Milestone Week 2 - Data Science Capstone Project

Paulo Cardoso

May 1, 2016

Executive Summary

This milestone report is for the Coursera Data Science Capstone project. Which aims to explain the preprocessing method of data employed and the exploratory data analysis performed. The data used are from the data packet provided by the course organizer and from the provided data files that have been used are:

- Blog "./data/Coursera-SwiftKey/final/en_US/en_US.blogs.txt"
- News "./data/Coursera-SwiftKey/final/en_US/en_US.news.txt"
- Twitter "./data/Coursera-SwiftKey/final/en_US/en_US.twitter.txt"

The Data used on this project can be found on: https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip

Input Data And Setting Up

Downloading the Data through R and unzipping

```
# Remove # commentary tag
#URL <- "https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip"
#download.file(URL,destfile = "Coursera-SwiftKey.zip")
#unzip(zipfile = "Coursera-SwiftKey.zip")
```

Setting Work Directory

```
# Remove # commentary tag
#setwd("./final/en_US")
```

Loading the data into Variables

```
blogsRaw <- readLines("en_US.blogs.txt", encoding = "UTF-8", skipNul=TRUE)
newsRaw <- readLines("en_US.news.txt", encoding = "UTF-8", skipNul=TRUE)
```

```
## Warning in readLines("en_US.news.txt", encoding = "UTF-8", skipNul = TRUE):
## incomplete final line found on 'en_US.news.txt'
```

```
twitterRaw <- readLines("en_US.twitter.txt", encoding = "UTF-8", skipNul=TRUE)
```

Loading Libraries

```
library(stringi)
library(ggplot2)
library(gridExtra)
library(tm)
library(wordcloud)
library(RWeka)
```

Getting and Cleaning the Data

The goal of this task is to get familiar with the databases and do the necessary cleaning. In order to be able to accomplish this it is necessary to performing the following steps:

- Sampling

```
# Random seed
set.seed(1234)
# Sampling
blogsSample <- sample(blogsRaw,size =  0.05*length(blogsRaw),replace = FALSE)
newsSample <- sample(newsRaw,size =  0.05*length(newsRaw),replace = FALSE)
twitterSample <- sample(twitterRaw,size =  0.05*length(twitterRaw),replace = FALSE)
# Merging Samples
dataSample <- paste0(c(blogsSample, newsSample,twitterSample))
```

- Profanity filtering and Data cleaning

The list of profanity words used on this project can be found on: http://www.freewebheaders.com/full-list-of-bad-words-banned-by-google/

```
# Creating Corpus
dataSample <- iconv(dataSample, "latin1", "ASCII", sub="")
dataCorpus <- Corpus(VectorSource(list(dataSample)))
# Setting up bad words
badWords <- read.csv("badwords.txt")
badWords <- as.vector(t(badWords))
# Cleaning data and removing profanity
dataCorpus <- tm_map(dataCorpus, content_transformer(tolower))
dataCorpus <- tm_map(dataCorpus, content_transformer(removePunctuation))
dataCorpus <- tm_map(dataCorpus, content_transformer(removeNumbers))
dataCorpus <- tm_map(dataCorpus, stripWhitespace)
dataCorpus <- tm_map(dataCorpus, removeWords, stopwords("english"))
dataCorpus <- tm_map(dataCorpus, removeWords, badWords)
dataCorpus <- tm_map(dataCorpus, stemDocument, language='english')
```

- Tokenization

```
# tokenizer function
tUni <- function(x) NGramTokenizer(x, Weka_control(min = 1, max = 1))
tBi <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
tTri <- function(x) NGramTokenizer(x, Weka_control(min = 3, max = 3))
```

```
# Tokenizing
uniMat <- TermDocumentMatrix(dataCorpus, control = list(tokenize = tUni))
biMat <- TermDocumentMatrix(dataCorpus, control = list(tokenize = tBi))
triMat <- TermDocumentMatrix(dataCorpus, control = list(tokenize = tTri))
```

