Milestone Report

Karishma Yadav

7/15/2020

Project Overview

The objective of this project is demonstrate that the project data can be read into RStudio, and an exploratory analysis subsequently performed. The major features of the datasets are shown in tables and plots, and plans for creating a prediction algorithm are discussed.

Load data

Load the required files and set up the work environment.

```
setwd("C:\\Users\\ADMIN\\Desktop\\data science\\10 Data Science Capstone\\DS PROJECT\\Coursera-SwiftKey
blogs<-readLines("en_US.blogs.txt",warn=FALSE,encoding="UTF-8")
news<-readLines("en_US.news.txt",warn=FALSE,encoding="UTF-8")
twitter<-readLines("en_US.twitter.txt",warn=FALSE,encoding="UTF-8")
```

Summarize data

To get a sense of what the data looks like, I summerized the main information from each of the 3 datasets (Blog, News and Twitter). I calculate the size of each file in MB,number of lines and words in each file,average word count per line in each file, max count of char per line in each file and others details.

```
size_blogs<-file.size(path="C:\\Users\\ADMIN\\Desktop\\data science\\10 Data Science Capstone\\DS PROJE
size_news<-file.size(path="C:\\Users\\ADMIN\\Desktop\\data science\\10 Data Science Capstone\\DS PROJECT
size_twitter<-file.size(path="C:\\Users\\ADMIN\\Desktop\\data science\\10 Data Science Capstone\\DS PRO
len_blogs<-length(blogs)
len_news<-length(news)
len_twitter<-length(twitter)
nchar_blogs<-sum(nchar(blogs))
nchar_news<-sum(nchar(news))
nchar_twitter<-sum(nchar(twitter))
library(stringi)
nword_blogs<-stri_stats_latex(blogs)[4]
nword_news<-stri_stats_latex(news)[4]
nword_twitter<-stri_stats_latex(twitter)[4]
table<-data.frame("File Name"=c("Blogs","News","Twitter"),
                  "File Size(MB)"=c(size_blogs,size_news,size_twitter),
                  "Num of rows"=c(len_blogs,len_news,len_twitter),
                  "Num of character"=c(nchar_blogs,nchar_news,nchar_twitter),
                  "Num of words"=c(nword_blogs,nword_news,nword_twitter))
table
```

```
##   File.Name File.Size.MB. Num.of.rows Num.of.character Num.of.words
## 1     Blogs      200.4242      899288        206824505     37570839
## 2      News      196.2775       77259         15639408      2651432
## 3   Twitter      159.3641     2360148        162096031     30451128
```

Clean data

Data sets are really big, so using sample() function, I sample 1% of each file.

```
set.seed(12345)
blogs1<-iconv(blogs,"latin1","ASCII",sub="")
news1<-iconv(news,"latin1","ASCII",sub="")
twitter1<-iconv(twitter,"latin1","ASCII",sub="")
rm(blogs)
rm(news)
rm(twitter)
# sample data set only 1% of each file
sample_data<-c(sample(blogs1,length(blogs1)*0.01),
               sample(news1,length(news1)*0.01),
               sample(twitter1,length(twitter1)*0.01))
rm(blogs1)
rm(news1)
rm(twitter1)
```

Build corpus

This section will use the text mining library 'tm' (loaded previously) to perform Data cleaning tasks, which are meaningful in Predictive Text Analytics. Main cleaning steps are :

1. Converting the document to lowercase
2. Removing punctuation marks
3. Removing numbers
4. Removing stopwords (i.e. "and", "or", "not", "is", etc)
5. Removing undesired terms
6. Removing extra whitespaces generated in previous 5 steps

```
library(tm)
```

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

```
library(NLP)
corpus<-VCorpus(VectorSource(sample_data))
corpus1<-tm_map(corpus,removePunctuation)
corpus2<-tm_map(corpus1,stripWhitespace)
corpus3<-tm_map(corpus2,tolower)
corpus4<-tm_map(corpus3,removeNumbers)
corpus5<-tm_map(corpus4,PlainTextDocument)
corpus6<-tm_map(corpus5,removeWords,stopwords("english"))
corpus_result<-data.frame(text=unlist(sapply(corpus6,'[',"content")),stringsAsFactors = FALSE)
head(corpus_result)
```

```
##
## 1
## 2
## 3
```

```
## 4
## 5
## 6   answer  pretty straightforward  need    muscle biopsy    painful muscles yes  know  something
```

```
rm(corpus)
rm(corpus1)
rm(corpus2)
rm(corpus3)
rm(corpus4)
rm(corpus5)
```

Build corpus, and check it making data frame.

