DSCapstone-SwiftKey data Analysis - Milestone Report

Code

Ariful I. Mondal

13 November 2017

1 Synopsis - Data Science Capstone Project

This milestone report is part of the data science capstone project from “Data Science Specialization” (https://www.coursera.org/specializations/jhu-data-science) Learning path on Coursera (https://www.coursera.org) by Johns Hopkins University (https://www.jhu.edu/). The capstone project class allows students to create a usable/public data product that can be used to show your skills to potential employers. Projects is drawn from real-world problems and will be conducted with industry, government, and academic partners. This project is supported by SwiftKey (https://swiftkey.com/en), corporate partner in this capstone, builds a smart keyboard that makes it easier for people to type on their mobile devices.

2 Introduction

Around the world, people are spending an increasing amount of time on their mobile devices for email, social networking, banking and a whole range of other activities. But typing on mobile devices can be a serious pain. SwiftKey, our corporate partner in this capstone, builds a smart keyboard that makes it easier for people to type on their mobile devices. One cornerstone of their smart keyboard is predictive text models. When someone types:

“I went to the”"

the keyboard presents three options for what the next word might be. For example, the three words might be gym, store, restaurant. In this capstone you will work on understanding and building predictive text models like those used by SwiftKey.

In this project we will start with the basics, analyzing a large corpus of text documents to discover the structure in the data and how words are put together. It will cover cleaning and analyzing text data, then building and sampling from a predictive text model. Finally, you will use the knowledge you gained in data products to build a predictive text product you can show off to your family, friends, and potential employers.

3 About the Data

This is the training data to get you started that will be the basis for most of the capstone. In this project I will use the data from the Coursera site as instructed in the coursera course from below link.

Capstone Dataset (https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip)

We have text documents in 4 different languages such as English, US (`en_US` , ~550MB), German (`de_DE` , ~240MB), Russian (ru_RU, ~325MB) and Finish (`fi_FI` , ~220MB) in the forms of `blogs` , `news` and twitter`. I am planning to use English, US data for this analysis where we have

- en_US.blogs.txt (~200MB)
- en_US.news.txt (~200MB)
- en_US.twitter.txt (~160MB)

I have downloaded the data manually but it can be downloaded diagrammatically using below scripts

Hide

```
setwd("D:\\dscapstone")  # Project Workspace

wbsource<-"https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip" #
 Data Scource

projFileName<-"Coursera-SwiftKey.zip" # Destination.

download.file(wbsource, projWorkspace) # Download and save

unzip(projFileName)  # Uncompress file
```

4 Scope of the project

In this capstone project we would be performing following tasks in order to achieve the overall objective of the project:

- Understanding the problem
- Data acquisition and cleaning
- Exploratory analysis
- Statistical modeling
- Predictive modeling
- Creative exploration
- Creating a data product
- Creating a short slide deck pitching your product

5 Preditive Model Development Goals

The goal of this exercise is to build and evaluate your first predictive model. You will use the n-gram and backoff models you built in previous tasks to build and evaluate your predictive model. The goal is to make the model efficient and accurate.

5.1 Tasks to accomplish

- Build a predictive model based on the previous data modeling steps - you may combine the models in any way you think is appropriate
- Evaluate the model for efficiency and accuracy - use timing software to evaluate the computational complexity of your model
- Evaluate the model accuracy using different metrics like perplexity, accuracy at the first word, second word, and third word
- Explore new models and data to improve your predictive model
- Evaluate your new predictions on both accuracy and efficiency

5.1.1 Questions to consider

- How does the model perform for different choices of the parameters and size of the model?
- How much does the model slow down for the performance you gain?
- Does perplexity correlate with the other measures of accuracy?
- Can you reduce the size of the model (number of parameters) without reducing performance?

6 Required **R** packages

I am planning to use following R packages. We would need to install these packages if they are not installed already using `install.packages()` and load them using `library()`.

- **tm:** (https://cran.r-project.org/web/packages/tm/tm.pdf) for text mining and natural language processing(NLP)
- **wordcloud:** (https://cran.r-project.org/web/packages/wordcloud/wordcloud.pdf) to create word clouds
- **stringi:** (https://cran.r-project.org/web/packages/stringi/stringi.pdf) to leverage character string processing facilities
- **clue:** (https://cran.r-project.org/web/packages/clue/clue.pdf) for Cluster Ensembles
- **SnowballC:** (https://cran.r-project.org/web/packages/SnowballC/SnowballC.pdf) for stemming text
- **RColorBrewer:** (https://cran.r-project.org/web/packages/RColorBrewer/RColorBrewer.pdf) color palettes for nice colours of the graphs
- **RWeka:** (https://cran.r-project.org/web/packages/RWeka/RWeka.pdf) text mining and N-Gram analysis
- **ggplot2** (https://cran.r-project.org/package=ggplot2/ggplot2.pdf) to Create Elegant Data Visualizations Using the Grammar of Graphics

Hide

```
## Load Packages
library("tm")
library("stringi")
library("wordcloud")
library("clue")
library("ggplot2")
library("RColorBrewer")
library("SnowballC")
library("RWeka")
```

7 Check for data availability

Before going ahead, it is important to to download the data keep in the right path. So I wish to check whether they files exist, if not then download from the web-source.

