HarvardX: PH125.9x Data Science: Capstone Course

Text Prediction; SwiftKey Dataset

Bijal Ashwin Ved 12/29/2019

Contents

1 Introduction

This Report is for the Coursera Data Science Capstone final project. The goal of the project is to create a predictive text model using a large text quantity of documents as training data.

This report describes the major features of the training data with our exploratory data analysis and summarizes plan for creating the predictive model.

Goal of this project is to develop and train a recommendation machine learning algorithm to predict the next text in the algorithm.

This report will present an overview of the data, analysis, results and a conclusion.

1.1 Dataset

The data was downloaded from https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip and unzipped. The 3 text files containing blogs, news articles and tweets are used for basic summary statistical calculation.

```
#Libraries:
```

```
library(tm)
library(stringi)
library(RWeka)
library(ggplot2)
library(quanteda)
```

```
# Package Installs:
```

```
blogs <- readLines("C:/Users/jivra/OneDrive/Documents/Capstone-MovieLens-Project/en_US.blogs.txt",
                   encoding = "UTF-8", skipNul = TRUE)
news <- readLines("C:/Users/jivra/OneDrive/Documents/Capstone-MovieLens-Project/en_US.news.txt",
                  encoding = "UTF-8", skipNul = TRUE)
twitter <- readLines("C:/Users/jivra/OneDrive/Documents/Capstone-MovieLens-Project/en_US.twitter.txt",
                     encoding = "UTF-8", skipNul = TRUE)
```

2 Examine the data and summarize findings:

```
##################################################
# Get file sizes
blogs.size <- file.info("final/en_US/en_US.blogs.txt")$size / 1024 ^ 2
news.size <- file.info("final/en_US/en_US.news.txt")$size / 1024 ^ 2
twitter.size <- file.info("final/en_US/en_US.twitter.txt")$size / 1024 ^ 2


##################################################
# Get words in files
blogs.words <- stri_count_words(blogs)
news.words <- stri_count_words(news)
twitter.words <- stri_count_words(twitter)

##################################################
# Summary of the data sets
data.frame(source = c("blogs", "news", "twitter"),
           file.size.MB = c(blogs.size, news.size, twitter.size),
           num.lines = c(length(blogs), length(news), length(twitter)),
           num.words = c(sum(blogs.words), sum(news.words), sum(twitter.words)),
           mean.num.words = c(mean(blogs.words), mean(news.words), mean(twitter.words)))
```

```
##    source file.size.MB num.lines num.words mean.num.words
## 1   blogs           NA    899288  37546246       41.75108
## 2    news           NA   1010242  34762395       34.40997
## 3 twitter           NA   2360148  30093410       12.75065
```

2.2 Data Cleanup:

```
#Before performing the analysis, URLs, special characters, punctuations, numbers etc needs
#to be #removed from the data sets.
#Due to the size of the data, 1% of the dataset was used randomly for the analysis:
```