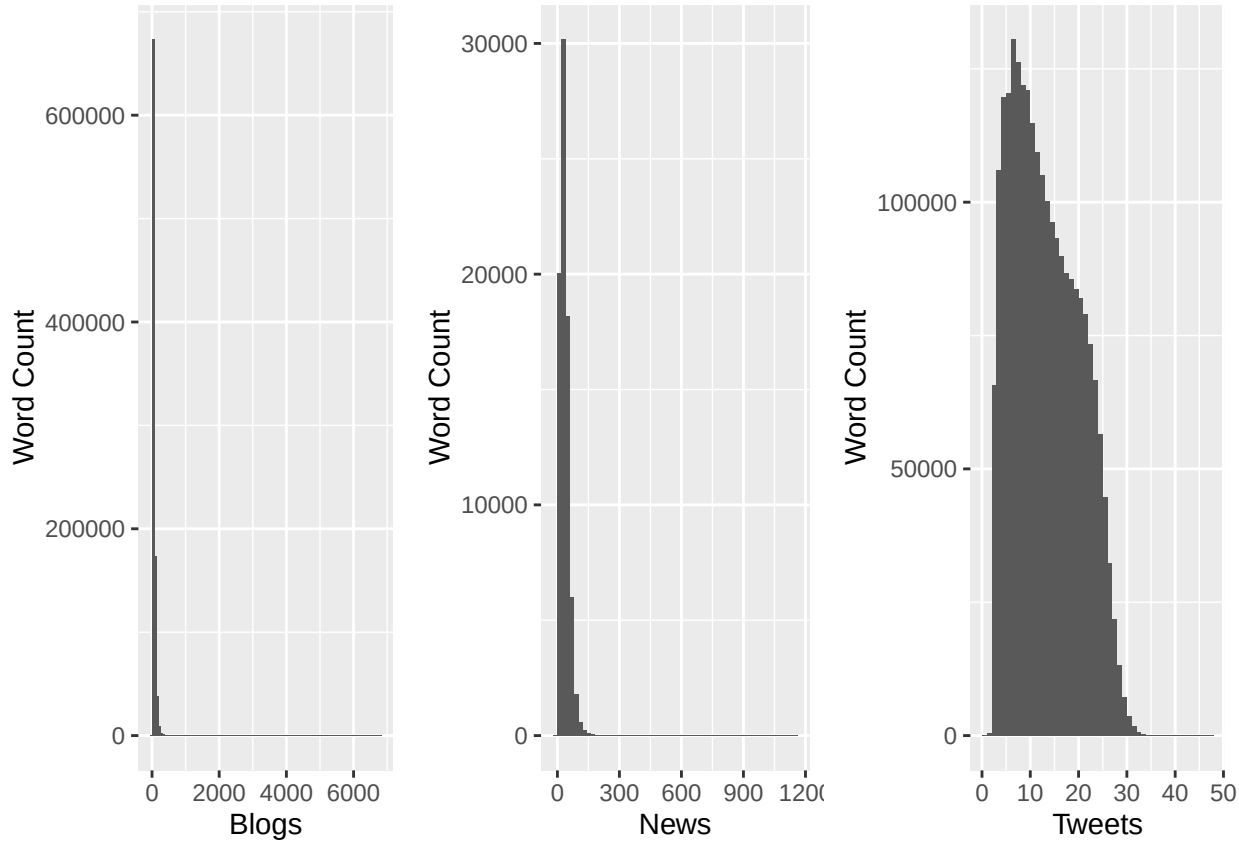
Summary Statistics

- Basic Stats

```
# General String Statistics
blogsStat <- as.data.frame(stri_stats_general(blogsRaw))
newsStat <- as.data.frame(stri_stats_general(newsRaw))
twitterStat <- as.data.frame(stri_stats_general(twitterRaw))
dataStat <- cbind(blogsStat,newsStat,twitterStat)
colnames(dataStat)<-c("Blogs","News","Tweets")
head(format(as.data.frame(dataStat),big.mark=",",scientific=F))
# Distribution of words per line
blogsNrow <- stri_count_words(blogsRaw)
newsNrow <- stri_count_words(newsRaw)
twitterNrow <- stri_count_words(twitterRaw)
blogsSumm <- summary(blogsNrow)
newsSumm <- summary(newsNrow)
twitterSumm <- summary(twitterNrow)
dataSumm <- cbind(blogsSumm,newsSumm,twitterSumm)
colnames(dataSumm)<-c("Blogs","News","Tweets")
head(format(as.data.frame(dataSumm),big.mark=",",scientific=F))
```

```
##                   Blogs       News      Tweets
## Lines           899,288     77,259   2,360,148
## LinesNEmpty     899,288     77,259   2,360,148
## Chars       206,824,382 15,639,408 162,096,241
## CharsNWhite 170,389,539 13,072,698 134,082,806
##            Blogs     News Tweets
## Min.        0.00     1.00   1.00
## 1st Qu.     9.00    19.00   7.00
## Median     28.00    32.00  12.00
## Mean       41.75    34.62  12.75
## 3rd Qu.    60.00    46.00  18.00
## Max.    6,726.00 1,123.00  47.00
```

