Milestone Report for Data Science Specialization

May 12, 2017

Objective

The goal of this project is just to display that we've gotten used to working with the data and that we are on track to create our prediction algorithm for the most likely next word in a sequence of words. That would be the Capstone of our Data Science Specialization courses - an algorithm behind a Shiny app. The data set can be downloaded from here: Capstone Data (https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip)
The zip file contains the following files:
- en_US.blogs.txt
- en_US.news.txt
- en_US.twitter.txt

The goal of this milestone report 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.

```r
# Required packages
library(tm)
library(ggplot2)
library(stringi)
library(RWeka)
library(wordcloud)
library(RColorBrewer)
```

Import Data Sets

The data sets are downloaded and unzipped. Now we are going to import them in R

```r
# Import news, blogs and twitter files
news <- readLines("./en_US/en_US.news.txt",encoding = "UTF-8", skipNul = TRUE)
blogs <- readLines("./en_US/en_US.blogs.txt",encoding = "UTF-8", skipNul = TRUE)
twitts <- readLines("./en_US/en_US.twitter.txt",encoding = "UTF-8", skipNul = TRUE)
```

Make Summary Statistics of the Data Sets

Here we make a data frame of the summary statistics for each file. We calculate the size in Megabytes, the number of words, the number of lines and the total characters for each file.

```
# Make the data frame
summaryDT <- data.frame(
      DataSet = c("news", "blogs", "twitter"),
      FileSizeMB = c(file.info("./en_US/en_US.news.txt")$size/1024^2,
                     file.info("./en_US/en_US.blogs.txt")$size/1024^2,
                     file.info("./en_US/en_US.twitter.txt")$size/1024^2),
      Words = c(stri_stats_latex(news)["Words"],
                stri_stats_latex(blogs)["Words"],
                stri_stats_latex(twitts)["Words"]),
      Lines = c(stri_stats_general(news)["Lines"],
                stri_stats_general(blogs)["Lines"],
                stri_stats_general(twitts)["Lines"]),
      Chars = c(stri_stats_general(news)["Chars"],
                stri_stats_general(blogs)["Chars"],
                stri_stats_general(twitts)["Chars"])
      )

# Display the data frame
summaryDT
```

```
  DataSet FileSizeMB    Words   Lines     Chars
1    news   196.2775  2651432   77259  15639408
2   blogs   200.4242 37570839  899288 206824382
3 twitter   159.3641 30451170 2360148 162096241
```

Sample and Merge the Data Sets

Data sets are very very large, so we are going to sample 2% of them to work with.

```
# Set seed for reproducibility
set.seed(123)

# Sample the datasets
sNews <- news[sample(1:length(news), 0.02*length(news), replace = FALSE)]
sBlogs <- blogs[sample(1:length(blogs), 0.02*length(blogs), replace = FALSE)]
sTwitter <- twitts[sample(1:length(twitts), 0.02*length(twitts), replace = FALSE)]

# Combine the three data vectors into one
sData <- c(sNews, sBlogs, sTwitter)
```

Build the Corpus and Clean the Corpus

Here we are going to build the corpus necessary for our analysis and we are going to clean in with the means of some functions, which are: removing punctuation, numbers, everything different than letters, we convert every letter to a lower case letter, also we remove the stop words and remove unnecessary white space. I believe removing the stop words isn’t a good idea for the application that we are going to build in the last assignment, but I will do it here for the sake of it.

```
# Make the corpus with the Text Mining package `tm`
corpusDO <- VCorpus(VectorSource(sData))

# Make a copy for later comparison
corpusD <- corpusDO

# Make a custom function to remove non-alpha characters to clean the corpus
nonalphasF <- function(corpus) { corpus <- gsub("[^a-zA-Z]"," ",corpus)}

# Clean the corpus
corpusD <- tm_map(corpusD, removePunctuation)
corpusD <- tm_map(corpusD, removeNumbers)
corpusD <- tm_map(corpusD, content_transformer(nonalphasF))
corpusD <- tm_map(corpusD, content_transformer(tolower))
## I believe removing the stop words isn't a nessesary step for the future applicatio
n, so you can skip it.
corpusD <- tm_map(corpusD, removeWords, stopwords("english"))
corpusD <- tm_map(corpusD, stripWhitespace)

# show some line of text before cleaning
corpusDO[[111]][1]
```

```
## $content
## [1] "Malik High, 23, of Hoboken, was arrested and charged with unlawful entry of s
tructure as well as possession and manufacturing or distribution of crack-cocaine aft
er officers spotted him walking out of an HHA building near Second and Harrison stree
ts around 8:50 p.m. on Monday, reports said."
```

```
# show some line of text after cleaning
corpusD[[111]][1]
```

```
## $content
## [1] "malik high hoboken arrested charged unlawful entry structure well possession
manufacturing distribution crackcocaine officers spotted walking hha building near se
cond harrison streets around pm monday reports said"
```