Build N-gram

```
library(RWeka)
one<-function(x) NGramTokenizer(x,Weka_control(min=1,max=1))
two<-function(x) NGramTokenizer(x,Weka_control(min=2,max=2))
thr<-function(x) NGramTokenizer(x,Weka_control(min=3,max=3))
one_table<-TermDocumentMatrix(corpus6,control=list(tokenize=one))
two_table<-TermDocumentMatrix(corpus6,control=list(tokenize=two))
thr_table<-TermDocumentMatrix(corpus6,control=list(tokenize=thr))
one_corpus<-findFreqTerms(one_table,lowfreq=1000)
two_corpus<-findFreqTerms(two_table,lowfreq=80)
thr_corpus<-findFreqTerms(thr_table,lowfreq=10)
one_corpus_num<-rowSums(as.matrix(one_table[one_corpus,]))
one_corpus_table<-data.frame(Word=names(one_corpus_num),frequency=one_corpus_num)
one_corpus_sort<-one_corpus_table[order(-one_corpus_table$frequency),]
head(one_corpus_sort)
```

```
##      Word frequency
## just just      2576
## like like      2218
## will will      2211
## one   one      2049
## get   get      1869
## can   can      1866
```

```
two_corpus_num<-rowSums(as.matrix(two_table[two_corpus,]))
two_corpus_table<-data.frame(Word=names(two_corpus_num),frequency=two_corpus_num)
two_corpus_sort<-two_corpus_table[order(-two_corpus_table$frequency),]
head(two_corpus_sort)
```

```
##                 Word frequency
## cant wait   cant wait       208
## right now   right now       206
## dont know   dont know       164
## last night last night       148
## im going     im going       130
## feel like   feel like       125
```

```
thr_corpus_num<-rowSums(as.matrix(thr_table[thr_corpus,]))
thr_corpus_table<-data.frame(Word=names(thr_corpus_num),frequency=thr_corpus_num)
thr_corpus_sort<-thr_corpus_table[order(-thr_corpus_table$frequency),]
head(thr_corpus_sort)
```

```
##                                     Word frequency
## cant wait see                cant wait see        45
## happy mothers day       happy mothers day        36
## happy new year             happy new year        24
## im pretty sure             im pretty sure        18
## italy lakes holidays italy lakes holidays        18
## little italy boston   little italy boston        17
```

Extract the word and frequency of N-grams.

