Milestone-Report

Mohammad Ali Shaaban

January 8, 2019

Introductions

The goal of this project is just to display that you've gotten used to working with the data and that you are on track to create your prediction algorithm. Please submit a report on R Pubs (http://rpubs.com/) that explains your exploratory analysis and your goals for the eventual app and algorithm. This document should be concise and explain only the major features of the data you have identified and briefly summarize your plans for creating the prediction algorithm and Shiny app in a way that would be understandable to a non-data scientist manager. You should make use of tables and plots to illustrate important summaries of the data set. The motivation for this project 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.

Downloading & Loading Data :

```
library(tidytext)

## Warning: package 'tidytext' was built under R version 3.5.2

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(ggplot2)
library(qdapRegex)

## Warning: package 'qdapRegex' was built under R version 3.5.2

##
## Attaching package: 'qdapRegex'

## The following object is masked from 'package:ggplot2':
##
##     %+%

## The following object is masked from 'package:dplyr':
##
##     explain

wd <- getwd()
f <- "Coursera-SwiftKey.zip"
if (!dir.exists("Coursera-SwiftKey")) {
    url <- "https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Courser
a-SwiftKey.zip"
    if (!file.exists("Coursera-SwiftKey.zip")) {
        download.file(url, destfile = "Coursera-SwiftKey.zip", method = #"cur
l")
        unzip("Coursera-SwiftKey.zip")
        download.file(url, file.path(wd, f), method = "curl")
        unzip(f)
    }
}
```

basic report of summary statistics

Reading blogs, twitter, and news files

```
blogs_file<-"../en_US/en_US.blogs.txt"
twitter_file<-"../en_US/en_US.twitter.txt"
news_file<-"../en_US/en_US.news.txt"

blogs <- readLines(blogs_file, warn = FALSE, encoding = "UTF-8", skipNul = TR
UE)
twitter <- readLines(twitter_file, warn = FALSE, encoding = "UTF-8", skipNul
= TRUE)
news <- readLines(news_file, warn = FALSE, encoding = "UTF-8", skipNul = TRUE
)
```

Summarizing the number of lines and words:

blogs:

```
## blogs number of lines
length(blogs)

## [1] 899288

## blogs number of words
sum(sapply(gregexpr("[[:alpha:]]+", blogs), function(x) sum(x > 0)))

## [1] 37873860
```

twitter:

```
## twitter number of lines
length(twitter)

## [1] 2360148

## twitter number of words
sum(sapply(gregexpr("[[:alpha:]]+", twitter), function(x) sum(x > 0)))

## [1] 30555611
```

news:

```
## news number of lines
length(news)

## [1] 77259

## news number of words
sum(sapply(gregexpr("[[:alpha:]]+", news), function(x) sum(x > 0)))

## [1] 2662070
```

Exploratory Analysis:–

sample of data: 5% of each dataset:

```
set.seed(12345)
twitter_sample  <- sample(twitter, length(twitter) * 0.05, replace = FALSE)
blogs_sample    <- sample(blogs, length(blogs) * 0.05, replace = FALSE)
news_sample     <- sample(news, length(news) * 0.05, replace = FALSE)
data_sample = c(twitter_sample, blogs_sample, news_sample)
```

Cleaning Data:

```
# 1. Remove lines with unidentify characters
NotKnown <- grep("NotKnown", iconv(data_sample, "latin1", "ASCII", sub="NotKn
```

