Data Science Capstone Project - Swiftkey

Jeet Tanna

7/30/2020

Introduction

This data science capstone project is to build a text prediction model from the given datasets that contain a large amount of text files from the internet. This kind of project is related to Natural Language Processing where text files are mainly used.

We will first import the data. ### Data Source The data source is : https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip (https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip)

Importing The Data

We will first import the data.

```
twitter<-readLines("C:/Users/JeetsPC-1/Desktop/Study Material/R DataSets/Coursera-SwiftKey/fina
l/en_US/en_US.twitter.txt",encoding = "UTF-8", skipNul = TRUE)
blog<-readLines("C:/Users/JeetsPC-1/Desktop/Study Material/R DataSets/Coursera-SwiftKey/final/en
_US/en_US.blogs.txt",encoding = "UTF-8", skipNul = TRUE)
news<-readLines("C:/Users/JeetsPC-1/Desktop/Study Material/R DataSets/Coursera-SwiftKey/final/en
_US/en_US.news.txt",encoding = "UTF-8", skipNul = TRUE)
```

```
## Warning in readLines("C:/Users/JeetsPC-1/Desktop/Study Material/R DataSets/
## Coursera-SwiftKey/final/en_US/en_US.news.txt", : incomplete final line
## found on 'C:/Users/JeetsPC-1/Desktop/Study Material/R DataSets/Coursera-
## SwiftKey/final/en_US/en_US.news.txt'
```

Summarizing the Data

```
sizes<-c(
  file.size("C:/Users/JeetsPC-1/Desktop/Study Material/R DataSets/Coursera-SwiftKey/final/en_US/en_US.twitter.txt"),
  file.size("C:/Users/JeetsPC-1/Desktop/Study Material/R DataSets/Coursera-SwiftKey/final/en_US/en_US.blogs.txt"),
  file.size("C:/Users/JeetsPC-1/Desktop/Study Material/R DataSets/Coursera-SwiftKey/final/en_US/en_US.news.txt")
)
sizes<-sizes/2^20

lengths<-c(length(twitter),
           length(blog),
           length(news))

chars<-c(sum(nchar(twitter)),sum(nchar(blog)),sum(nchar(news)))
library(stringi)
```

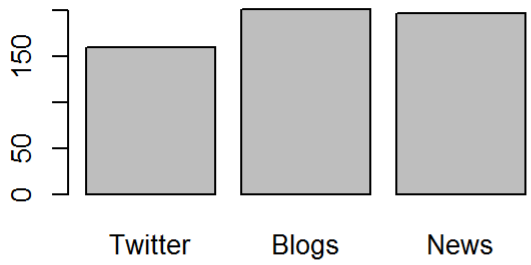
```
## Warning: package 'stringi' was built under R version 3.5.3
```

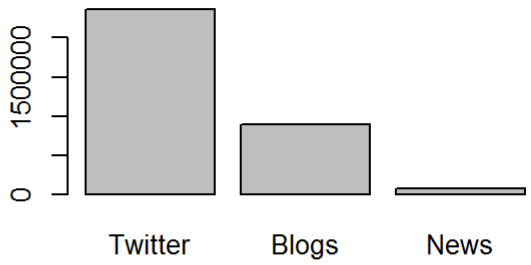
```
words<-c(stri_stats_latex(twitter)[4],stri_stats_latex(blog)[4],stri_stats_latex(news)[4])
file<-c("Twitter","Blogs","News")
table<-data.frame(file,sizes,lengths,chars,words)
table
```

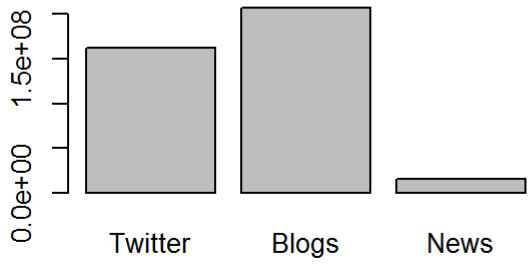
```
##      file    sizes lengths     chars    words
## 1 Twitter 159.3641 2360148 162096241 30451170
## 2   Blogs 200.4242  899288 206824505 37570839
## 3    News 196.2775   77259  15639408  2651432
```

