Coursera Data Science Capstone: Exploratory Data Analysis

Venkatesh Attinti

5/26/2020

Executive Summary

This milestone report is based on the exploratory data analysis of the swift key data provided in context of Data science capstone project.The data consist of 3 data file from different sources - (twitter,blogs,news).This report showcases the tidytext approach used for data analysis.More information regarding the tidy text approach can be accessed from here https://www.tidytextmining.com/

Data Summary

It is assumed that the data from https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip is downloaded, unziped and available in the working directory.

Below is the summary of the data loaded

```
library("stringi")

twitterRawData <- readLines("en_US.twitter.txt",warn=FALSE,encoding="UTF-8")
blogsRawData <- readLines("en_US.blogs.txt",warn=FALSE,encoding="UTF-8")
newsRawData <- readLines("en_US.news.txt",warn=FALSE,encoding="UTF-8")

datasummary <- data.frame("FileNames" =c("Twitter","Blogs","News"),
                          "FileSize"=c(format(object.size(twitterRawData), units = "MB", standard ="auto
                                       format(object.size(blogsRawData), units = "MB", standard = "auto
                                       format(object.size(newsRawData), units = "MB", standard = "auto")
                          "FileLength"=c(length(twitterRawData),length(blogsRawData),length(newsRawData
                          "Wordcount"=c(sum(stri_stats_latex(twitterRawData)[4]),sum(stri_stats_latex(bl
                          "NoOfChars"=c(sum(nchar(twitterRawData)),sum(nchar(blogsRawData)),sum(nchar(n

datasummary
```

```
##   FileNames FileSize FileLength Wordcount NoOfChars
## 1   Twitter   319 Mb    2360148  30451128 162096031
## 2     Blogs 255.4 Mb     899288  37570839 206824505
## 3      News  19.8 Mb      77259   2651432  15639408
```

Exploratory Data Analysis

In this section we will perform some exploratory data analysis using tidy data principles which is a powerful way to make handling data easier and more effective.

we will perform this analysis on the sample data set which is 2% of the original dataset.

Below are packages requried to perform this analysis library("tidyr") library("dplyr") library("tidytext") library("tm") library("openNLP") library("RWeka") library("tm")

```
# Remove all non english characters as they cause issues down the road
twitterRawData <- iconv(twitterRawData, "latin1", "ASCII", sub="")
blogsRawData <- iconv(blogsRawData, "latin1", "ASCII", sub="")
newsRawData <- iconv(newsRawData, "latin1", "ASCII", sub="")

#sampling of the data set

twitterRawData_sample<- sample(twitterRawData,length(twitterRawData)*0.02)
blogsRawData_sample<- sample(blogsRawData,length(blogsRawData)*0.02)
newsRawData_sample<- sample(newsRawData,length(newsRawData)*0.02)

#write the sample files

dir.create("sampleDatafiles", showWarnings = FALSE)

write(twitterRawData_sample, "sampleDatafiles/twitterRawData_sample.txt")
write(blogsRawData_sample, "sampleDatafiles/blogsRawData_sample.txt")
write(newsRawData_sample, "sampleDatafiles/newsRawData_sample.txt")


remove(twitterRawData)
remove(blogsRawData)
remove(newsRawData)
```

Merging the sample data files into single corpus and then converting to tibble data frame format which will be used in all the further analysis

```
library("tidyr")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("tidytext")
library("tm")
```

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

```
library("openNLP")
library("RWeka")
library("tm")
finalSampleData <- c(twitterRawData_sample,blogsRawData_sample,newsRawData_sample)
sampleData <- tibble(text = finalSampleData)
```

Pre-processing the data(invloves operations like removing the whitespaces, punctuation,stopwords, stemming etc) In tidy text the punctuations and converting to lower cases are automatically done during the unnesting tokens.

