Exploratory data analysis Milestone report Week2

Jasmina

05 junij 2021

Background

The aim of the assigmnet is making an app that will predict the next word, when a user is typing. This report is a Milestone report from Week 2. The aim of this week is to: 1. Demonstrate that you've downloaded the data and have successfully loaded it in. 2. Create a basic report of summary statistics about the data sets. 3. Report any interesting findings that you amassed so far. 4. Get feedback on your plans for creating a prediction algorithm and Shiny app.

Library and Data loading

Data are downloaded from following link and unzipped in a directory.Link

```
library(R.utils)
library(tm)
library(dplyr)
library(tidytext)
library(RWeka)
library(textmineR)
library(ggplot2)

USBlog <- readLines("Coursera-SwiftKey/final/en_US/en_US.blogs.txt")
USNews <- readLines("Coursera-SwiftKey/final/en_US/en_US.news.txt")
```

```
## Warning in readLines("Coursera-SwiftKey/final/en_US/en_US.news.txt"): incomplete
## final line found on 'Coursera-SwiftKey/final/en_US/en_US.news.txt'
```

```
USTwitter <- readLines("Coursera-SwiftKey/final/en_US/en_US.twitter.txt")
```

```
## Warning in readLines("Coursera-SwiftKey/final/en_US/en_US.twitter.txt"): line
## 167155 appears to contain an embedded nul

## Warning in readLines("Coursera-SwiftKey/final/en_US/en_US.twitter.txt"): line
## 268547 appears to contain an embedded nul

## Warning in readLines("Coursera-SwiftKey/final/en_US/en_US.twitter.txt"): line
## 1274086 appears to contain an embedded nul

## Warning in readLines("Coursera-SwiftKey/final/en_US/en_US.twitter.txt"): line
## 1759032 appears to contain an embedded nul
```

Data statistics

```
data_stat <- data.frame(File_Name=c("US_Blog", "US_News", "Us_Twitter"), FileSize_Mb = c(file.info("Cou
data_stat
```

```
##    File_Name FileSize_Mb N_Lines N_characters Longest_row
## 1    US_Blog    200.4242  899288    208361438       40835
## 2    US_News    196.2775   77259     15683765        5760
## 3 Us_Twitter    159.3641 2360148    162384825         213
```

The size of the datasets, number of lines (N_Lines), number of characters (N_characters) and Longest row are shown in the table. Because the datasets are very large we will use only 5% of the data for the calculations

Sampling and cleaning the data

Sampling of the data (5%)

```
set.seed(123)
sample_size <- 5/100
blog_index <- sample(seq_len(length(USBlog)), length(USBlog)*sample_size)
News_index <- sample(seq_len(length(USNews)), length(USNews)*sample_size)
Twitter_index <- sample(seq_len(length(USTwitter)), length(USTwitter)*sample_size)

blog_s <- USBlog[blog_index[]]
News_s <- USNews[News_index[]]
Twitter_s <- USTwitter[Twitter_index[]]
```

First we load the sample data into a corpus (collection of documents) which is the data structure used by package tm. To preprocess the data we will get rid of any whitespaces, punctuation and numbers. We will also change all the characters to lower characters and remove stopwords (the, any...)

```
Data <- VCorpus(VectorSource(c(blog_s, News_s, Twitter_s)), readerControl = list(reader =readPlain, lan
Data <- tm_map(Data, stripWhitespace)
Data <- tm_map(Data, content_transformer(tolower))
Data <- tm_map(Data, removePunctuation)
Data <- tm_map(Data, removeNumbers)
Data <- tm_map(Data, removeWords, stopwords("english"))
```

