Coursera Data Science Capstone - Milestone Report

Gudur Guy

7/6/2023

Introduction

The goal of the Data Science Capstone project is to develop an application with textual prediction algorithm using Natural Language Processing techniques. With that in mind, the objective of this Data Science Capstone Project's Milestone report is the following:

- Load and create basic summaries of the provided data set provided by SwiftKey Company.
- Cleanse and Prepare the data
- Do some Exploratory Data Analysis
- Then in-brief describe the further steps for the completion of the Final Project

The dataset provided has three different sources of data (**blogs, news and twitter**) for four different languages. We will be focusing on just English data set for the purposes of this project.

Initial Setup and Data Loading & Summarization

Load Libraries

```
library(knitr)
library(ggplot2)
library(stringi)
library(kableExtra)
library(tm)
library(RWeka)
library(wordcloud)
library(RColorBrewer)
```

Load Data

```
set.seed(5432)

dataDir <- "./final/en_US/"

blogsFileName   <- "en_US.blogs.txt"
newsFileName    <- "en_US.news.txt"
twitterFileName <- "en_US.twitter.txt"
```

Table 1: Files Data Summary

fileNames	fileSize	linesCount	wordsCount	charsCount
en_US.blogs.txt	200 MB	899288	37570839	206824505
en_US.news.txt	196 MB	1010242	34494539	203223159
en_US.twitter.txt	159 MB	2360148	30451170	162096241

```
blogsFile   <- paste(dataDir, "en_US.blogs.txt", sep = "")
newsFile    <- paste(dataDir,"en_US.news.txt", sep = "")
twitterFile <- paste(dataDir,"en_US.twitter.txt", sep = "")

blogs   <- readLines(blogsFile, skipNul = TRUE)
news    <- readLines(newsFile, skipNul = TRUE)
twitter <- readLines(twitterFile,  skipNul = TRUE)
```

Create Data Summarization

```
fileNames  <- c(blogsFileName, newsFileName, twitterFileName)

fileSizesMB <- round(c(file.info(blogsFile, newsFile, twitterFile))$size /1024 ^ 2)

linesCount = sapply(list(blogs, news, twitter), length)
wordsCount = sapply(list(blogs, news, twitter), stri_stats_latex)[4,]
charsCount = sapply(list(nchar(blogs), nchar(news), nchar(twitter)), sum)

dataSummary <- data.frame( fileNames, fileSize = paste(fileSizesMB, " MB"), linesCount, wordsCount, cha
kable(dataSummary, row.names = FALSE,  align = c("l", rep("r", 7)), caption = "Files Data Summary") %>%
```

Cleanse and Prepare the data

Prepare Sample Data

Given the massive amount of data, we will train the model by getting a sample of data to reduce the computational burden of the model.

We will use a binomial function to decide which lines should be included or not.

```
samplePercent <- .01

# extract sample data using binomial function
blogsSampl <- blogs[as.logical(rbinom(length(blogs),1,samplePercent))]
newsSampl <- news[as.logical(rbinom(length(news),1,samplePercent))]
twitterSampl <- twitter[as.logical(rbinom(length(twitter),1,samplePercent))]

# remove all non english characters
blogsSampl <- iconv(blogsSampl, "latin1", "ASCII", sub="")
newsSampl <- iconv(newsSampl, "latin1", "ASCII", sub="")
twitterSampl <- iconv(twitterSampl, "latin1", "ASCII", sub="")
```

```r
# combine all three data sets into a single data set and write to disk for the future
sampleData <- c(blogsSampl, newsSampl, twitterSampl)
sampleDataFileName <- paste(dataDir, "en_US.sample.txt", sep = "")
con <- file(sampleDataFileName, open = "w")
writeLines(sampleData, con)
close(con)
```