Unigrams

Unnesting tokens and removing the stopwords

```
data(stop_words)
tidySampleData <- sampleData %>% unnest_tokens(word, text) %>% anti_join(stop_words)
```

```
## Joining, by = "word"
```

```
#Removing whitespaces
tidySampleData$word <- gsub("\\s+","",tidySampleData$word)
#Removing Numbers
tidySampleData<-tidySampleData[-grep("\\b\\d+\\b", tidySampleData$word),]

tidySampleData %>% count(word, sort = TRUE);
```

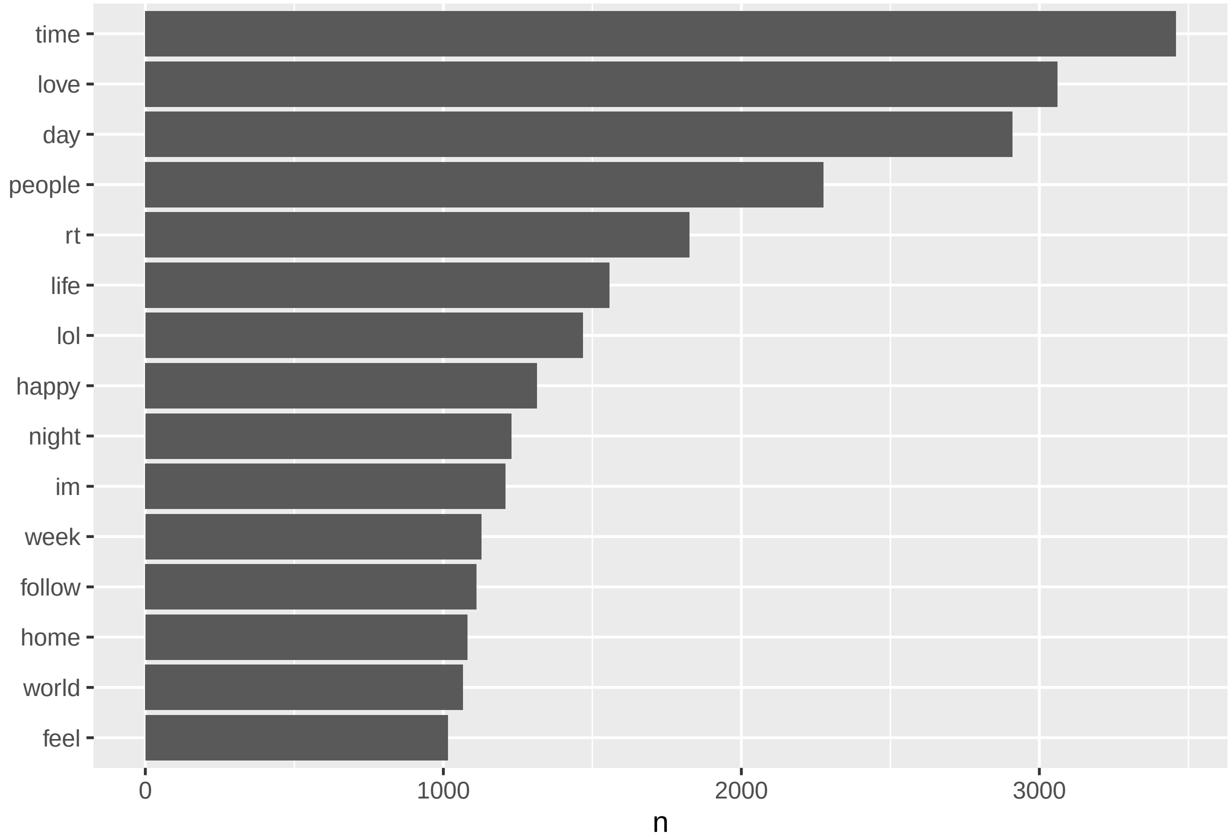
```
## # A tibble: 61,200 x 2
##    word       n
##    <chr>  <int>
##  1 time    3458
##  2 love    3061
##  3 day     2910
##  4 people  2276
##  5 rt      1826
##  6 life    1558
##  7 lol     1468
##  8 happy   1314
##  9 night   1228
## 10 im      1209
## # ... with 61,190 more rows
```

using tidy tools,the word counts are stored in a tidy data frame.This allows us to pipe directly to the ggplot2 package

```
library(ggplot2)
```

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

```
tidySampleData %>% count(word, sort = TRUE) %>% filter(n > 1000) %>% mutate(word = reorder(word, n)) %>
```