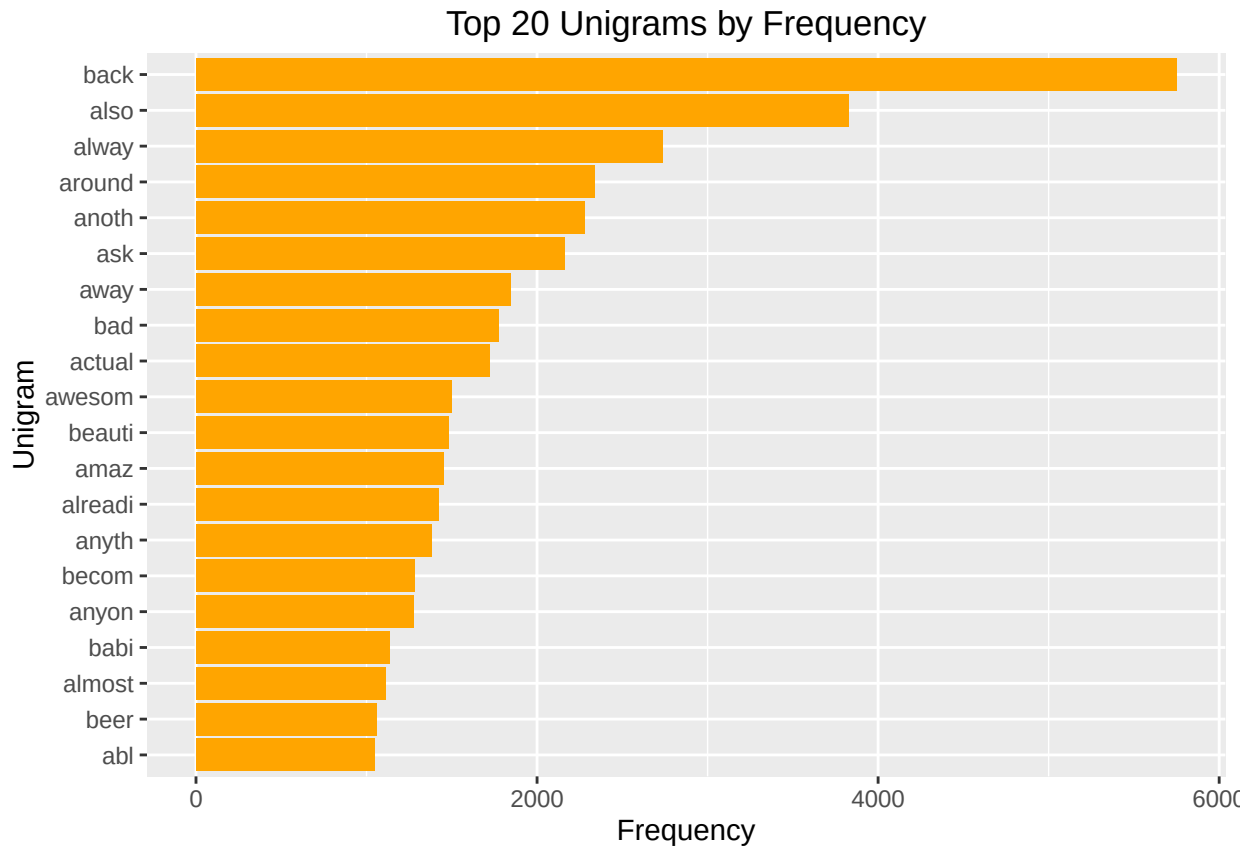
- Histograms of word frequency



Unigram

- Ploting frequency of words on unigram

```
freqOFterm <- findFreqTerms(uniMat, lowfreq = 1000)
termOFfreq <- rowSums(as.matrix(uniMat[freqOFterm,]))
termOFfreq <- data.frame(unigram=names(termOFfreq), frequency=termOFfreq)
termOFfreq[1:20,]

uniPlot <- ggplot(termOFfreq[1:20,], aes(x=reorder(unigram, frequency), y=frequency)) +
  geom_bar(stat = "identity", fill = "orange") +  coord_flip() +
  theme(legend.title=element_blank()) +
  xlab("Unigram") + ylab("Frequency") +
  labs(title = "Top 20 Unigrams by Frequency")
print(uniPlot)
```




```
##          unigram frequency
## abl          abl      1045
## actual    actual      1722
## almost    almost      1113
## alreadi  alreadi      1421
## also        also      3828
## alway      alway      