Clean Data and Create a Corpus From Sample Data

We can clean the data (e.g. removing white spaces, numbers, UTR, punctuation etc.) along with removing the profanities at this point while building the Corpus

```r
# profanities file
profFile <- "full-list-of-bad-words_text-file_2022_05_05.txt"
profFileName <- paste(dataDir, profFile, sep = "")

# start creating a clean corpus
sampleCorpus <- VCorpus(VectorSource(sampleData))

# remove all the heavy objects as we can work with sample data going forward
rm(blogs, news, twitter, blogsSampl, newsSampl, twitterSampl, sampleData)

# to lower case
sampleCorpus <- tm_map(sampleCorpus, content_transformer(tolower))

# remove profane words
con <- file(profFileName, open = "r")
profanities <- readLines(con, encoding = "UTF-8", skipNul = TRUE)
sampleCorpus <- tm_map(sampleCorpus, removeWords, profanities)
close(con)

# remove all english stop words
sampleCorpus <- tm_map(sampleCorpus,removeWords, stopwords("english"))
# remove numbers
sampleCorpus <- tm_map(sampleCorpus, removeNumbers)
# remove punctuation
sampleCorpus <- tm_map(sampleCorpus, removePunctuation)
# removing stray letters
sampleCorpus <- tm_map(sampleCorpus, removeWords, letters)
# strip extra white space
sampleCorpus <- tm_map(sampleCorpus, stripWhitespace)
```

Exploratory Data Analysis

Tokenizing and N-Gram Generation

We will do exploratory data analysis to see the following:

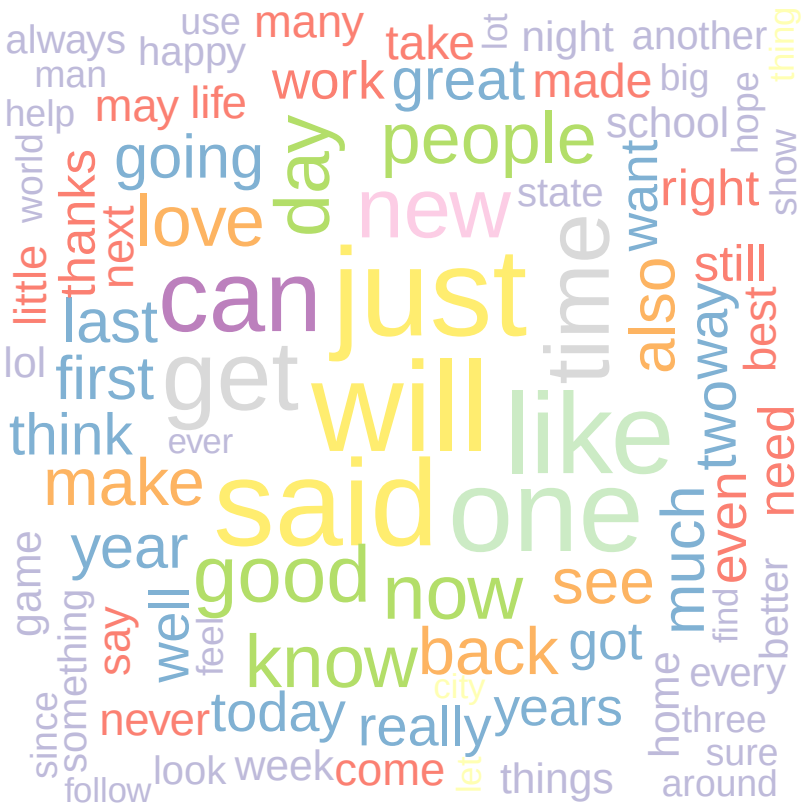
What are the most frequently occurring words, what are the most frequently occurring words together by tokenizing and n-gram generation by generating 1-gram, 2-gram and 3-grams from the sample corpus we built previously.

```
# unigram tokenizer
unigramTokenizer <- function(x) {NGramTokenizer(x, Weka_control(min = 1, max = 1))}

# bigram tokenizer
bigramTokenizer <- function(x) {NGramTokenizer(x, Weka_control(min = 2, max = 2))}

# trigram tokenizer
trigramTokenizer <- function(x) {NGramTokenizer(x, Weka_control(min = 3, max = 3))}

# generating unigram term document matrix
unigrams <- TermDocumentMatrix(sampleCorpus, control = list(tokenize = unigramTokenizer))

# generating bigram term document matrix
bigrams <- TermDocumentMatrix(sampleCorpus, control = list(tokenize = bigramTokenizer))

# generating trigram term document matrix
trigrams <- TermDocumentMatrix(sampleCorpus, control = list(tokenize = trigramTokenizer))
```