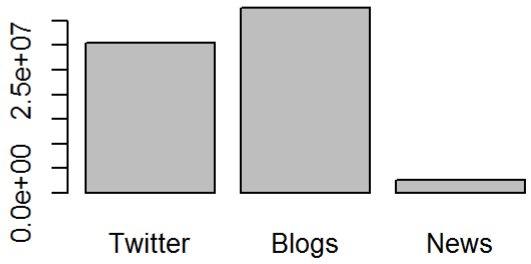
Basic Exploratory Analysis

```
library(ggplot2)
par(mfrow=c(2,2))
barplot(table$sizes,names.arg = c("Twitter","Blogs","News"))
barplot(table$lengths,names.arg = c("Twitter","Blogs","News"))
barplot(table$chars,names.arg = c("Twitter","Blogs","News"))
barplot(table$words,names.arg = c("Twitter","Blogs","News"))
```









We notice a few things from these barplots: 1. The News dataset has the most size in mb but the least content in the form of words or characters. So we can expect long lines in the news dataset. 2. Blogs and Twitter datasets are similar in terms of size, characters, words and length.

Approach to take

The approach that will be taken in further weeks is as follows: 1. I will have to make a corpus that cleans the data and organizes it well. 2. Then an NGram will be made so words can be recognized using packages like NLP and tm 3. Prediction models will be made using this NGram.

Cleaning Data

```
blog<-iconv(blog,"latin1","ASCII",sub="")
news<-iconv(news,"latin1","ASCII",sub="")
twitter<-iconv(twitter,"latin1","ASCII",sub="")
```

Build corpus

```
sample_data<-c(sample(twitter,length(twitter)*0.01),
               sample(blog,length(blog)*0.01),
               sample(news,length(news)*0.01))
library(tm)
```

```
## Warning: package 'tm' was built under R version 3.5.3
```

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

```
##
## Attaching package: 'NLP'
```

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

```
library(NLP)
corpus<-VCorpus(VectorSource(sample_data))
corpus<-tm_map(corpus,removePunctuation)
corpus<-tm_map(corpus,stripWhitespace)
corpus<-tm_map(corpus,tolower)
corpus<-tm_map(corpus,removeNumbers)
corpus<-tm_map(corpus,PlainTextDocument)
corpus<-tm_map(corpus,removeWords,stopwords("english"))
corpus1<-data.frame(text=unlist(sapply(corpus,'[',"content")),stringsAsFactors = FALSE)
corpus1[1:6,]
```

```
## [1] "dj spooky asks   free culture explore  apps like make    creativecommons  education"
## [2] "gmwill don make sure u seize  daymake   ur lil bitch lol"
## [3] "ancientaliens marathon yes"
## [4] "hahahaaah right people    confident"
## [5] "maltonikesb  awesome"
## [6] " thank   helping    need great cause"
```

Build corpus, and check it making data frame.

Build N-gram

```
library(RWeka)
```

```
## Warning: package 'RWeka' was built under R version 3.5.3
```

```
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(corpus,control=list(tokenize=one))
two_table<-TermDocumentMatrix(corpus,control=list(tokenize=two))
thr_table<-TermDocumentMatrix(corpus,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      2554
## like like      2245
## will will      2167
## one   one      2110
## can   can      1938
## get   get      1930
```

```
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
## right now   right now       219
## cant wait   cant wait       217
## dont know   dont know       181
## last night last night       148
## im going     im going       113
## feel like   feel like       111
```