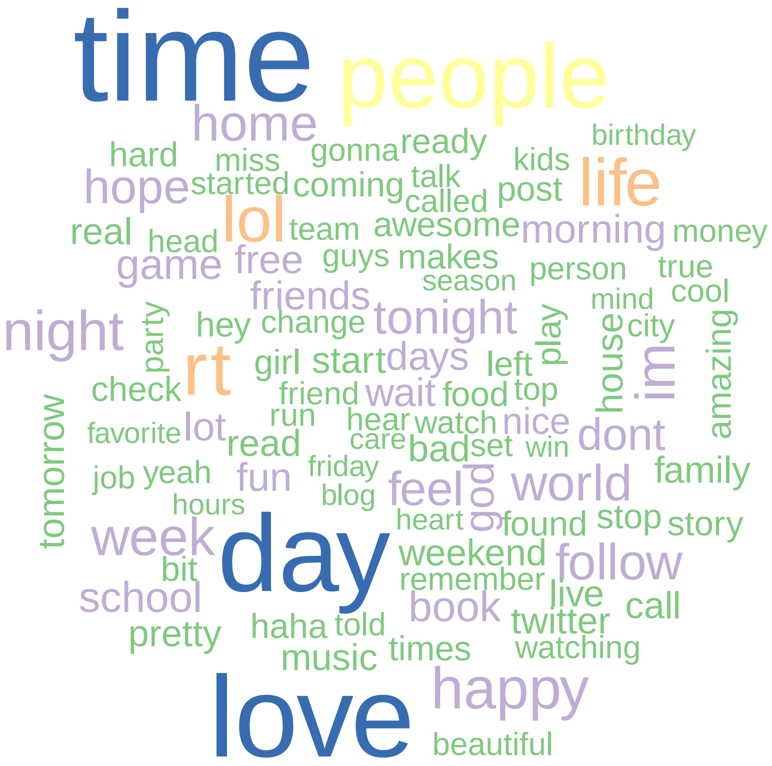


Displaying the most common unnigrams using wordcloud

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library("RColorBrewer")
dark2 <- brewer.pal(5, "Accent")

tidySampleData %>% count(word) %>% with(wordcloud(word, n, max.words = 100, rot.per=0.1, colors=dark2))
```

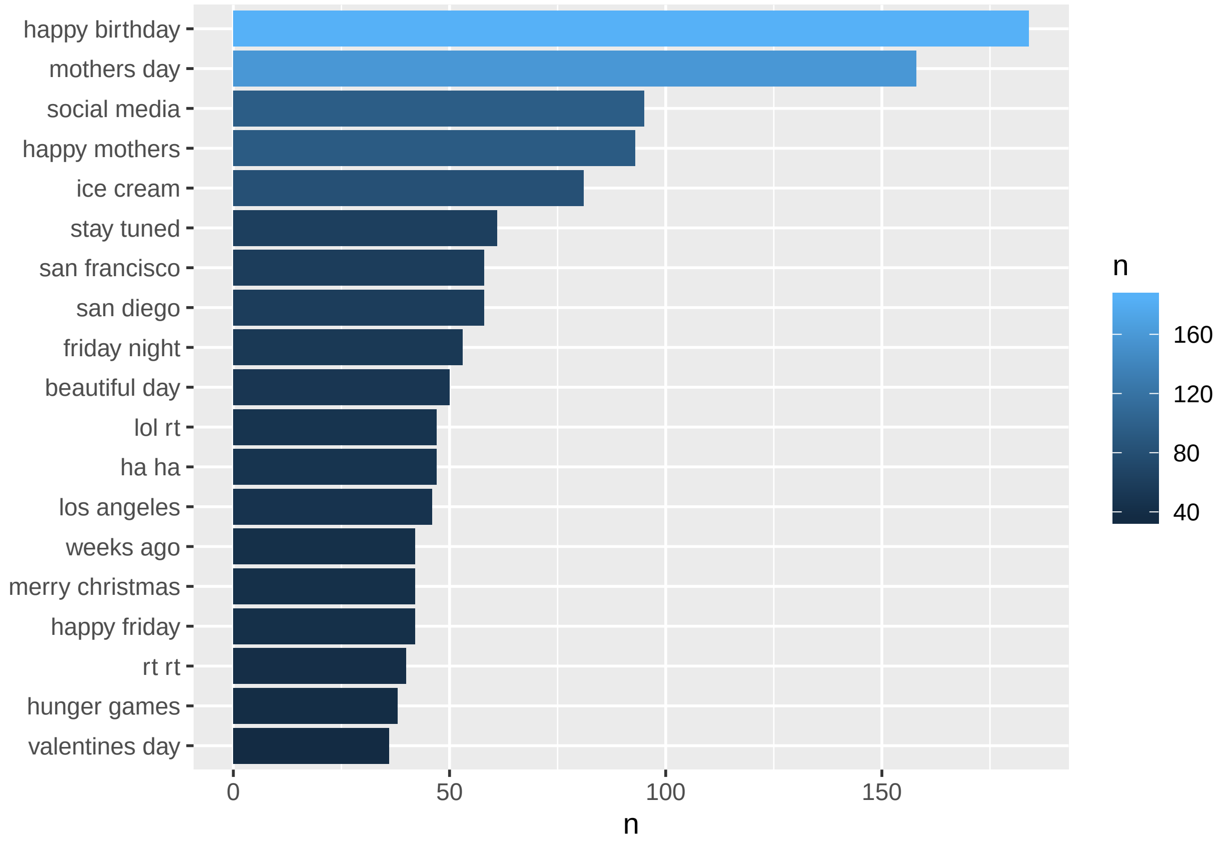


Bigrams – Tokenizing by 2-gram