Exploratory Plots

```
unigramMatrixFreq <- sort(rowSums(as.matrix(removeSparseTerms(unigrams, 0.99))), decreasing = TRUE)
unigramMatrixFreq <- data.frame(word = names(unigramMatrixFreq), freq = unigramMatrixFreq)

wordcloud(words = unigramMatrixFreq$word, freq = unigramMatrixFreq$freq, min.freq = 1, max.words=100, ra
```

Unigram WordCloud

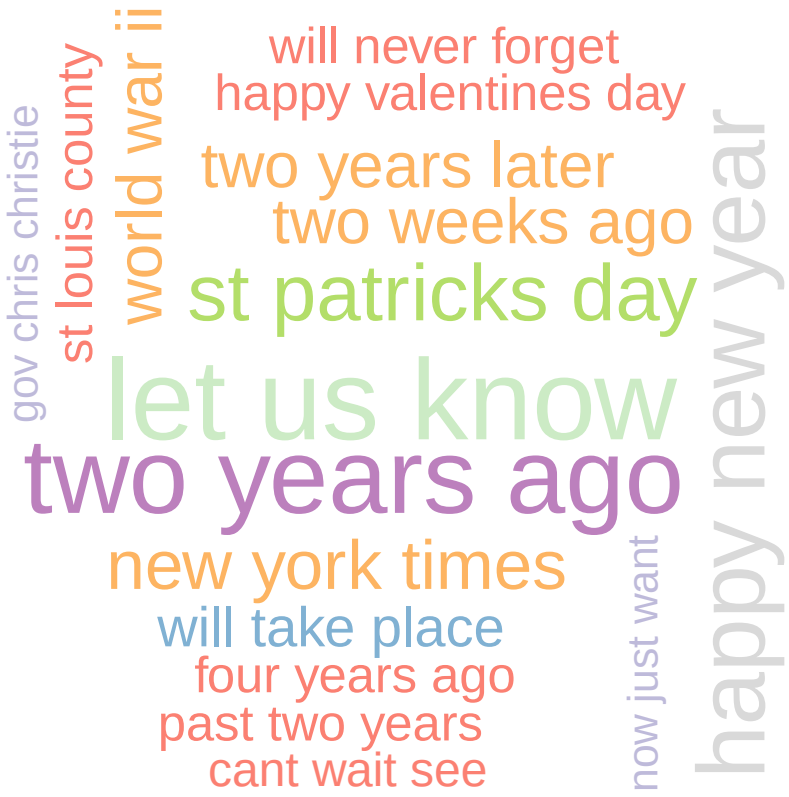



```
bigramMatrixFreq <- sort(rowSums(as.matrix(removeSparseTerms(bigrams, 0.999))), decreasing = TRUE)
bigramMatrixFreq <- data.frame(biword = names(bigramMatrixFreq), bifreq = bigramMatrixFreq)

wordcloud(words = bigramMatrixFreq$biword, freq = bigramMatrixFreq$bifreq, min.freq = 1, max.words=50, r
```