Tokenization

Tokenization is used to break the text into parts of a word. We will use Rweka library to construct n-grams of the data. N-gram is a continious sequence of n items (words in this case) from a given sample of text or speech. We will construc a unigram (1 word), bigram (2 words) and threegram (3 words).

```
unigram <- function(x) NGramTokenizer(x, Weka_control(min=1, max=1))
bigram <- function(x) NGramTokenizer(x, Weka_control(min=2, max=2))
trigram <- function(x) NGramTokenizer(x, Weka_control(min=3, max=3))

uni_dtm <- TermDocumentMatrix(Data, control=list(tokenize=unigram))
```

```
bi_dtm <- TermDocumentMatrix(Data, control = list(tokenize=bigram))
tri_dtm <- TermDocumentMatrix(Data, control = list(tokenize=trigram))

uni_frqt <- findFreqTerms(uni_dtm, lowfreq = 50)
bi_frqt <- findFreqTerms(bi_dtm, lowfreq = 50)
tri_frqt <- findFreqTerms(tri_dtm, lowfreq = 10)

u <- rowSums(as.matrix(uni_dtm[uni_frqt,]))
b <- rowSums(as.matrix(bi_dtm[bi_frqt,]))
t <- rowSums(as.matrix(tri_dtm[tri_frqt,]))
```

```
uni_freq <- data.frame(term=rownames(u), freq=u, row.names = NULL)
```

```
## Error in data.frame(term = rownames(u), freq = u, row.names = NULL): arguments imply differing numbe
```

```
Uni_freq_table <- uni_freq[order(-uni_freq$frequency),]

bi_freq <- data.frame(term=rownames(b), freq=b, row.names = NULL)
```

```
## Error in data.frame(term = rownames(b), freq = b, row.names = NULL): arguments imply differing numbe
```

```
bi_freq_table <- bi_freq[order(-bi_freq$frequency), ]

tri_freq <- data.frame(term=rownames(t), freq=t, row.names = NULL)
```

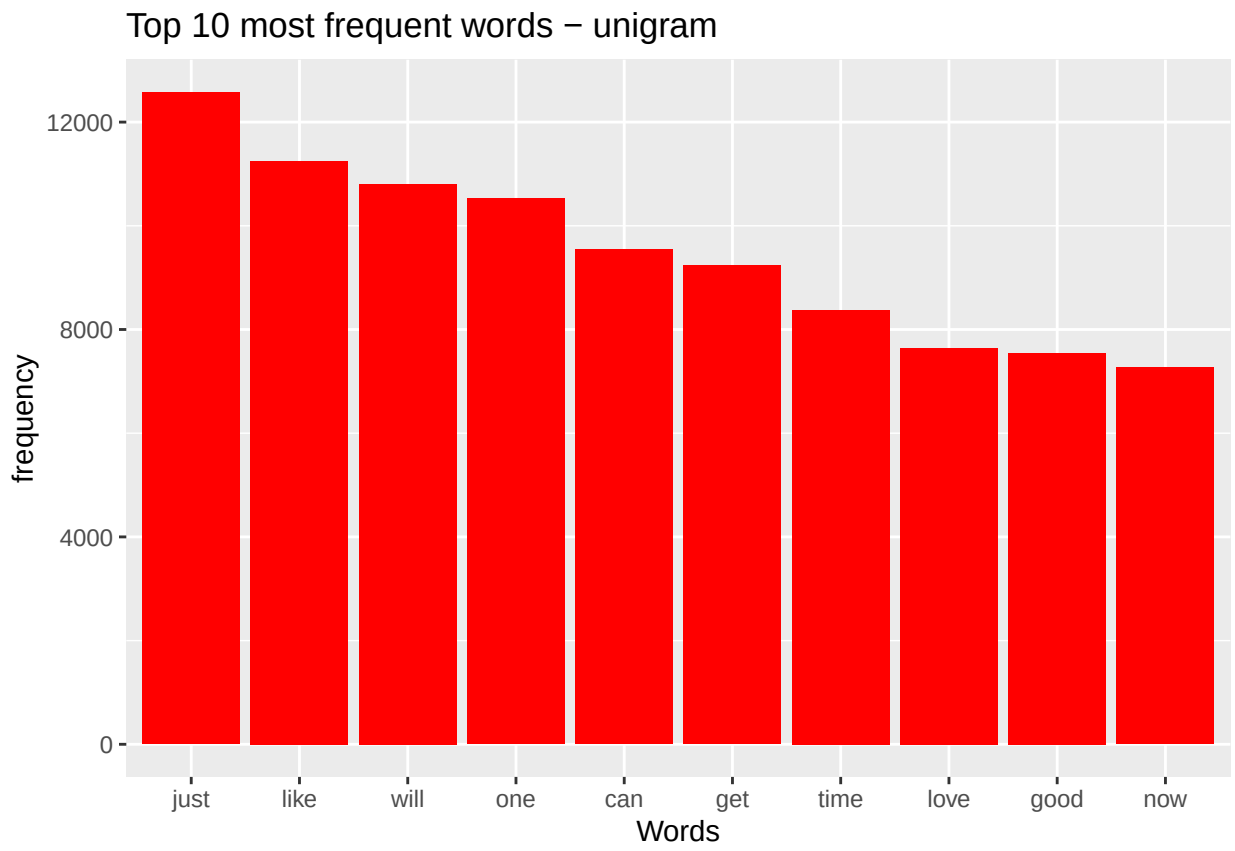
```
## Error in data.frame(term = rownames(t), freq = t, row.names = NULL): arguments imply differing numbe
```

```
tri_freq_table <- tri_freq[order(-tri_freq$frequency),]
```