2733
## amaz        amaz      1453
## anoth      anoth      2280
## anyon      anyon      1273
## anyth      anyth      1382
## around    around      2337
## ask          ask      2162
## away        away      1846
## awesom    awesom      1496
## babi        babi      1135
## back        back      5749
## bad          bad      1775
## beauti    beauti      1482
## becom      becom      1281
## beer        beer      1058
```

- Amount of words needed to cover 50% of the instances in Unigram

```
uniSum <- cumsum(termOFfreq$frequency)
uni50 <- sum(termOFfreq$frequency)*0.5
length((uniSum[uniSum <=  uni50]))
```

```
## [1] 157
```

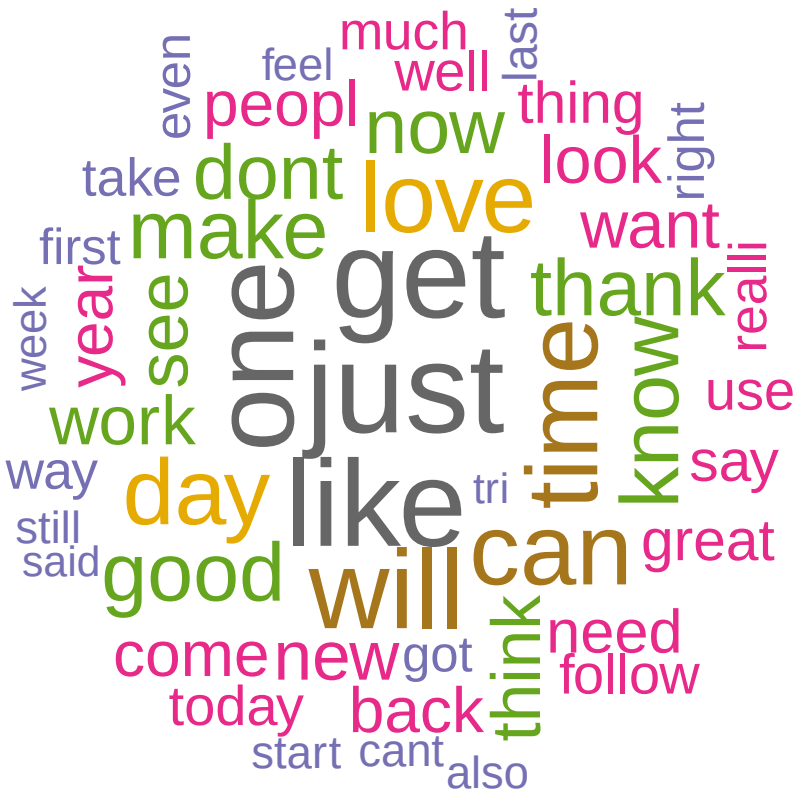
- Amount of words to cover 90% of the instances in Unigram

```
uni90 <- sum(termOFfreq$frequency)*0.9
length((uniSum[uniSum <=  uni90]))
```

```
## [1] 280
```