Bigram WordCloud

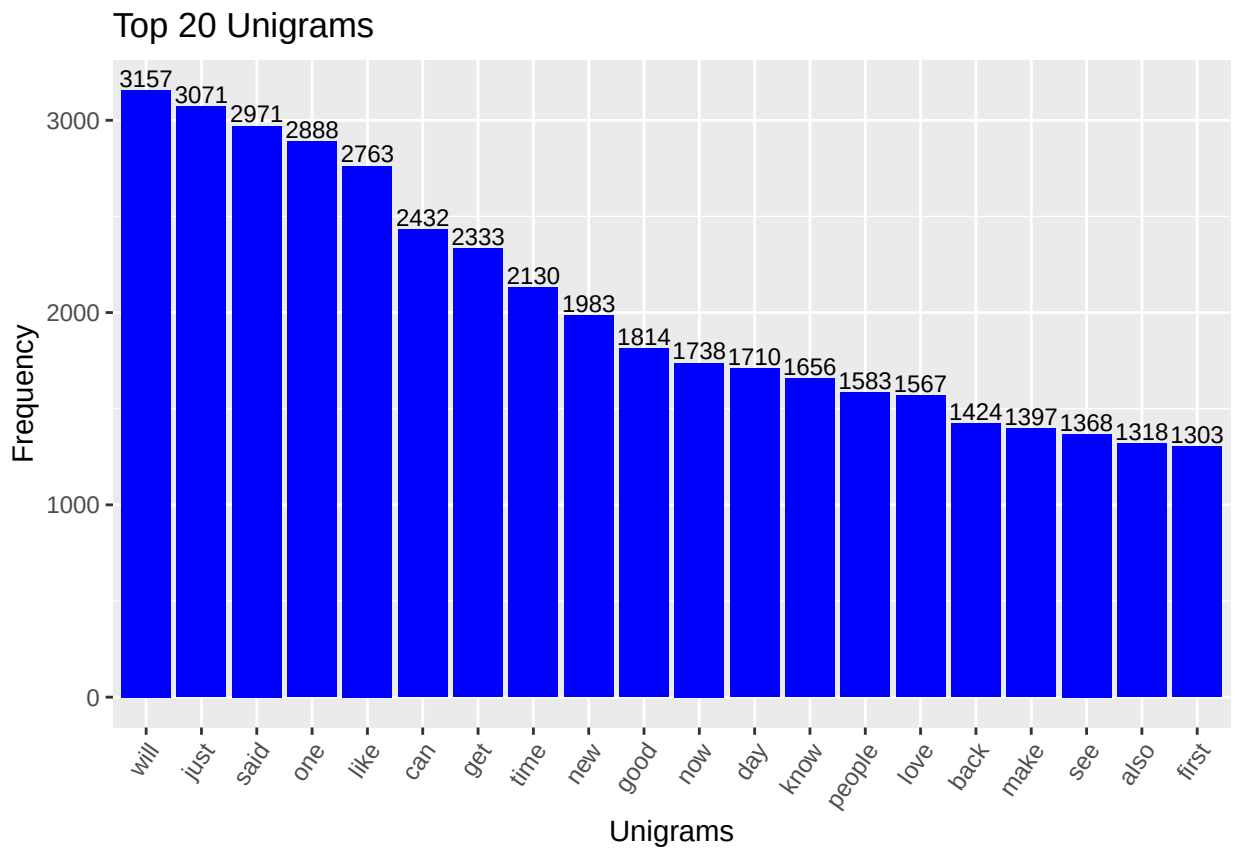

```
trigramMatrixFreq <- sort(rowSums(as.matrix(removeSparseTerms(trigrams, 0.9999))), decreasing = TRUE)
trigramMatrixFreq <- data.frame(triword = names(trigramMatrixFreq), trifreq = trigramMatrixFreq)

wordcloud(words = trigramMatrixFreq$triword, freq = trigramMatrixFreq$trifreq, min.freq = 1, max.words=5
```