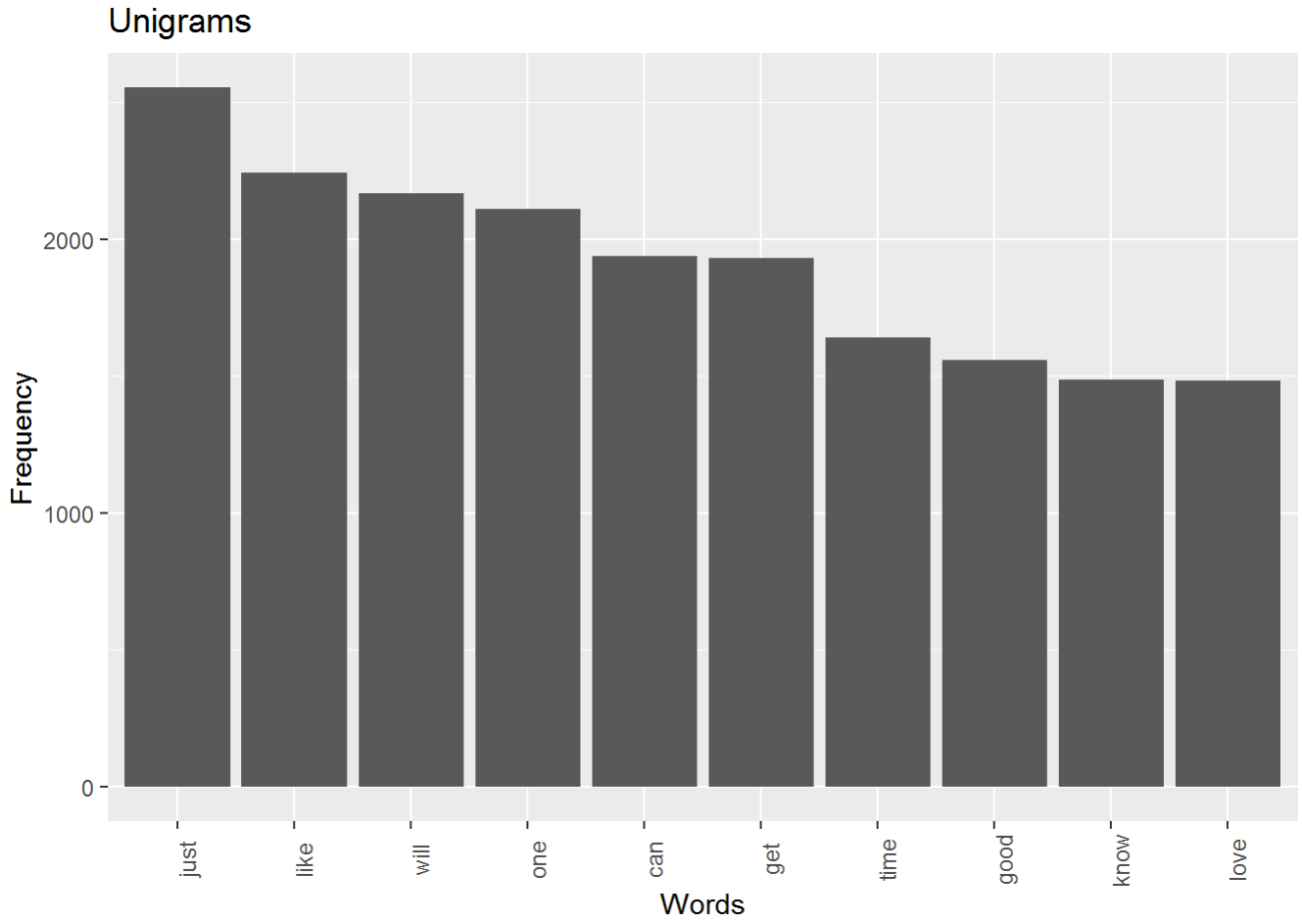
```
thr_corpus_num<-rowSums(as.matrix(thr_table[thr_corpus,]))
thr_corpus_table<-data.frame(Word=names(thr_corpus_num),frequency=thr_corpus_num)
thr_corpus_sort<-thr_corpus_table[order(-thr_corpus_table$frequency),]
head(thr_corpus_sort)
```

```
##                               Word frequency
## happy mothers day happy mothers day        44
## cant wait see         cant wait see        31
## let us know             let us know        29
## happy new year       happy new year        22
## cant wait get         cant wait get        12
## dont even know       dont even know        12
```