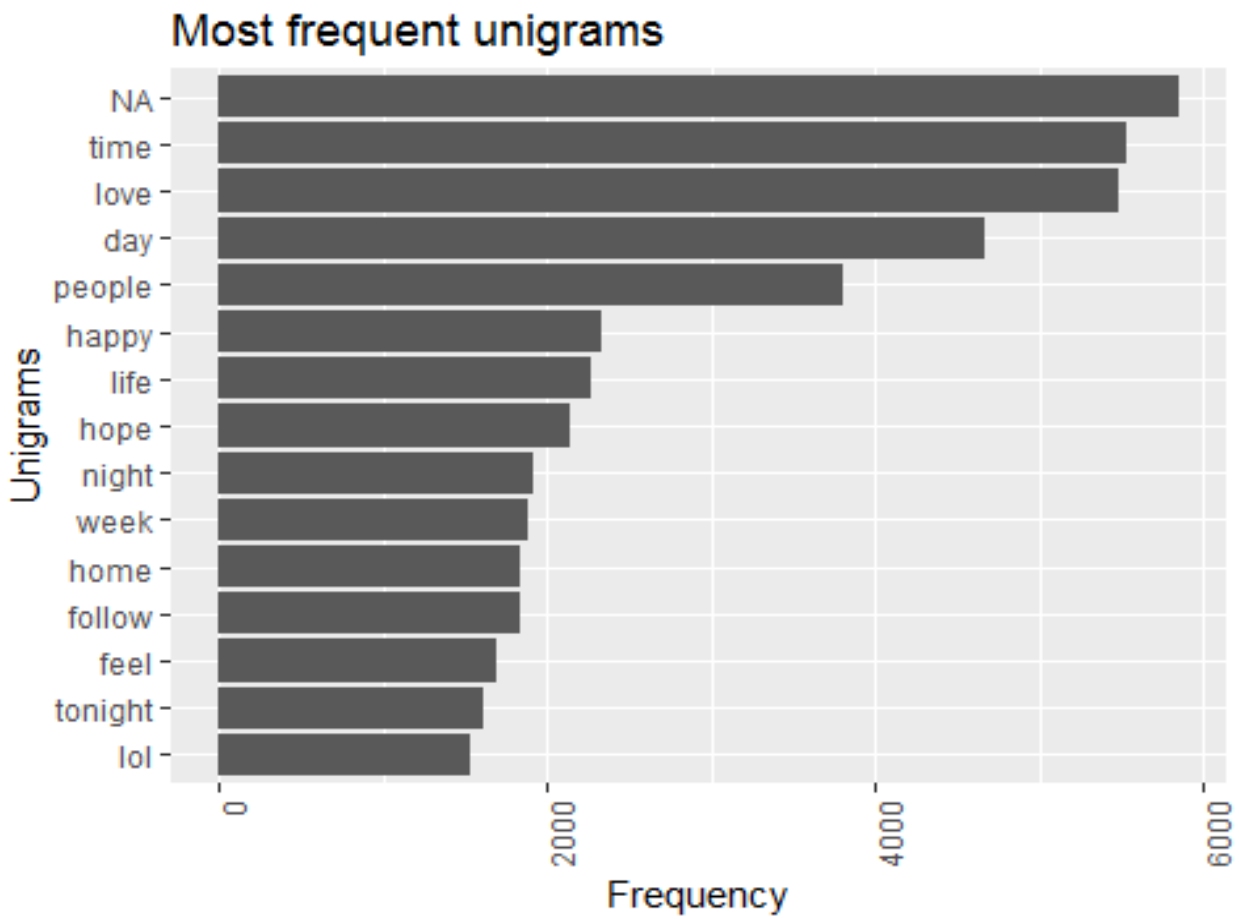
```
own"))
data_sample <- data_sample[-NotKnown]
# doing some simple cleaning
data_sample <- gsub("&amp", "", data_sample)
data_sample <- gsub("RT :|@[a-z,A-Z]*: ", "", data_sample) # remove tweets
data_sample <- gsub("@\\w+", "", data_sample)
data_sample <- gsub("[[:digit:]]", "", data_sample) # remove digits
data_sample <- gsub(" #\\S*","", data_sample)  # remove hash tags
data_sample <- gsub(" ?(f|ht)tp(s?)://(.*)[.][a-z]+", "", data_sample) # remo
ve url
data_sample <- rm_white(data_sample) # remove extra spaces
```

dataframe for n-gram analysis:

```
data_sample_df <- data_frame(line = 1:length(data_sample),
                             text = data_sample)
```