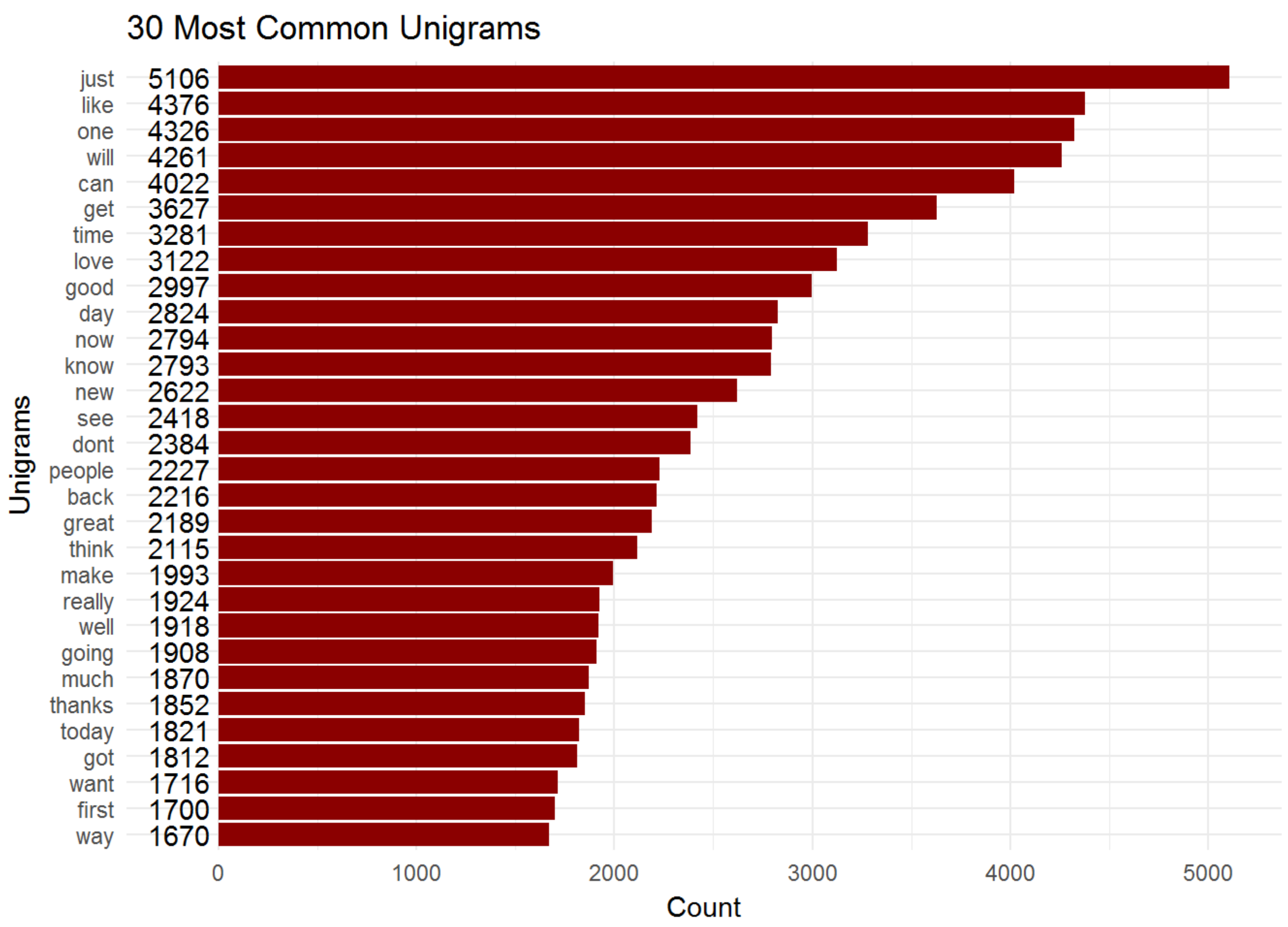
Tokenize and Calculate Frequencies of N-Grams