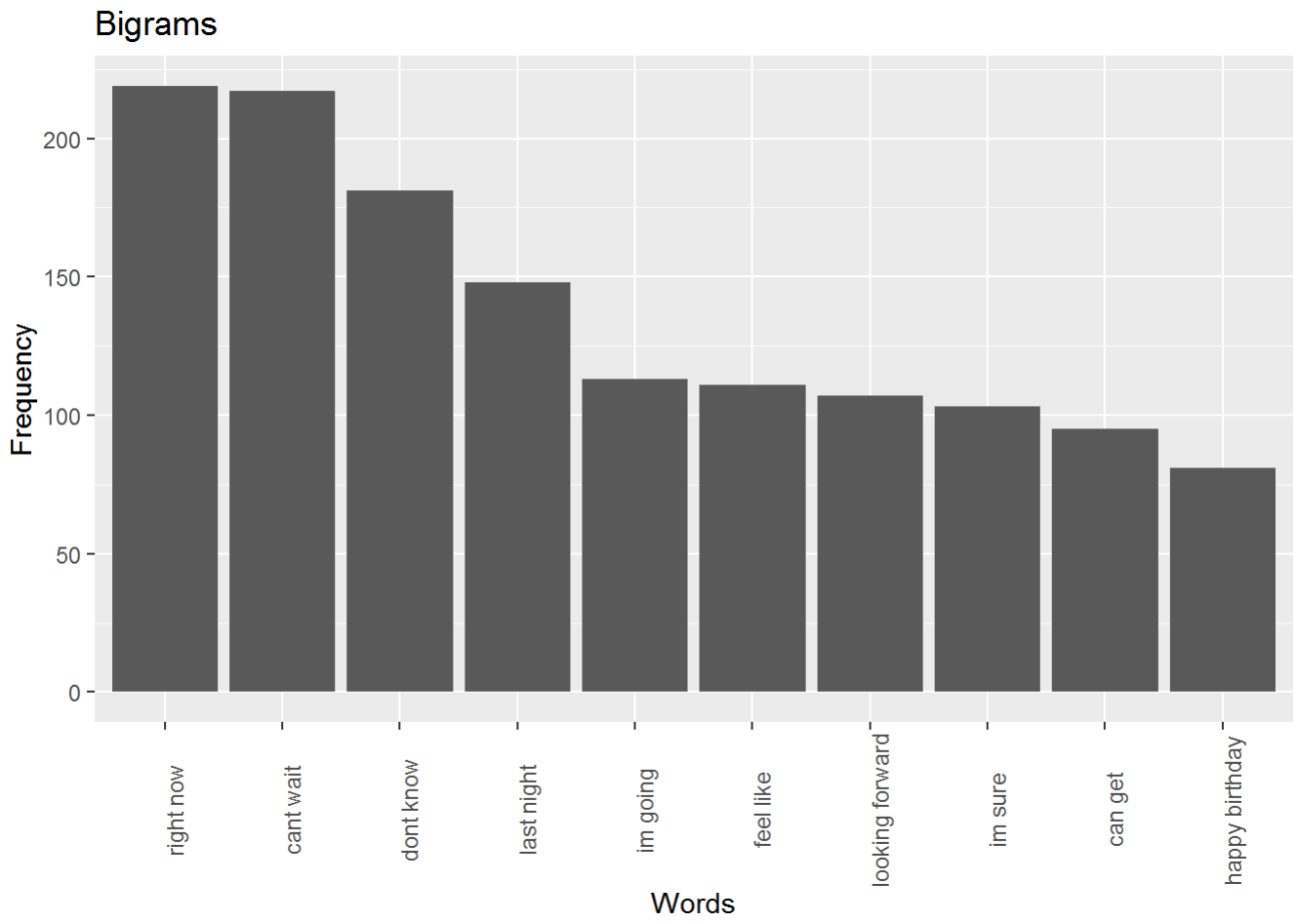
Extract the word and frequency of N-grams.