- Unigram WorldCloud

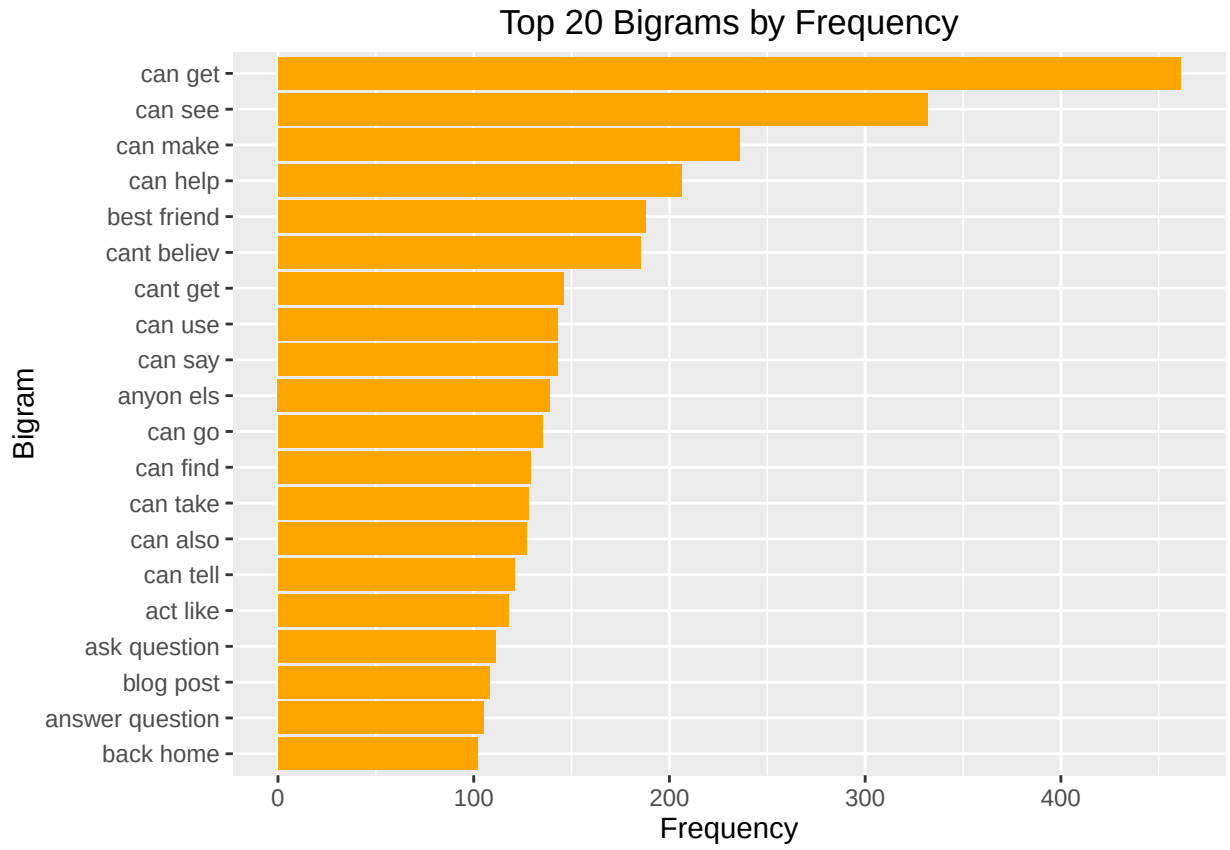
```
uniSort <- as.matrix(uniMat)
uniSort <- sort(rowSums(uniSort),decreasing=TRUE)
uni20 <- data.frame(word = names(uniSort),freq=uniSort)

wordcloud(uni20$word,uni20$freq, scale=c(4,.3),min.freq=2,max.words=50, random.order=F, rot.per=.15, col
```