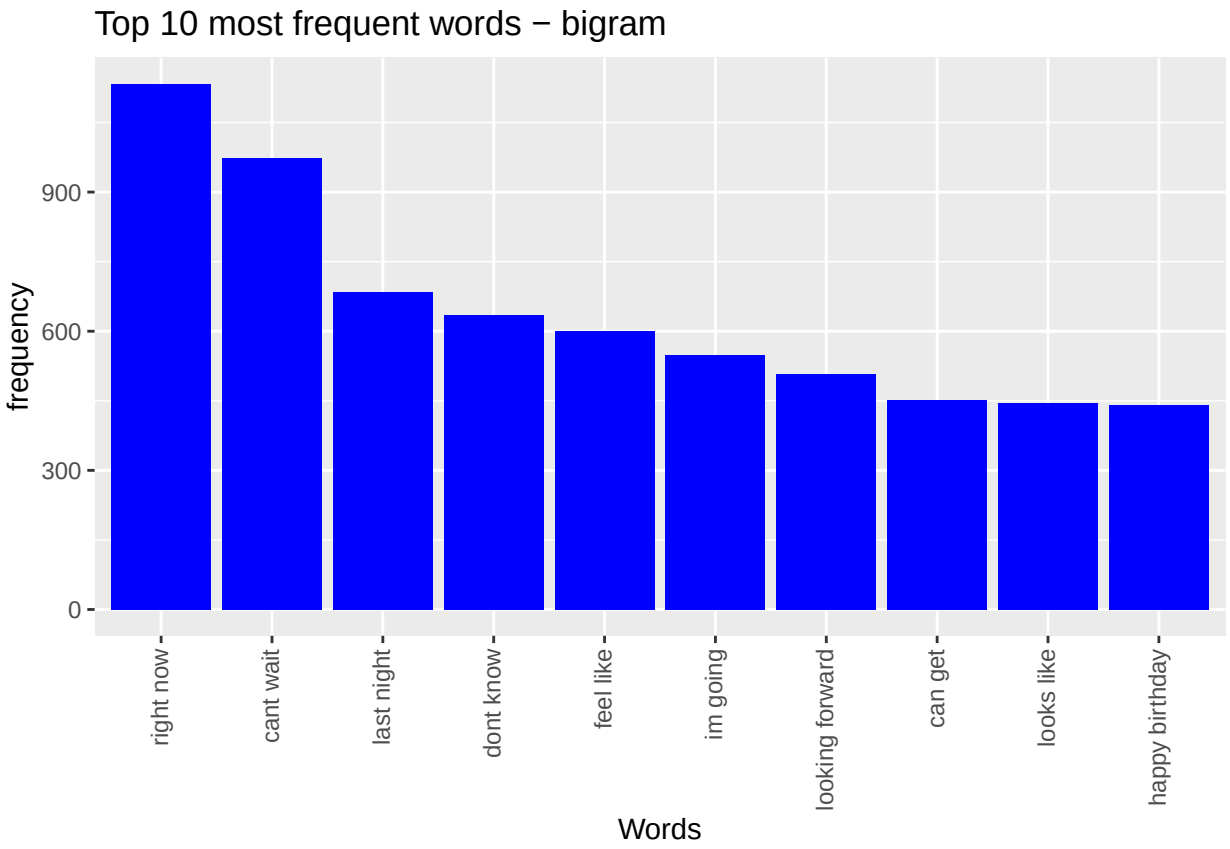
Data visualization

On the graphs we will show the 10 most frequent words (or bigram and threegram) from the dataset