Unigrams

We are going to display the Frequencies of the top 30 individual words /Unigrams/

```r
# build the 1-Gram function
unigram <- function(x) NGramTokenizer(x, Weka_control(min = 1, max = 1))
# Create Term Document Matrix for 1-grams
td1 <- TermDocumentMatrix(corpusD, control = list(tokenize = unigram))
# Remove sparse elements
td1s <- removeSparseTerms(td1, 0.99)
# Create a frequency matrix and sort it out by decreasing frequency
td1sm <- sort(rowSums(as.matrix(td1s)), decreasing = T)
# Convet to a dataframe
td1smF <- data.frame(Word = names(td1sm), Frequency = td1sm)
# Reorder words based on the frequency
td1smF$Word <- reorder(td1smF$Word, td1smF$Frequency)

# Make the plot
plot1 <-
      ggplot(data = td1smF[1:30, ], aes(x = Word, y = Frequency)) +
            geom_bar(stat = "identity", fill = "darkred") +
            coord_flip() +
            ggtitle("30 Most Common Unigrams") +
            xlab("Unigrams") +
            ylab("Count") +
            geom_text(data = td1smF[1:30, ],
                      aes(Word, y = -200, label = Frequency),
                      vjust = 0.5) +
            theme_minimal()
# Display plot1
plot1
```