```
library(tidyr)
stop_words <- rbind(stop_words,data.frame(word="amp",lexicon=""))
tidyBigramSampleData <- sampleData %>% unnest_tokens(bigram, text,token = "ngrams", n = 2)
#Seperating the bigram
tidyBigramSampleData_separated <- tidyBigramSampleData %>% separate(bigram, c("word1", "word2"), sep =
bigrams_filtered <- tidyBigramSampleData_separated %>% filter(!word1 %in% stop_words$word) %>% filter(!
#Removing whitespaces
bigrams_filtered$word1 <- gsub("\\s+","",bigrams_filtered$word1)
bigrams_filtered$word2 <- gsub("\\s+","",bigrams_filtered$word2)

bigrams_filtered$word1 <- gsub("\\'+","",bigrams_filtered$word1)
bigrams_filtered$word2 <- gsub("\\'+","",bigrams_filtered$word2)

#Removing Numbers
bigrams_filtered<-bigrams_filtered[-grep("\\b\\d+\\b", bigrams_filtered$word1),]
bigrams_filtered<-bigrams_filtered[-grep("\\b\\d+\\b", bigrams_filtered$word2),]

bigrams_united <- bigrams_filtered %>% unite(bigram, word1, word2, sep = " ")
bigrams_united %>% count(bigram,sort=TRUE)
```

```
## # A tibble: 172,552 x 2
##    bigram              n
##    <chr>           <int>
##  1 happy birthday    184
##  2 mothers day       158
##  3 social media       95
##  4 happy mothers      93
##  5 ice cream          81
##  6 stay tuned         61
##  7 san diego          58
##  8 san francisco      58
##  9 friday night       53
## 10 beautiful day      50
## # ... with 172,542 more rows
```