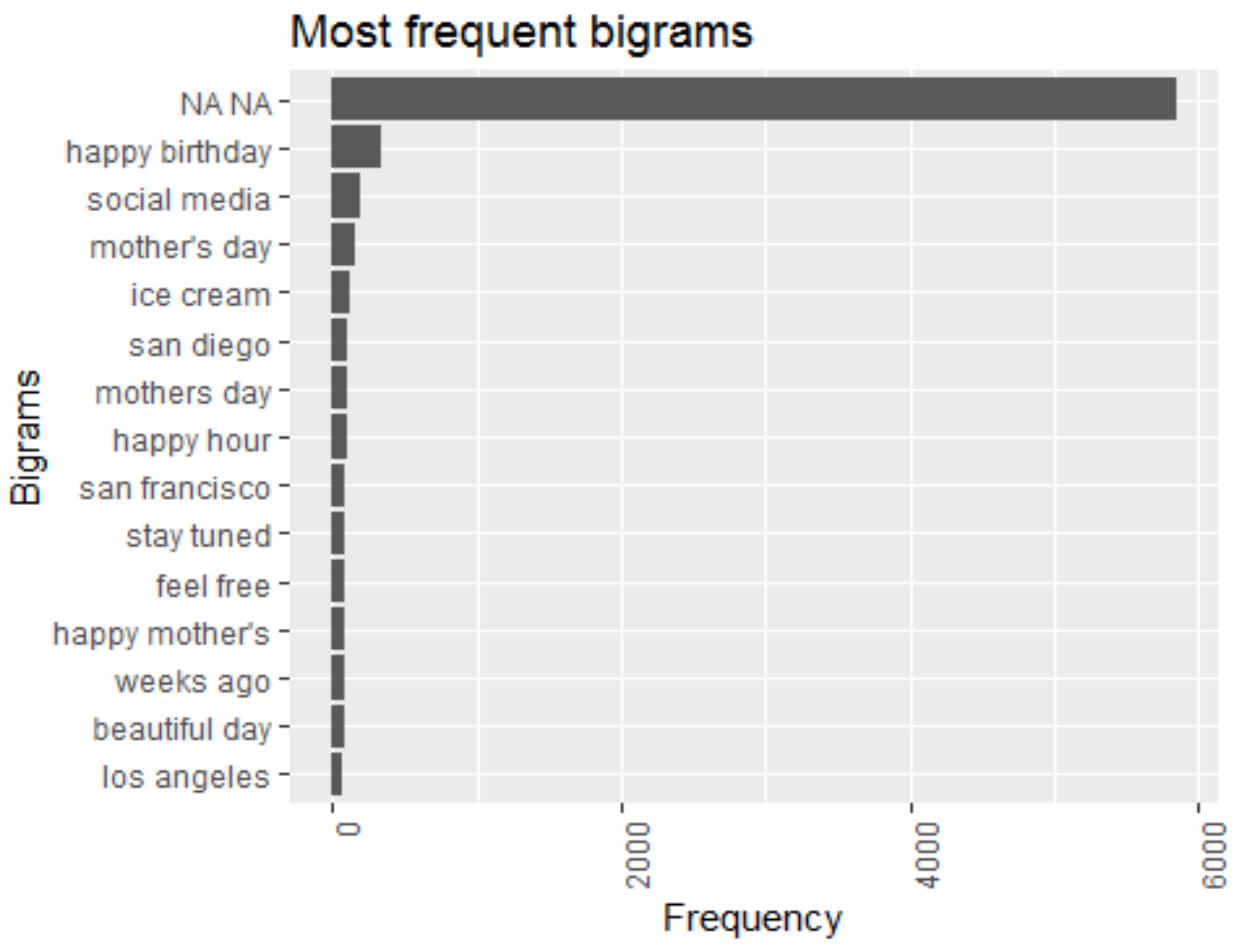
Unigram:

```
UnigramFreq <- data_sample_df %>%
    unnest_tokens(unigram, text, token = "ngrams", n = 3) %>%
    separate(unigram, c("word1"), sep = " ",
             extra = "drop", fill = "right") %>%
    filter(!word1 %in% stop_words$word) %>%
    unite(unigram, word1, sep = " ") %>%
    count(unigram, sort = TRUE)

ggplot(head(UnigramFreq,15), aes(reorder(unigram,n), n)) +
    geom_bar(stat="identity") + coord_flip() +
    xlab("Unigrams") + ylab("Frequency") +
    ggtitle("Most frequent unigrams") +
    theme(axis.text.x=element_text(angle=90, hjust=1))
```