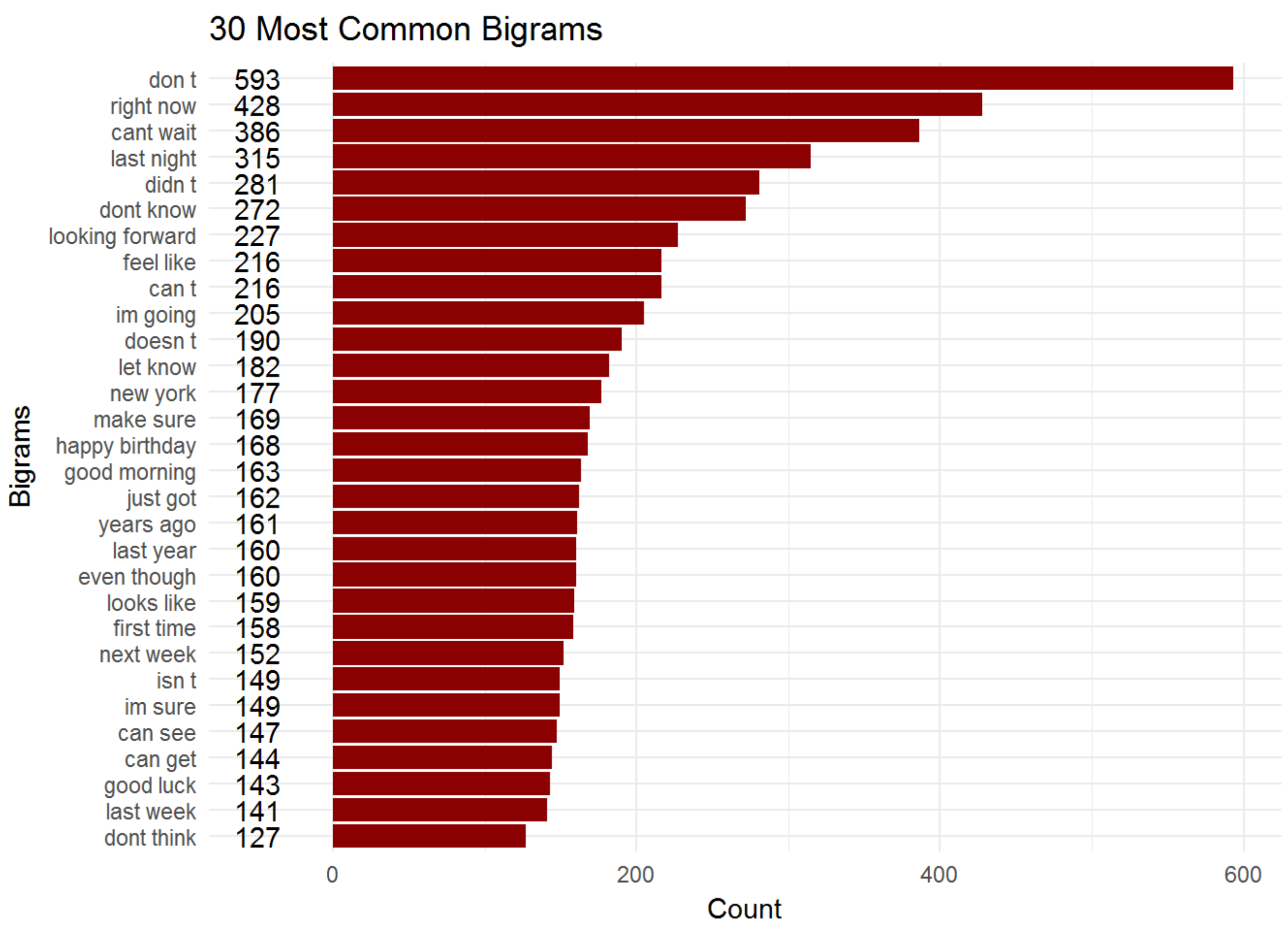


Bigrams

We are going to display the Frequencies of the top 30 pairs of words /Bigrams/

```r
# build the 2-Gram function
bigram  <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
# Create Term Document Matrix for the 2-grams
td2 <- TermDocumentMatrix(corpusD, control = list(tokenize = bigram))
# Remove sparse elements
td2s <- removeSparseTerms(td2, 0.999)
# Create a frequency matrix and sort it out by decreasing frequency
td2sm <- sort(rowSums(as.matrix(td2s)), decreasing = T)
# Convet to a dataframe
td2smF <- data.frame(Word = names(td2sm), Frequency = td2sm)
# Reorder words based on the frequency
td2smF$Word <- reorder(td2smF$Word, td2smF$Frequency)

# Make the plot
plot2 <-
      ggplot(data = td2smF[1:30, ], aes(x = Word, y = Frequency)) +
            geom_bar(stat = "identity", fill = "darkred") +
            coord_flip() +
            ggtitle("30 Most Common Bigrams") +
            xlab("Bigrams") +
            ylab("Count") +
            geom_text(data = td2smF[1:30, ],
                      aes(Word, y = -50, label = Frequency),
                      vjust = 0.5) +
            theme_minimal()
# Display plot2
plot2
```