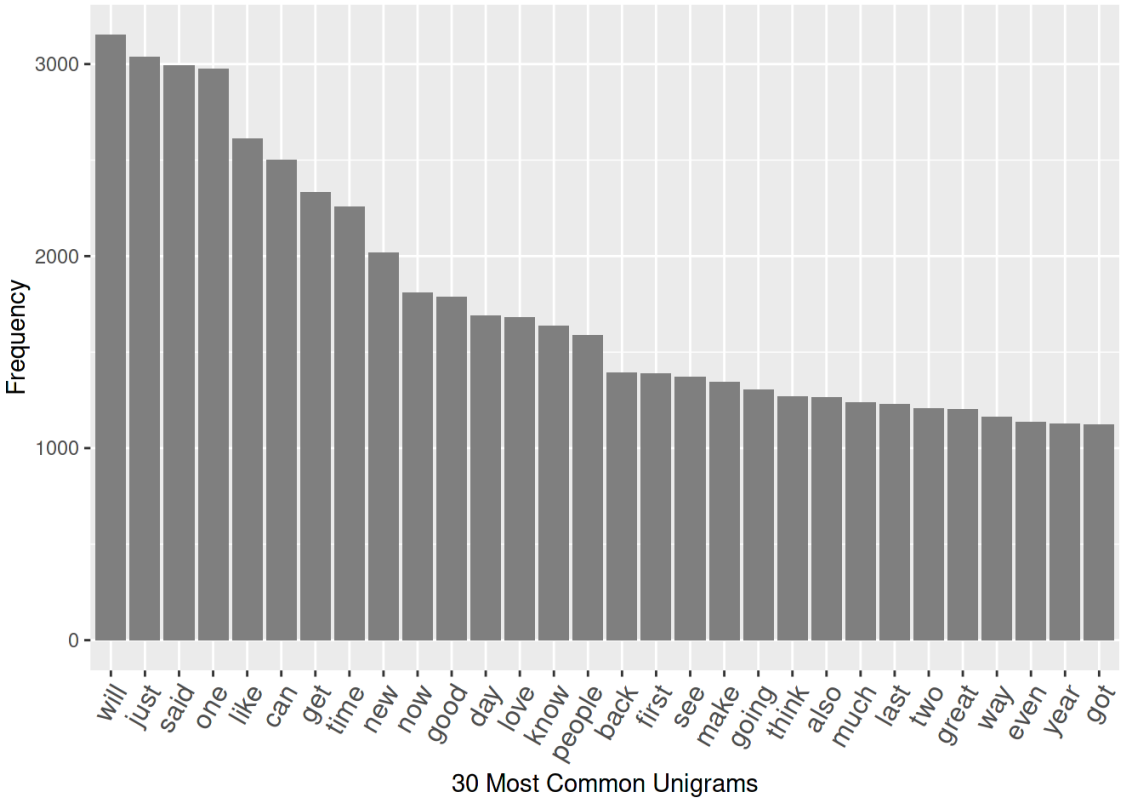
```
set.seed(42)
data.sample <- c(sample(blogs, length(blogs) * 0.01),
                 sample(news, length(news) * 0.01),
                 sample(twitter, length(twitter) * 0.01))

# Create corpus and clean the data
corpus <- VCorpus(VectorSource(data.sample))
toSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))
corpus <- tm_map(corpus, toSpace, "(f|ht)tp(s?)://(.*)[.][a-z]+")
corpus <- tm_map(corpus, toSpace, "@[^\\s]+")
corpus <- tm_map(corpus, tolower)
corpus <- tm_map(corpus, removeWords, stopwords("en"))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, PlainTextDocument)
```

2.3 Exploratory Analysis:

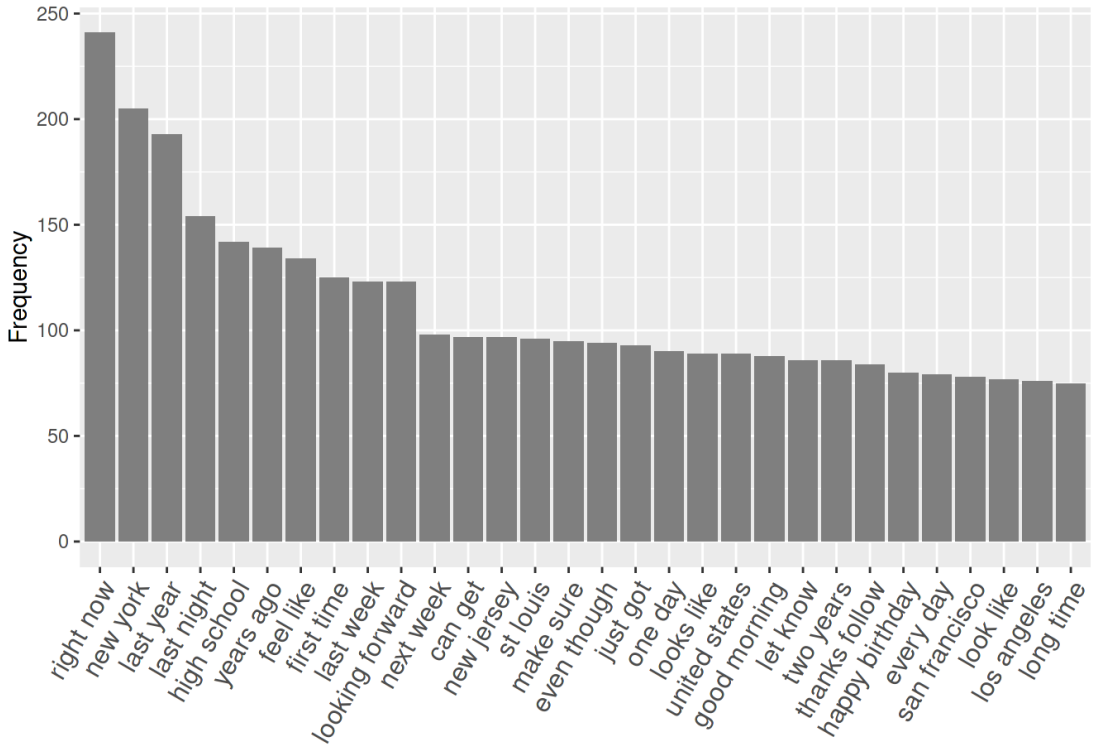
```
## Exploratory Analysis
options(mc.cores=1)
getFreq <- function(tdm) {
  freq <- sort(rowSums(as.matrix(tdm)), decreasing = TRUE)
  return(data.frame(word = names(freq), freq = freq))
}
bigram <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
trigram <- function(x) NGramTokenizer(x, Weka_control(min = 3, max = 3))
makePlot <- function(data, label) {
  ggplot(data[1:30,], aes(reorder(word, -freq), freq)) +
    labs(x = label, y = "Frequency") +
    theme(axis.text.x = element_text(angle = 60, size = 12, hjust = 1)) +
    geom_bar(stat = "identity", fill = I("grey50"))
}

# Get frequencies of most common n-grams in data sample
freq1 <- getFreq(removeSparseTerms(TermDocumentMatrix(corpus), 0.9999))
freq2 <- getFreq(removeSparseTerms(TermDocumentMatrix(corpus, control = list(tokenize = bigram)), 0.9999))
freq3 <- getFreq(removeSparseTerms(TermDocumentMatrix(corpus, control = list(tokenize = trigram)), 0.9999))
```