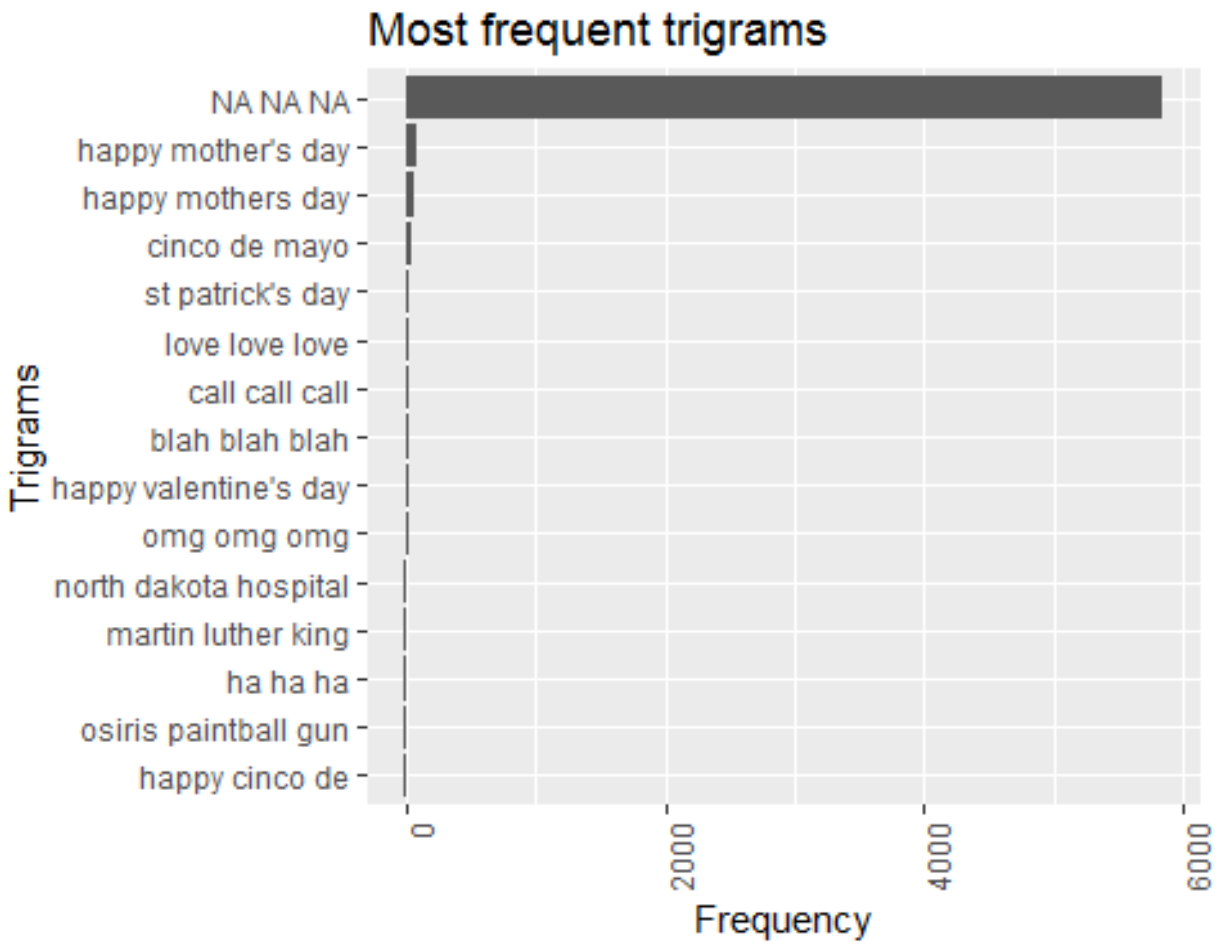
Bigram:

```
BigramFreq <- data_sample_df %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 3) %>%
    separate(bigram, c("word1", "word2"), sep = " ",
             extra = "drop", fill = "right") %>%
    filter(!word1 %in% stop_words$word,
           !word2 %in% stop_words$word) %>%
    unite(bigram, word1, word2, sep = " ") %>%
    count(bigram, sort = TRUE)

ggplot(head(BigramFreq,15), aes(reorder(bigram,n), n)) +
    geom_bar(stat="identity") + coord_flip() +
    xlab("Bigrams") + ylab("Frequency") +
    ggtitle("Most frequent bigrams") +
    theme(axis.text.x=element_text(angle=90, hjust=1))
```



Trigram:

```
TrigramFreq <- data_sample_df %>%
    unnest_tokens(trigram, text, token = "ngrams", n = 3) %>%
    separate(trigram, c("word1", "word2", "word3"), sep = " ",
             extra = "drop", fill = "right") %>%
    filter(!word1 %in% stop_words$word,
           !word2 %in% stop_words$word,
           !word3 %in% stop_words$word) %>%
    unite(trigram, word1, word2, word3, sep = " ") %>%
    count(trigram, sort = TRUE)

ggplot(head(TrigramFreq,15), aes(reorder(trigram,n), n)) +
    geom_bar(stat="identity") + coord_flip() +
    xlab("Trigrams") + ylab("Frequency") +
    ggtitle("Most frequent trigrams") +
    theme(axis.text.x=element_text(angle=90, hjust=1))
```



Summary

1. the data sets are pretty big and processing them requires time and computing resources.

2. most of the top ranking n-grams contains English stop words.
3. using the n-grams we can conceive a crude algorithm to suggest the next words in a text editor; For example, the probability of an untyped word can be estimated from the frequencies in the corpus of the n-grams containing that word in the last position conditioned on the presence the last typed word(s) as the first n - 1 words in the n-gram. One can use a weighted sum of frequencies, with the weights calculated using machine learning.
4. use a pre-built R algorithm, like one based on Hidden Markov model and the n-grams calculated from the data sets provided in this class.