```
bigrams_united %>% count(bigram, sort = TRUE) %>% filter(n > 35) %>% mutate(bigram = reorder(bigram, n)
```



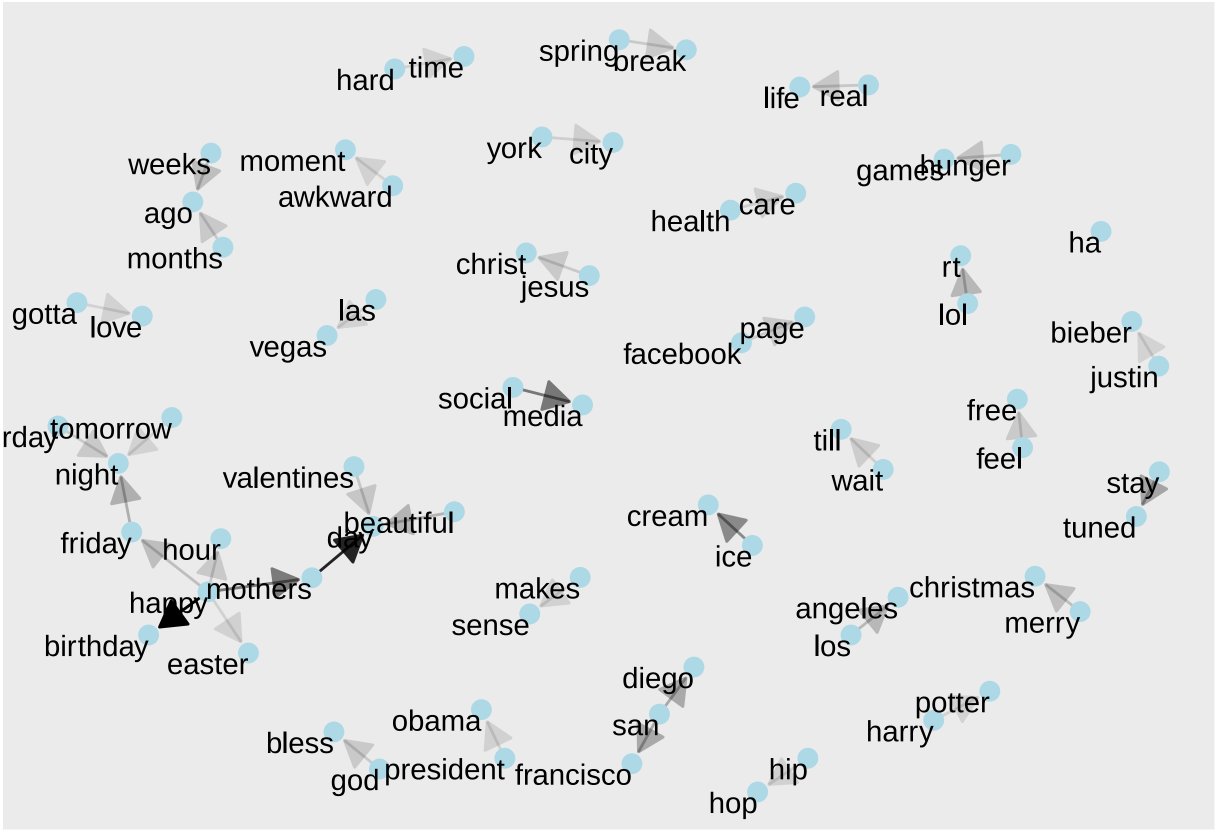
Visualizing a Network of Bigrams with ggraph

It may be interested in visualizing all of the relationships among words simultaneously,rather than just the top few at a time. As one common visualization, we can arrange the words into a network, or "graph."

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following object is masked from 'package:tidyr':
##
##     crossing

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
bigram_graph <- bigrams_filtered %>% count(word1, word2,sort=TRUE) %>% filter(n > 25) %>% graph_from_da
library(ggraph)
set.seed(123456)
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))
ggraph(bigram_graph, layout = "fr") +
geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
arrow = a, end_cap = circle(.07, 'inches')) +
geom_node_point(color = "lightblue",size=3) +
geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```

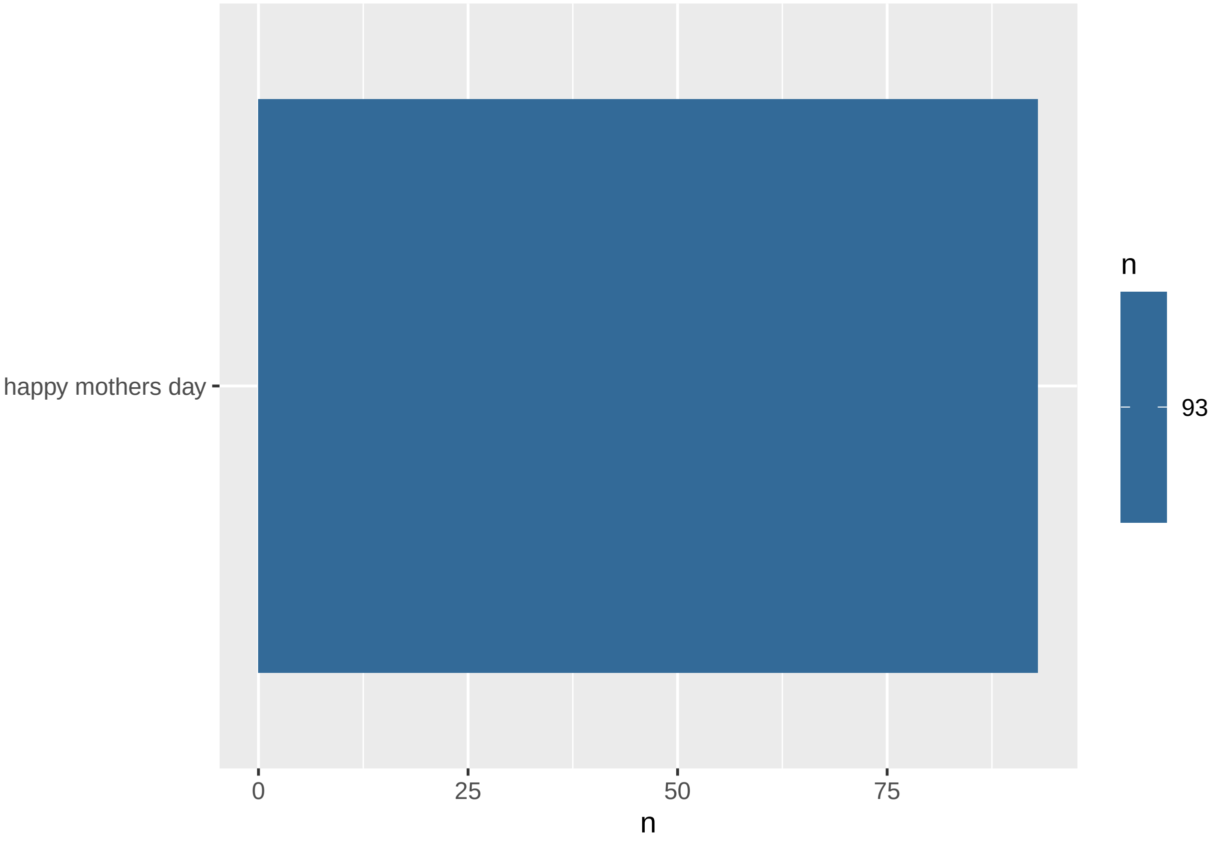


Trigrams – Tokenizing by 3-gram