Bigram

- Ploting frequency of words on bigram

```
freqOFterm <- findFreqTerms(biMat, lowfreq = 100)
termOFfreq <- rowSums(as.matrix(biMat[freqOFterm,]))
termOFfreq <- data.frame(bigram=names(termOFfreq), frequency=termOFfreq)
termOFfreq[1:20,]

biPlot <- ggplot(termOFfreq[1:20,], aes(x=reorder(bigram, frequency), y=frequency)) +
  geom_bar(stat = "identity", fill = "orange") +  coord_flip() +
  theme(legend.title=element_blank()) +
  xlab("Bigram") + ylab("Frequency") +
  labs(title = "Top 20 Bigrams by Frequency")
print(biPlot)
```




```
##                          bigram frequency
## act like               act like       118
## answer question answer question       105
## anyon els             anyon els       139
## ask question       ask question       111
## back home             back home       102
## best friend         best friend       188
## blog post             blog post       108
## can also               can also       127
```

```
## can find               can find       129
## can get                 can get       461
## can go                   can go       135
## can help               can help       206
## can make               can make       236
## can say                 can say       143
## can see                 can see       332
## can take               can take       128
## can tell               can tell       121
## can use                 can use       143
## cant believ         cant believ       185
## cant get               cant get       146
```

- Amount of words needed to cover 50% of the instances in bigram

```
biSum <- cumsum(termOFfreq$frequency)
bi50 <- sum(termOFfreq$frequency)*0.5
length((biSum[biSum <=  bi50]))
```

```
## [1] 130
```

- Amount of words to cover 90% of the instances in bigram

```
bi90 <- sum(termOFfreq$frequency)*0.9
length((biSum[biSum <=  bi90]))
```

```
## [1] 227
```

- Bigram WorldCloud

```
biSort <- as.matrix(biMat)
biSort <- sort(rowSums(biSort),decreasing=TRUE)
bi20 <- data.frame(word = names(biSort),freq=biSort)

wordcloud(bi20$word,bi20$freq, scale=c(4,.3),min.freq=2,max.words=50, random.order=F, rot.per=.15, colo
```