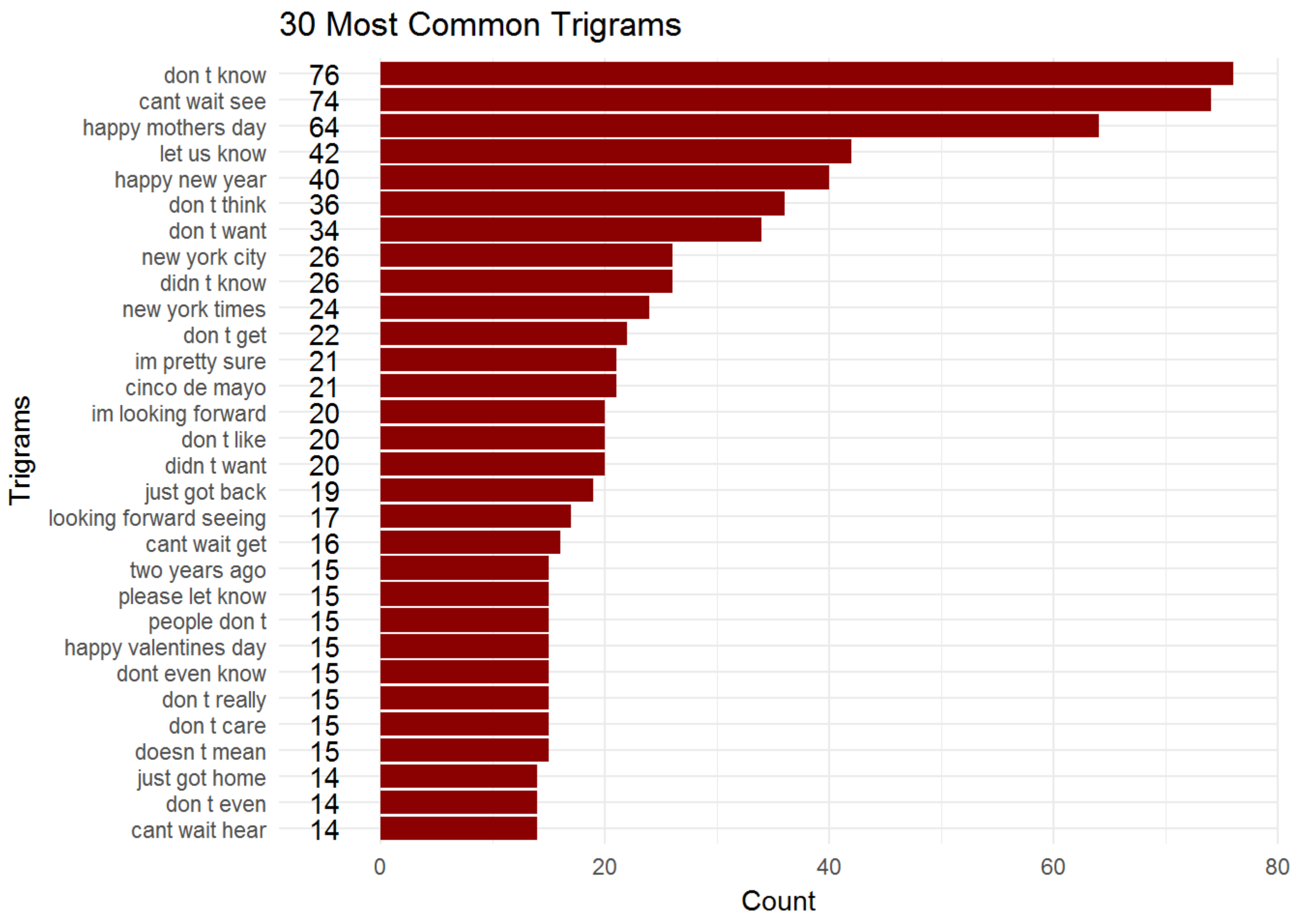


Trigrams

We are going to display the Frequencies of the top 30 triplets of words /Trigrams/

```r
# build the 3-Gram function
trigram <- function(x) NGramTokenizer(x, Weka_control(min = 3, max = 3))
# Create Term Document Matrix for the 3-grams
td3 <- TermDocumentMatrix(corpusD, control = list(tokenize = trigram))
# Remove sparse elements
td3s <- removeSparseTerms(td3, 0.9999)
# Create a frequency matrix and sort it out by decreasing frequency
td3sm <- sort(rowSums(as.matrix(td3s)), decreasing = T)
# Convet to a dataframe
td3smF <- data.frame(Word = names(td3sm), Frequency = td3sm)
# Reorder words based on the frequency
td3smF$Word <- reorder(td3smF$Word, td3smF$Frequency)

# Make the plot
plot3 <-
      ggplot(data = td3smF[1:30, ], aes(x = Word, y = Frequency)) +
             geom_bar(stat = "identity", fill = "darkred") +
             coord_flip() +
             ggtitle("30 Most Common Trigrams") +
             xlab("Trigrams") +
             ylab("Count") +
             geom_text(data = td3smF[1:30, ],
                       aes(Word, y = -5, label = Frequency),
                       vjust = 0.5) +
             theme_minimal()
# Display plot2
plot3
```