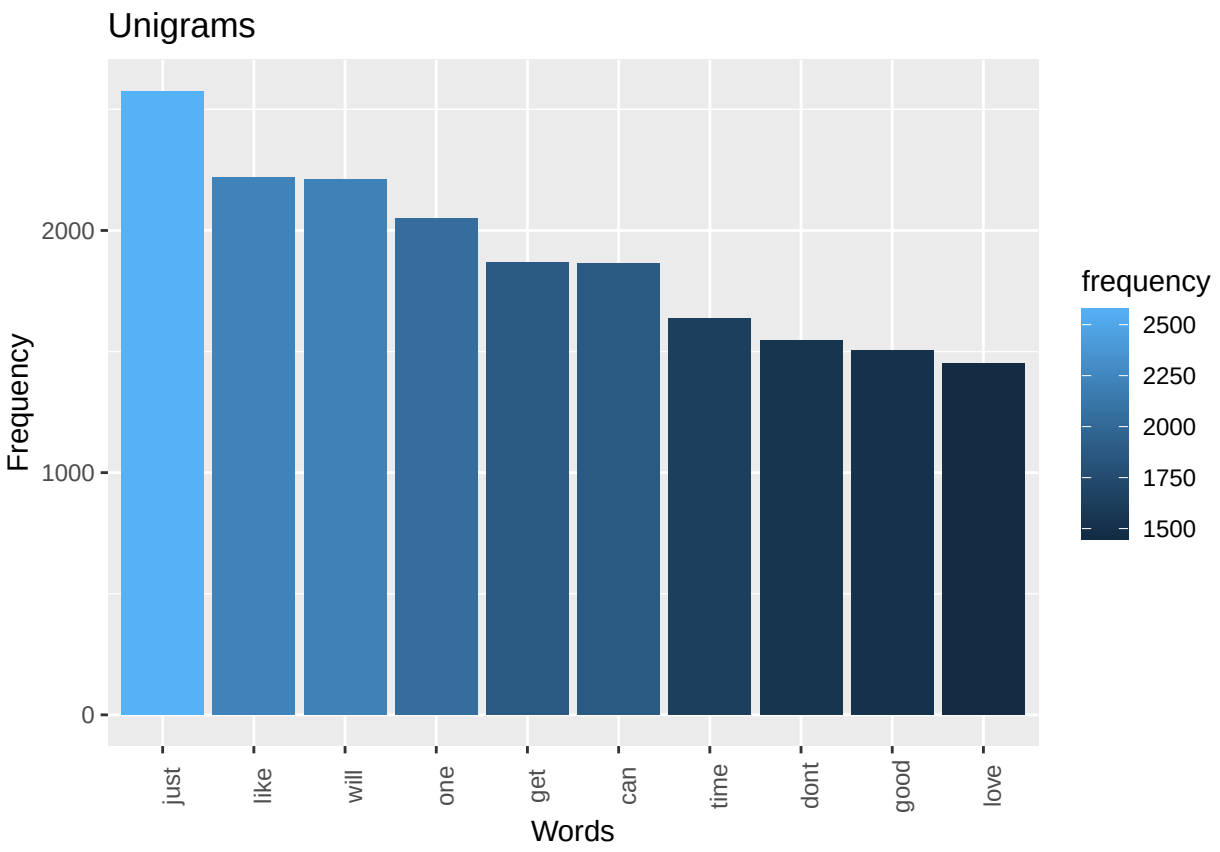
Plot graph

```
library(ggplot2)
```

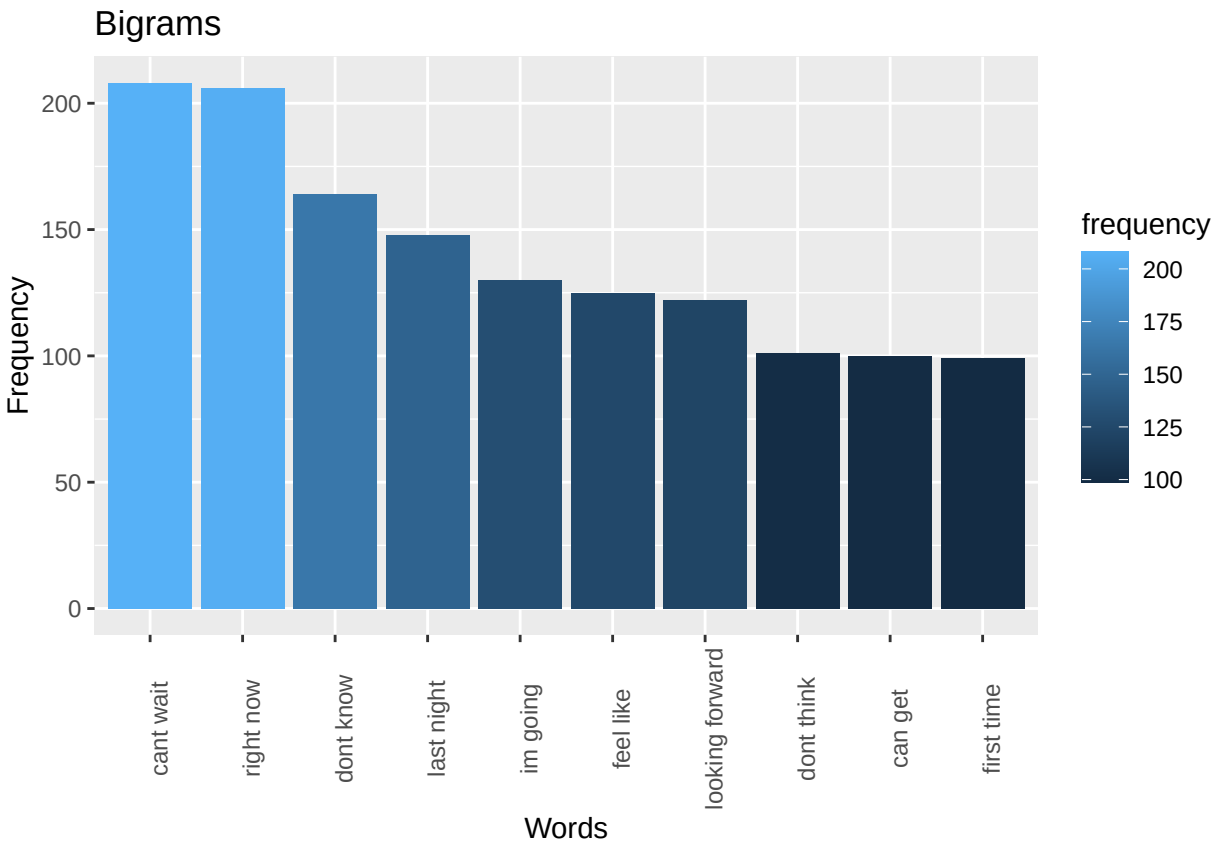
```
##
## Attaching package: 'ggplot2'
```