3 Graphs:

```
makePlot(freq1, "30 Most Common Unigrams")
```

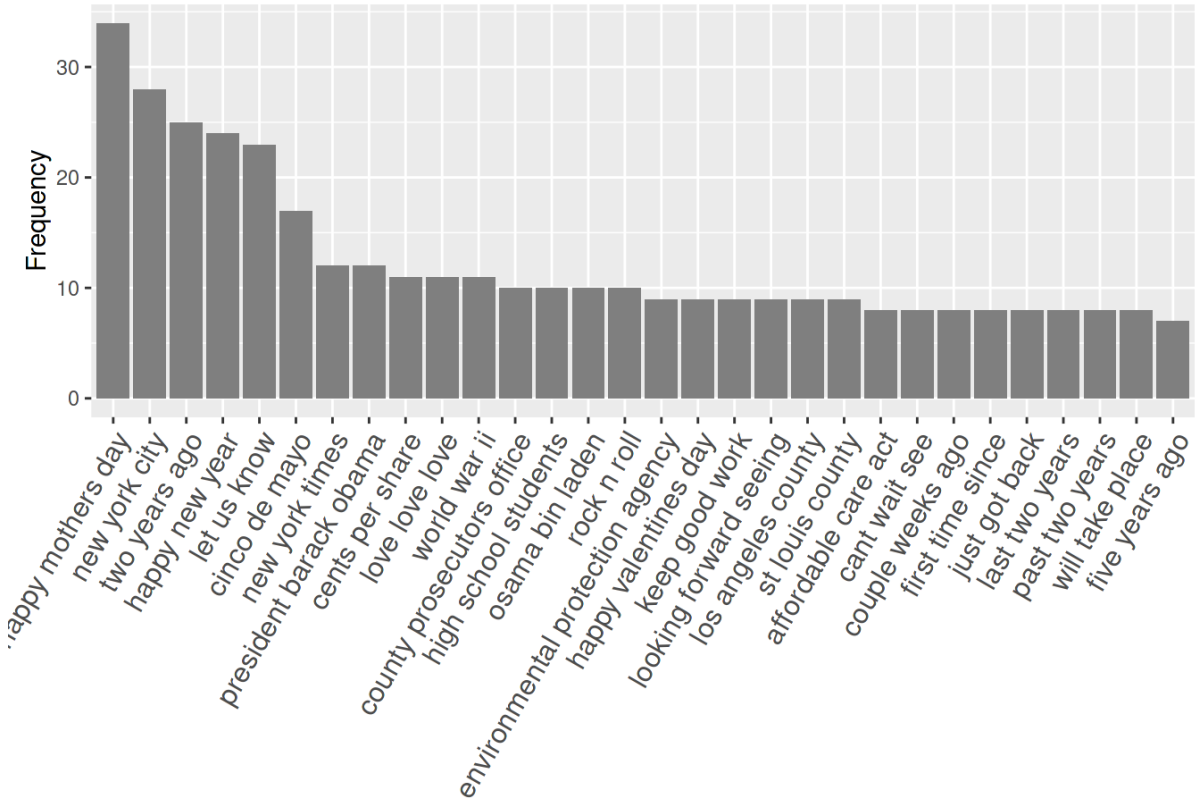



```
makePlot(freq2, "30 Most Common Bigrams")
```



30 Most Common Bigrams

```
makePlot(freq3, "30 Most Common Trigrams")
```



30 Most Common Trigrams

4 Conclusion & Next Steps: We can use the algorithm from this model to predict the next text word.