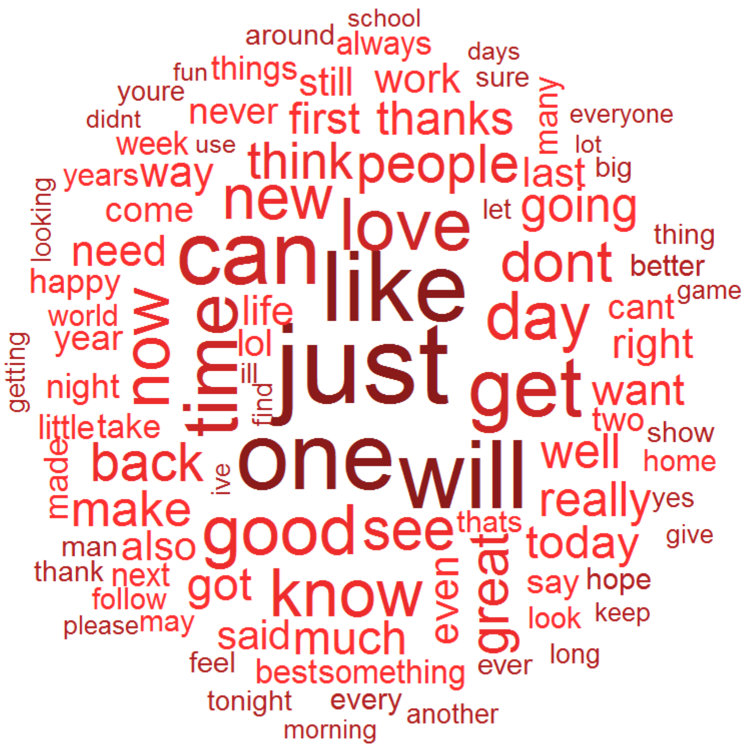


WordClouds

This is another way to visualize the frequency of words

Unigrams

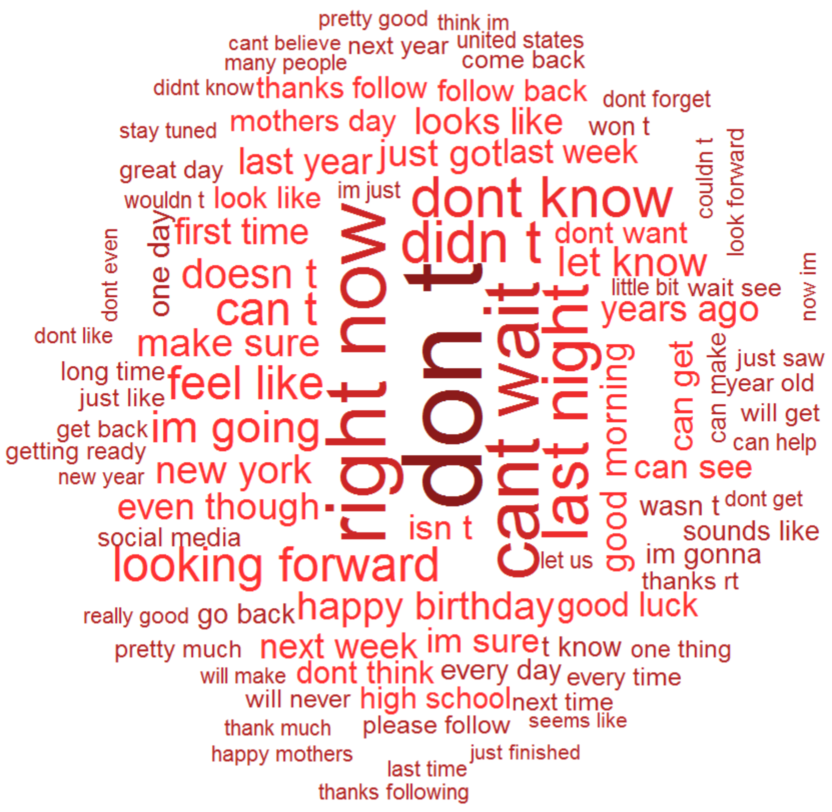
```
# make word cloud for unigrams
wordcloud(word = td1smF$Word, freq = td1smF$Frequency, scale = c(3.5,.3),
          max.words = 100, random.order = FALSE, random.color = FALSE, colors = c("fi
rebrick", "firebrick1","firebrick2","firebrick3","firebrick4"))
```