Plot graph

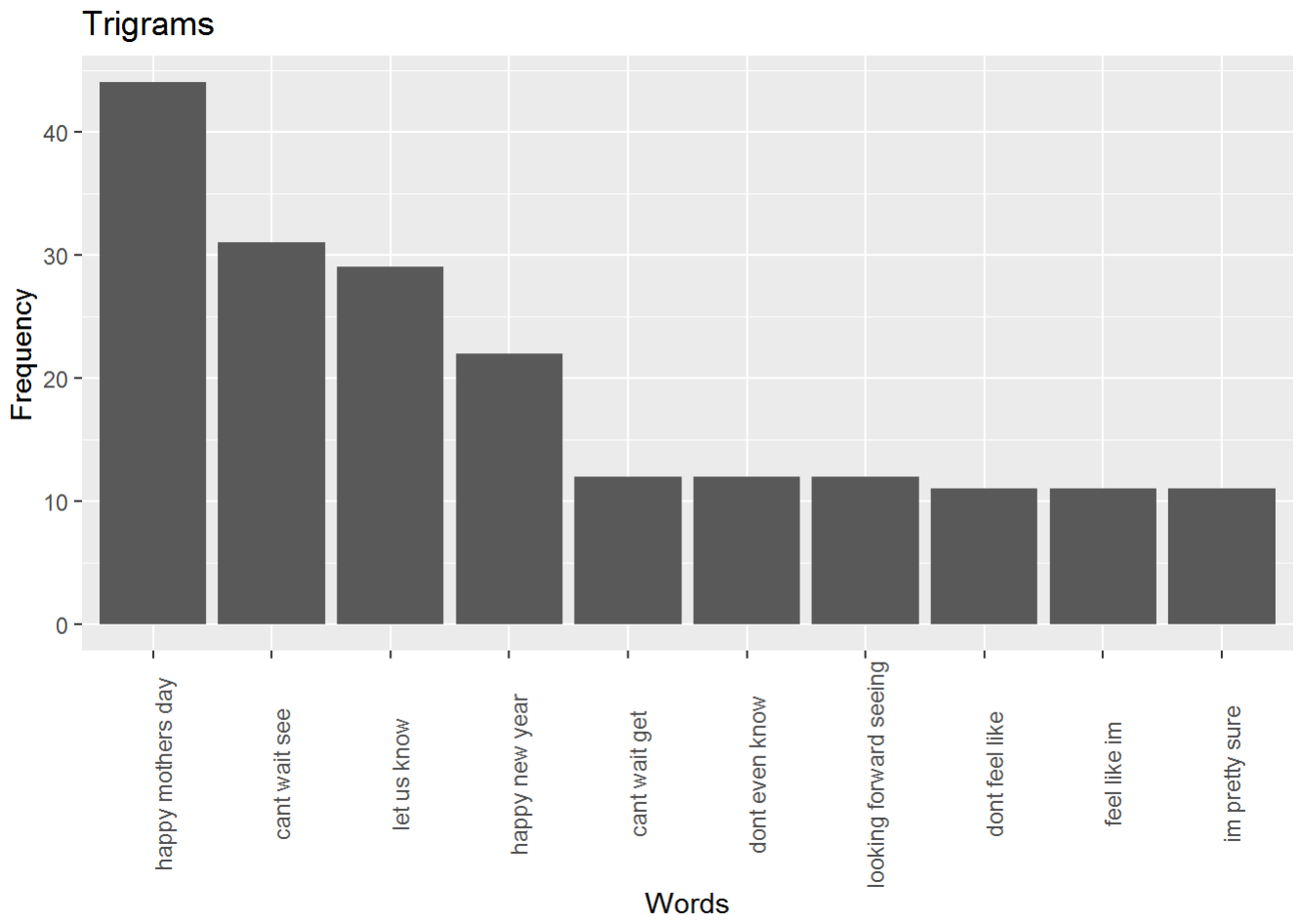
```
library(ggplot2)
one_g<-ggplot(one_corpus_sort[1:10,],aes(x=reorder(Word,-frequency),y=frequency))
one_g<-one_g+geom_bar(stat="identity")
one_g<-one_g+labs(title="Unigrams",x="Words",y="Frequency")
one_g<-one_g+theme(axis.text.x=element_text(angle=90))
one_g
```




```
two_g<-ggplot(two_corpus_sort[1:10,],aes(x=reorder(Word,-frequency),y=frequency))
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))
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
```




```
saveRDS(one_g, file = "unigram.RDS")
saveRDS(two_g, file = "bigram.RDS")
saveRDS(thr_g, file = "trigram.RDS")
```