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

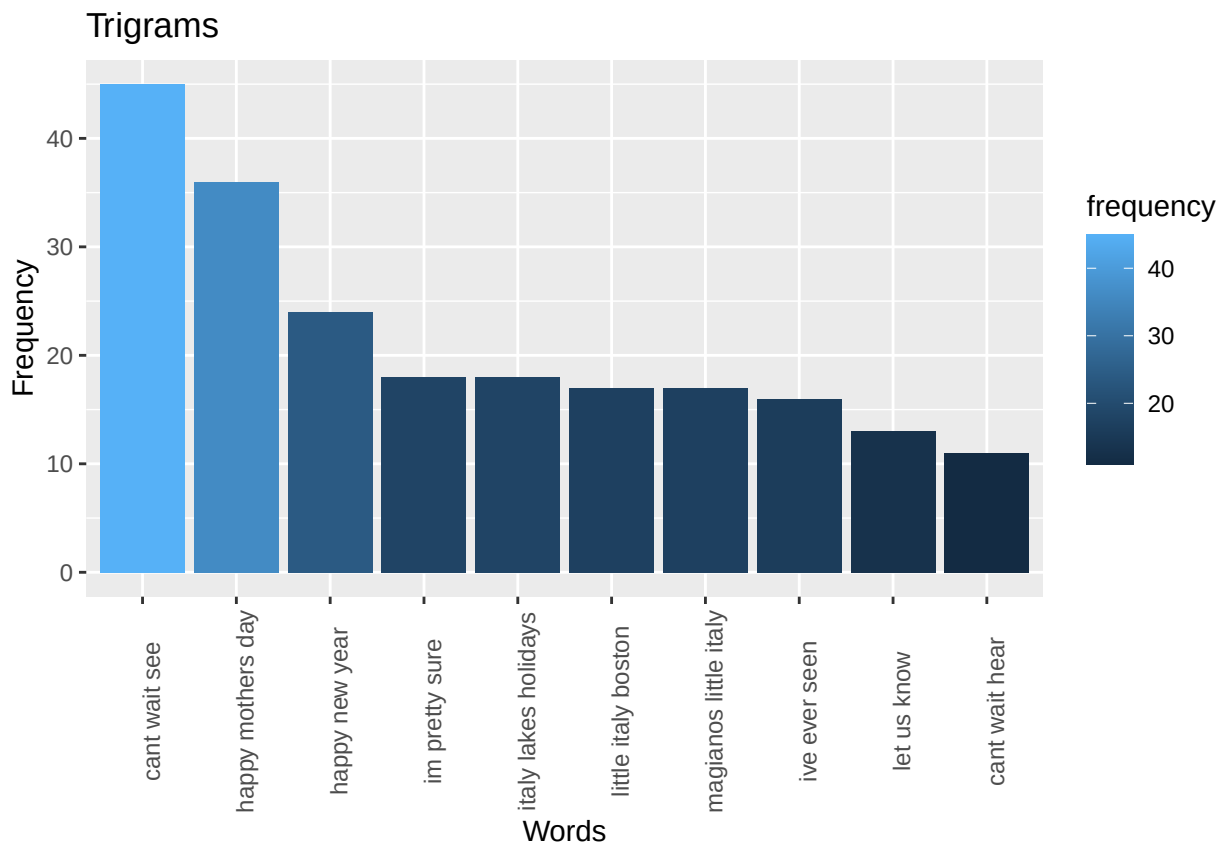
```
one_g<-ggplot(one_corpus_sort[1:10,],aes(x=reorder(Word,-frequency),y=frequency,fill=frequency))
one_g<-one_g+geom_bar(stat="identity")
one_g<-one_g+labs(title="Unigrams",x="Words",y="Frequency")
one_g<-one_g+theme(axis.text.x=element_text(angle=90))
one_g
```



```
two_g<-ggplot(two_corpus_sort[1:10,],aes(x=reorder(Word,-frequency),y=frequency,fill=frequency))
two_g<-two_g+geom_bar(stat="identity")
two_g<-two_g+labs(title="Bigrams",x="Words",y="Frequency")
two_g<-two_g+theme(axis.text.x=element_text(angle=90))
two_g
```




```
thr_g<-ggplot(thr_corpus_sort[1:10,],aes(x=reorder(Word,-frequency),y=frequency,fill=frequency))
thr_g<-thr_g+geom_bar(stat="identity")
thr_g<-thr_g+labs(title="Trigrams",x="Words",y="Frequency")
thr_g<-thr_g+theme(axis.text.x=element_text(angle=90))
thr_g
```



Plot graphs of each N-gram words. I can confirm which word is the most frequency in those files.

Evaluation

1. Prediction

We see that small parts of the data are responsible for the bulk of the corpus. This allows prediction to be a smaller model to just focus on the most important parts.

2. Next steps

- Reevaluate approach and see if sample size adjust,inclusion of stopwords, punctuation, numbers, etc improve prediction.
- Building a predictive model using the identified tokens.
- Wrapping up the results and the developed model as a data product, shiny app.

Apendix I - Source codes

This document has been generated using ***R Mardown***. Its ***.Rmd*** source code that can be found at: https://github.com/Karishma-Yadav/Data-Science-Capstone-Project.