Trigram WordCloud



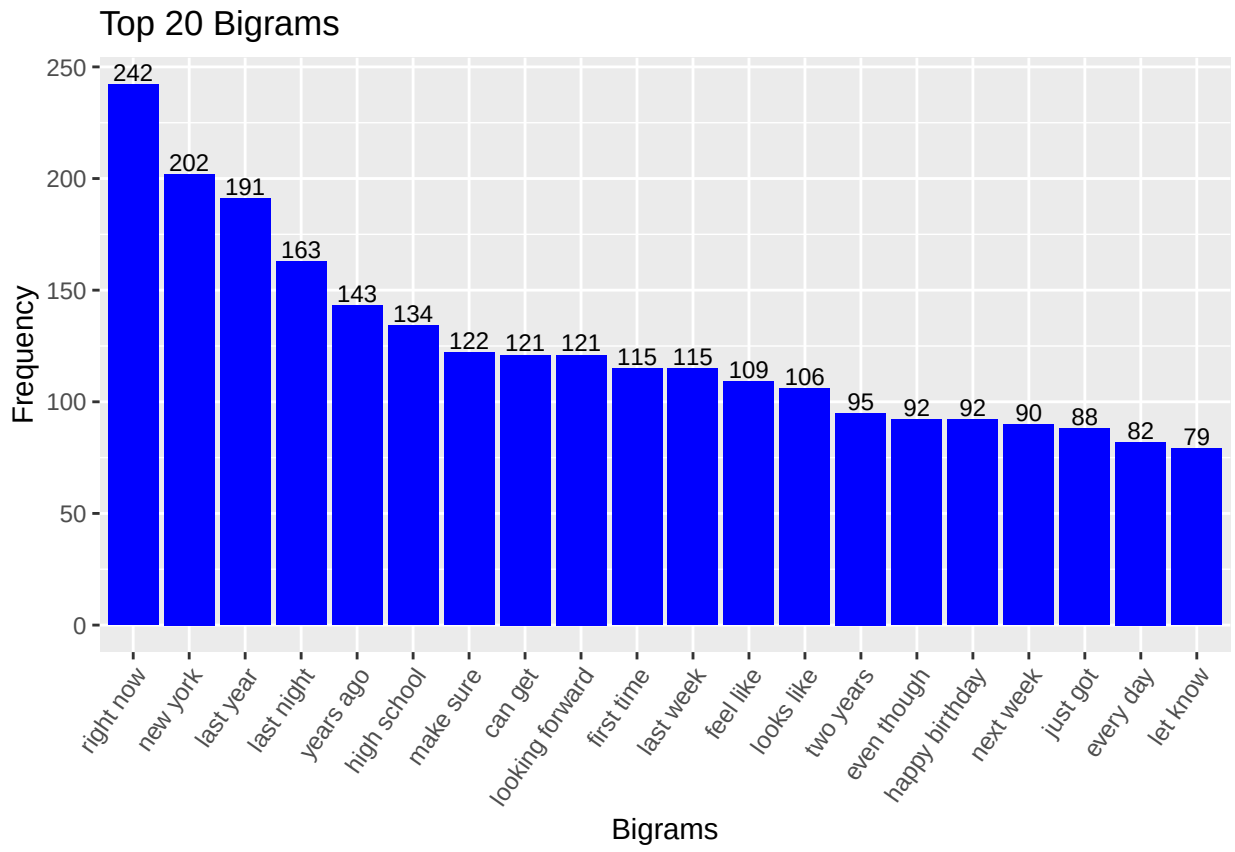

```
g <- ggplot(unigramMatrixFreq[1:20,],aes(x=reorder(word,-freq),y=freq))+geom_bar(stat="identity",fill="
g
```