```
library(tidyr)
stop_words <- rbind(stop_words,data.frame(word="amp",lexicon=""))
tidyBigramSampleData <- sampleData %>% unnest_tokens(bigram, text,token = "ngrams", n = 3)
#Seperating the bigram
tidyBigramSampleData_separated <- tidyBigramSampleData %>% separate(bigram, c("word1", "word2", "word3"
bigrams_filtered <- tidyBigramSampleData_separated %>% filter(!word1 %in% stop_words$word) %>% filter(!
#Removing whitespaces
bigrams_filtered$word1 <- gsub("\\s+","",bigrams_filtered$word1)
bigrams_filtered$word2 <- gsub("\\s+","",bigrams_filtered$word2)
bigrams_filtered$word3 <- gsub("\\s+","",bigrams_filtered$word3)

bigrams_filtered$word1 <- gsub("\\'+","",bigrams_filtered$word1)
bigrams_filtered$word2 <- gsub("\\'+","",bigrams_filtered$word2)
bigrams_filtered$word3 <- gsub("\\'+","",bigrams_filtered$word3)

#Removing Numbers
bigrams_filtered<-bigrams_filtered[-grep("\\b\\d+\\b", bigrams_filtered$word1),]
bigrams_filtered<-bigrams_filtered[-grep("\\b\\d+\\b", bigrams_filtered$word2),]
bigrams_filtered<-bigrams_filtered[-grep("\\b\\d+\\b", bigrams_filtered$word3),]

bigrams_united <- bigrams_filtered %>% unite(bigram, word1, word2,word3, sep = " ")
bigrams_united %>% count(bigram,sort=TRUE)
```

```
## # A tibble: 75,934 x 2
##    bigram                         n
##    <chr>                      <int>
##  1 happy mothers day             93
##  2 cinco de mayo                 19
##  3 coffee coffee coffee          18
##  4 omg omg omg                   18
##  5 st patricks day               16
##  6 ass ass ass                   13
##  7 cake cake cake                12
##  8 greenville newspaper south    12
##  9 ha ha ha                      12
## 10 happy valentines day          12
## # ... with 75,924 more rows
```

```
bigrams_united %>% count(bigram, sort = TRUE) %>% filter(n > 20) %>% mutate(bigram = reorder(bigram, n)
```



Plan of next steps

I have done the exploratory analysis. The next steps of this capstone project would be to finalize our predictive algorithm, and deploy our algorithm using shiny() app. As for the Shiny app it will consist of a simple user interface that will allow a user to enter text into a single textbox.