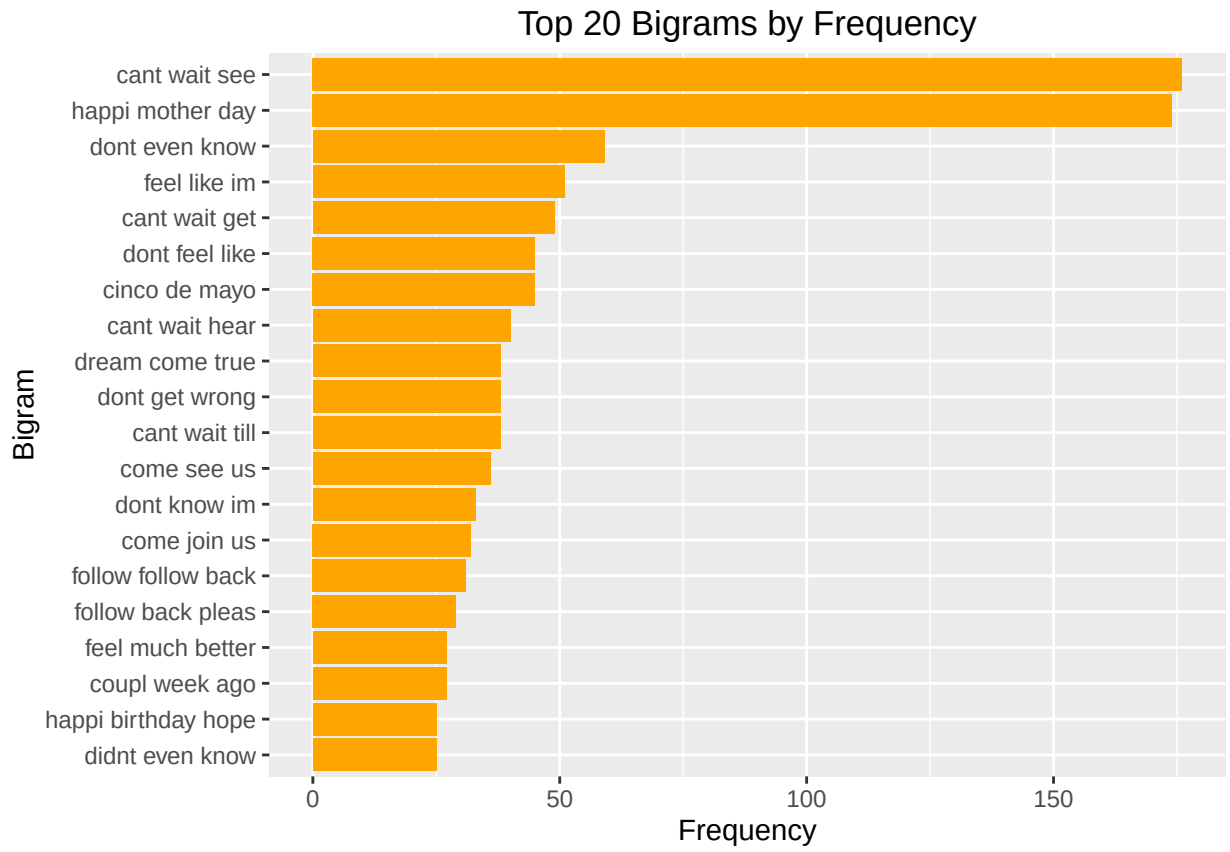
```
## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : year old could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : follow back could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : next week could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : sound like could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : high school could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : even though could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : go back could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : good luck could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : come back could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : new year could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : get readi could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : get back could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : social media could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : thank rt could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : everi time could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : dont get could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : im gonna could not be fit on page. It will not be plotted.

## Warning in wordcloud(bi20$word, bi20$freq, scale = c(4, 0.3), min.freq =
## 2, : one thing could not be fit on page. It will not be plotted.
```

Trigram

- Ploting frequency of words on trigram

```
freqOFterm <- findFreqTerms(triMat, lowfreq = 25)
termOFfreq <- rowSums(as.matrix(triMat[freqOFterm,]))
termOFfreq <- data.frame(trigram=names(termOFfreq), frequency=termOFfreq)
termOFfreq[1:20,]

triPlot <- ggplot(termOFfreq[1:20,], aes(x=reorder(trigram, frequency), y=frequency)) +
  geom_bar(stat = "identity", fill = "orange") +  coord_flip() +
  theme(legend.title=element_blank()) +
  xlab("Bigram") + ylab("Frequency") +
  labs(title = "Top 20 Bigrams by Frequency")
print(triPlot)
```




```
##                                trigram frequency
## cant wait get             cant wait get        49
## cant wait hear           cant wait hear        40
## cant wait see             cant wait see       176
## cant wait till           cant wait till        38
## cinco de mayo             cinco de mayo        45
## come join us               come join us        32
## come see us                 come see us        36
## coupl week ago           coupl week ago        27
## didnt even know         didnt even know        25
## dont even know           dont even know        59
## dont feel like           dont feel like        45
## dont get wrong           dont get wrong        38
## dont know im               dont know im        33
## dream come true         dream come true        38
## feel like im               feel like im        51
## feel much better       feel much better        27
## follow back pleas     follow back pleas        29
## follow follow back   follow follow back        31
## happi birthday hope happi birthday hope        25
## happi mother day       happi mother day       174
```

- Amount of words needed to cover 50% of the instances in trigram

```
triSum <- cumsum(termOFfreq$frequency)
tri50 <- sum(termOFfreq$frequency)*0.5
length((triSum[triSum <=  tri50]))
```

```
## [1] 22
```