Unigrams /individual words/

Bigrams

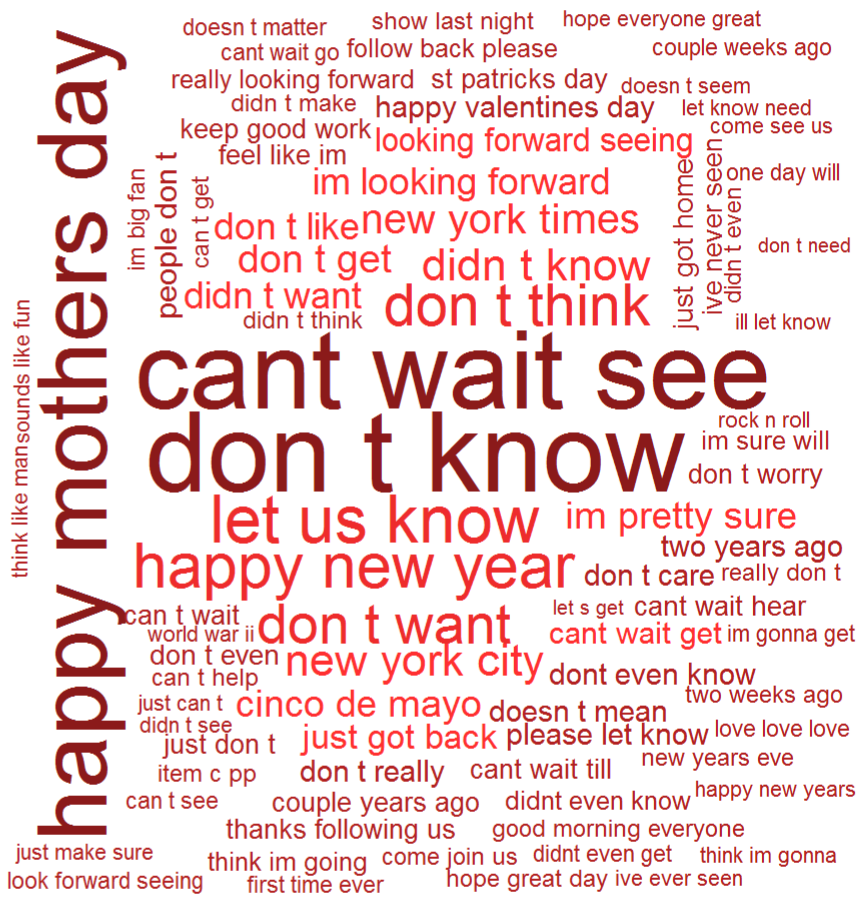
```
# make word cloud for bigrams
wordcloud(word = td2smF$Word, freq = td2smF$Frequency, scale = c(3.5,.3),
          max.words = 100, random.order = FALSE, random.color = FALSE, colors = c("fi
rebrick", "firebrick1","firebrick2","firebrick3","firebrick4"))
```



Bigrams /pairs of words/

Trigrams

```
# make word cloud for trigrams
wordcloud(word = td3smF$Word, freq = td3smF$Frequency, scale = c(3.5,.3),
          max.words = 100, random.order = FALSE, random.color = FALSE, colors = c("fi
rebrick", "firebrick1","firebrick2","firebrick3","firebrick4"))
```


Trigrams /triplets of words/

Future Plans

The plan for our Shiny app in which the user can enter a word or a sequence of words and based on our prediction model he/she will be provided with a list of possible next words. That will be based on conditional probabilities. It will use n-gram model analysis. A lot of testing and tweaking is required before the finalized model is ready.