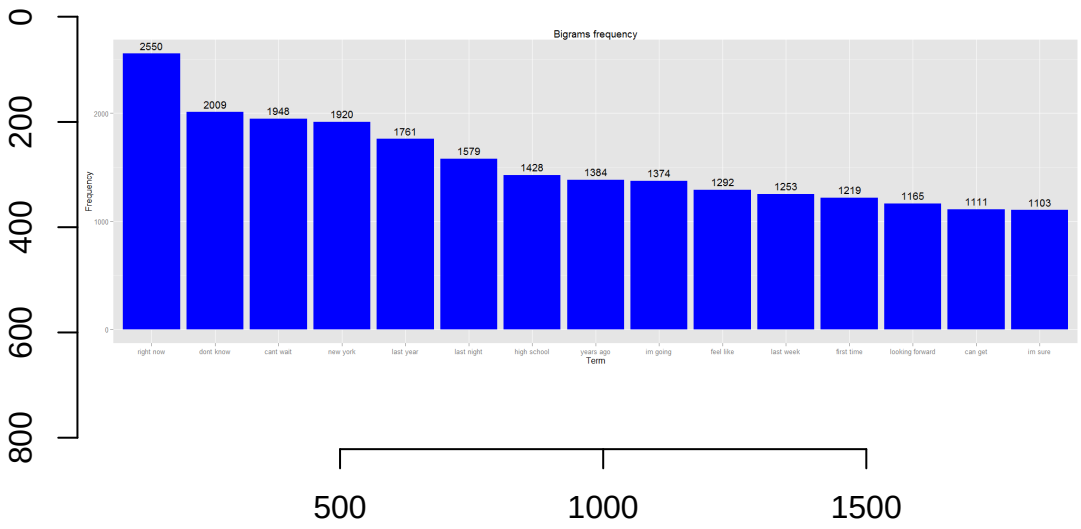
Bigram Analysis

Next, we will do the same for Bigrams, i.e. two word combinations. We follow exactly the same process, but this time we will pass the argument 2.

```
bigram.tokenizer <- ngram_tokenizer(2)
wordlist <- bigram.tokenizer(my.corpus)
bigram.df <- data.frame(V1 = as.vector(names(table(unlist(wordlist)))), V2 = as.numeric(table(unlist(wo
names(bigram.df) <- c("word","freq")
bigram.df <- bigram.df[with(bigram.df, order(-bigram.df$freq)),]
row.names(bigram.df) <- NULL
setwd("C:/Users/Erturk Memmedli/Documents//Coursera-Swiftkey//final//en_US//output")
save(bigram.df, file="bigram.Rda")
```

```
ggplot(head(bigram.df,15), aes(x=reorder(word,-freq), y=freq)) +
  geom_bar(stat="Identity", fill="blue") +
  geom_text(aes(label=freq), vjust = -0.5) +
  ggtitle("Bigrams frequency") +
  ylab("Frequency") +
  xlab("Term")
```

```
library(imager)
plot(load.image("C:/Users/Erturk Memmedli/Documents/Milestone/2.png"))
```



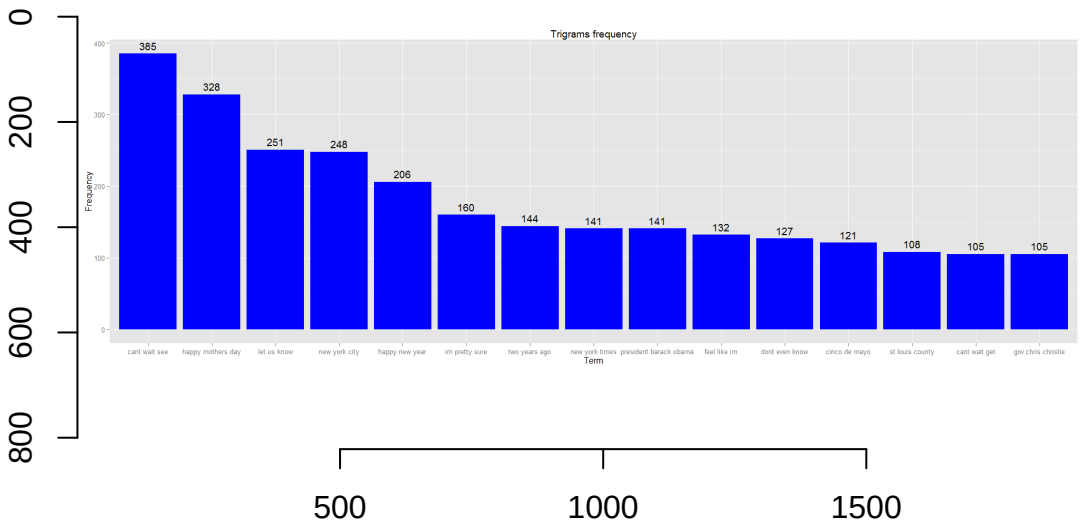
Trigram Analysis

Finally, we will follow exactly the same process for trigrams, i.e. three word combinations.

```
trigram.tokenizer <- ngram_tokenizer(3)
wordlist <- trigram.tokenizer(my.corpus)
trigram.df <- data.frame(V1 = as.vector(names(table(unlist(wordlist)))), V2 = as.numeric(table(unlist(w
names(trigram.df) <- c("word","freq")
trigram.df <- trigram.df[with(trigram.df, order(-trigram.df$freq)),]
row.names(trigram.df) <- NULL
save(trigram.df, file="trigram.Rda")
```

```
ggplot(head(trigram.df,15), aes(x=reorder(word,-freq), y=freq)) +
  geom_bar(stat="Identity", fill="blue") +
  geom_text(aes(label=freq), vjust = -0.5) +
  ggtitle("Trigrams frequency") +
  ylab("Frequency") +
  xlab("Term")
```

```
library(imager)
plot(load.image("C:/Users/Erturk Memmedli/Documents/Milestone/3.png"))
```



Next Steps

This concludes the exploratory analysis. As a next step a model will be created and integrated into a Shiny app for word prediction.

We will use the Ngram dataframes created to calculate the probability of the next word occuring. The input string will be tokenized and the last 2 (or 1 if it's a unigram) words will be isolated and cross checked against the data frames to get the highest probability next word.

The model will then be integrated into a shiny application that will provide a simple and intuitive front end for the ned user.