- Amount of words to cover 90% of the instances in trigram

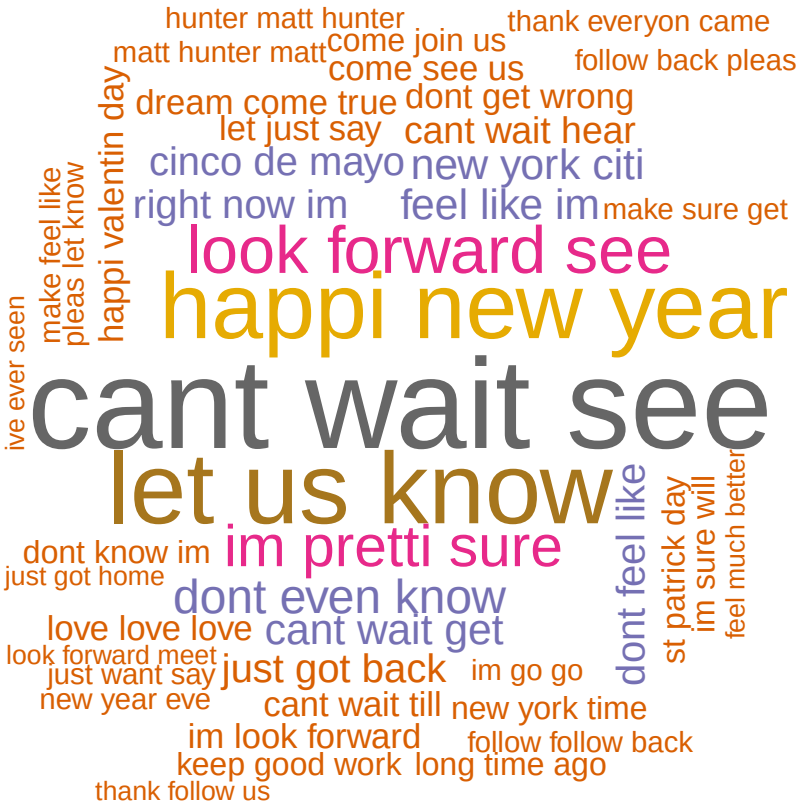
```
tri90 <- sum(termOFfreq$frequency)*0.9
length((triSum[triSum <=  tri90]))
```

```
## [1] 47
```

- Trigram WorldCloud

```
triSort <- as.matrix(triMat)
triSort <- sort(rowSums(triSort),decreasing=TRUE)
tri20 <- data.frame(word = names(triSort),freq=triSort)

wordcloud(tri20$word,tri20$freq, scale=c(4,.3),min.freq=2,max.words=50, random.order=F, rot.per=.15, col
```

```
## Warning in wordcloud(tri20$word, tri20$freq, scale = c(4, 0.3), min.freq =
## 2, : happi mother day could not be fit on page. It will not be plotted.

## Warning in wordcloud(tri20$word, tri20$freq, scale = c(4, 0.3), min.freq =
## 2, : realli look forward could not be fit on page. It will not be plotted.

## Warning in wordcloud(tri20$word, tri20$freq, scale = c(4, 0.3), min.freq =
## 2, : make feel better could not be fit on page. It will not be plotted.

## Warning in wordcloud(tri20$word, tri20$freq, scale = c(4, 0.3), min.freq =
## 2, : coupl week ago could not be fit on page. It will not be plotted.

## Warning in wordcloud(tri20$word, tri20$freq, scale = c(4, 0.3), min.freq =
## 2, : thank much follow could not be fit on page. It will not be plotted.
```



Findings

The set of files used in this project has a considerable size around 500Mb. This taking into account that will be analyzed in personal computers. Because the processing of the file size becomes a time-consuming task and because of that for the realization of the project the option to use a sample of the data became necessary. What one hand improves processing performance, but on the other hand reduces the accuracy. Removing all stopwords from the corpus is recommended, but, of course, stopwords are a fundamental part of languages. Therefore, consideration should be given to include these stop words in the prediction application again. And The line size differ dramatically between blogs/news and twitter data, though file sizes are comparable.

Probable next steps of the Capstone Project

I believe that the objective of this project is to create a prediction application. And for this it is necessary to develop a reliable application, fast and light.