Top 20 Unigrams and their Frequencies



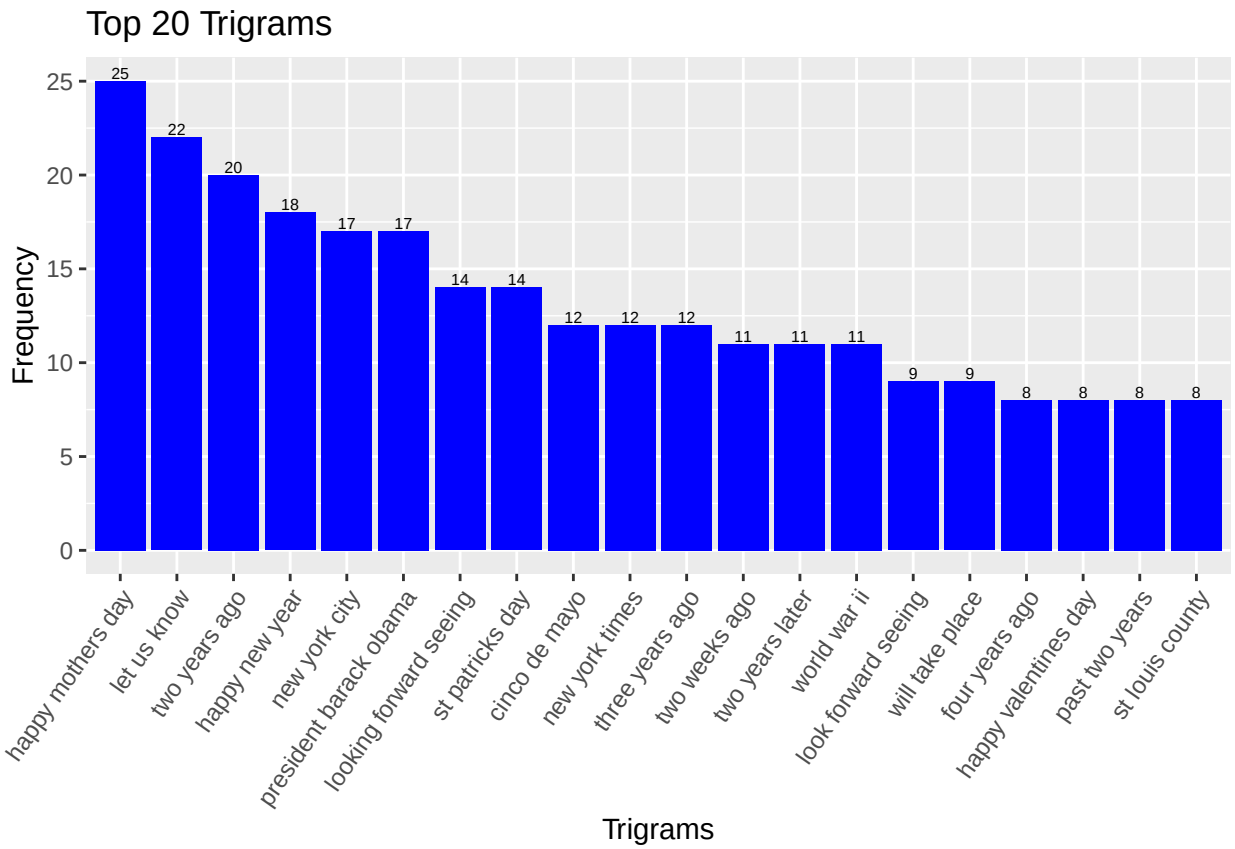

```
g <- ggplot(bigramMatrixFreq[1:20,],aes(x=reorder(biword,-bifreq),y=bifreq))+geom_bar(stat="identity",f
g
```

Top 20 Bigrams and their Frequencies




```
g <- ggplot(trigramMatrixFreq[1:20,],aes(x=reorder(triword,-trifreq),y=trifreq))+geom_bar(stat="identity
g
```

Top 20 Triigrams and their Frequencies



Next Steps

Now that we did the exploratory analysis of the data, we can go on the next steps to conclude the project. So, the final deliverable is to build a text prediction algorithm and it will be deployed as a Shiny Application. This will take in **a term or terms as input and predict the next word** based on our prediction algorithm.

The algorithm will use n-gram model with a frequency look up for predict the next word. This algorithm can be a **Katz back-off model** i.e. using trigrams, bigrams and unigrams in that order. Or we could use **Interpolation**, which combines the probability estimates from all the n-grams and then selects the best choice.

We can decide which algorithm based on the accuracy and simplicity to run the model.