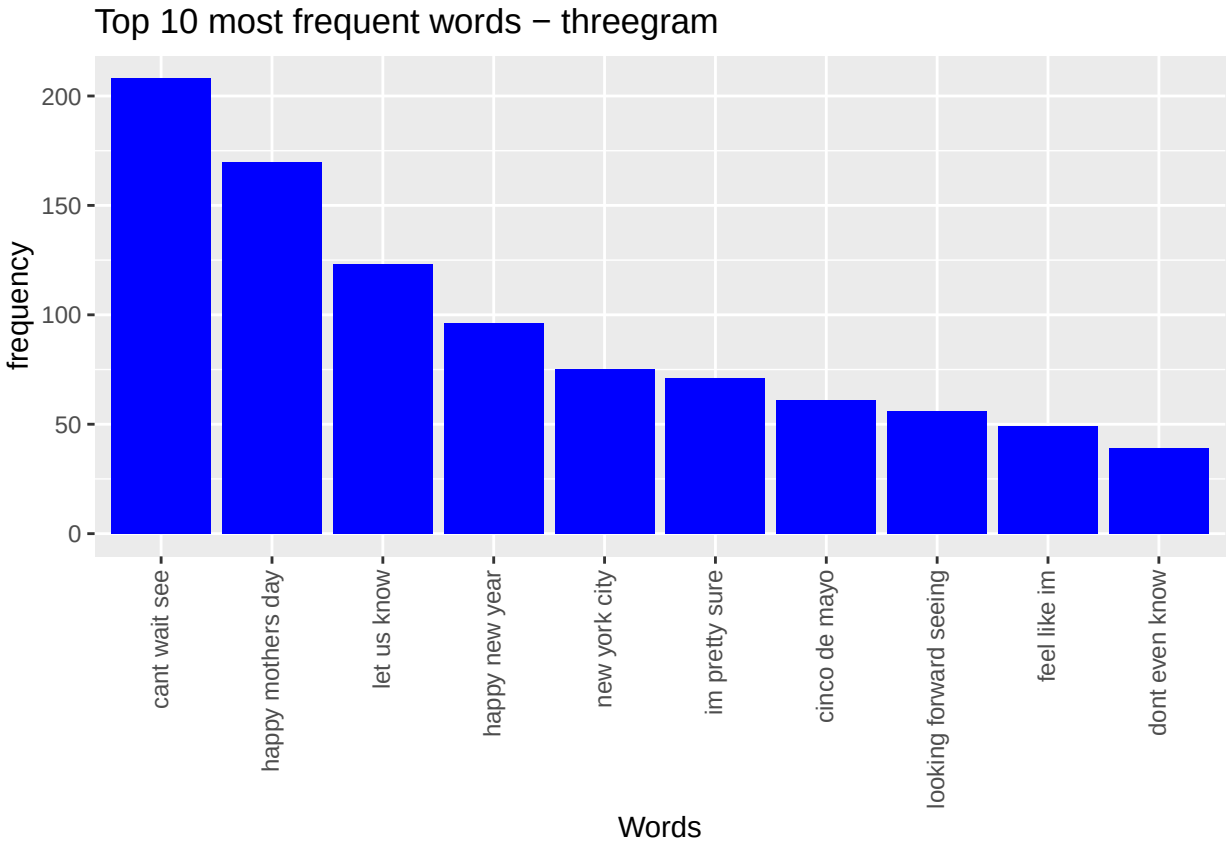
```
ggplot(Uni_freq_table[1:10,], aes(x=reorder(words, -frequency), y=frequency)) + geom_bar(stat="identity
```




```
ggplot(bi_freq_table[1:10,], aes(x=reorder(words, -frequency), y=frequency)) + geom_bar(stat="identity"
```




```
ggplot(tri_freq_table[1:10,], aes(x=reorder(words, -frequency), y=frequency)) + geom_bar(stat="identity
```



Next step

The next step will be to build a model which will be able to predict the text and then apply it to a shiny app.