Hide

```
setwd("D:\\dscapstone")

# Check if the file has been extracted
if (!file.exists("./final")) {

  if(!file.exists("Swiftkey.zip")){
    url = "https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip"
    download.file(url, "Swiftkey.zip", method = "curl")
    dateDownload <- date()
    }
  SwiftKey.zip<- "Coursera-SwiftKey.zip"
  outDir<-"."
  unzip(SwiftKey.zip,exdir=outDir)
}
```

8 Analysis of English, US data

For this project I am using documents with English-US (`en_US.blogs.txt` , `en_US.news.txt` and `en_US.twitter.txt`) , though I have documents with German, Russian and Finish.

8.1 Reading data

I have used `readLines()` function to read the text files, but for the large size of data we can also use a connection string.

Hide

```
# Read files in the R using
us_blogs<-readLines("final\\en_US\\en_US.blogs.txt",  encoding = "UTF-8", skipNul = TRUE)
us_news<-readLines("final\\en_US\\en_US.news.txt",  encoding = "UTF-8", skipNul = TRUE)
us_twitter<-readLines("final\\en_US\\en_US.twitter.txt",  encoding = "UTF-8", skipNul = TRUE)
```

8.2 Summary and Structure of the data

8.2.1 Blogs summary

Hide

```
# Summary US blogs
summary(us_blogs)
```

```
##    Length     Class      Mode
##    899288 character character
```

Hide

```
# Structure of the data
str(us_blogs)
```

```
##  chr [1:899288] "In the years thereafter, most of the Oil fields and platforms were named
after pagan <U+0093>gods<U+0094>." ...
```

8.2.1.1 View few lines from the file `us_blog` :

Hide

```
us_blogs[3:5]
```

```
## [1] "Chad has been awesome with the kids and holding down the fort while I work later than
usual! The kids have been busy together playing Skylander on the XBox together, after Kyan ca
shed in his $$$ from his piggy bank. He wanted that game so bad and used his gift card from h
is birthday he has been saving and the money to get it (he never taps into that thing either,
that is how we know he wanted it so bad). We made him count all of his money to make sure tha
t he had enough! It was very cute to watch his reaction when he realized he did! He also does
a very good job of letting Lola feel like she is playing too, by letting her switch out the c
haracters! She loves it almost as much as him."
## [2] "so anyways, i am going to share some home decor inspiration that i have been storing
in my folder on the puter. i have all these amazing images stored away ready to come to life
when we get our home."



## [3] "With graduation season right around the corner, Nancy has whipped up a fun set to hel
p you out with not only your graduation cards and gifts, but any occasion that brings on a ch
ange in one's life. I stamped the images in Memento Tuxedo Black and cut them out with circle
Nestabilities. I embossed the kraft and red cardstock with TE's new Stars Impressions Plate,
which is double sided and gives you 2 fantastic patterns. You can see how to use the Impressi
ons Plates in this tutorial Taylor created. Just one pass through your die cut machine using
the Embossing Pad Kit is all you need to do - super easy!"
```

8.2.2 News summary

Hide

```
# Summary US News
summary(us_news)
```

```
##    Length     Class      Mode
##     77259 character character
```

Hide

```
str(us_news)
```

```
##  chr [1:77259] "He wasn't home alone, apparently." ...
```

8.2.2.1 View few lines from the file `us_news` :

Hide

```
us_news[3:5]
```

```
## [1] "WSU's plans quickly became a hot topic on local online sites. Though most people appl
auded plans for the new biomedical center, many deplored the potential loss of the building."



## [2] "The Alaimo Group of Mount Holly was up for a contract last fall to evaluate and sugge
st improvements to Trenton Water Works. But campaign finance records released this week show
the two employees donated a total of $4,500 to the political action committee (PAC) Partners
for Progress in early June. Partners for Progress reported it gave more than $10,000 in both
direct and in-kind contributions to Mayor Tony Mack in the two weeks leading up to his victor
y in the mayoral runoff election June 15."
## [3] "And when it's often difficult to predict a law's impact, legislators should think twi
ce before carrying any bill. Is it absolutely necessary? Is it an issue serious enough to mer
it their attention? Will it definitely not make the situation worse?"
```

8.2.3 Twitter Summary

Hide

```
# Summary US Twitter
summary(us_twitter)
```

```
##    Length     Class      Mode
##   2360148 character character
```

Hide

```
str(us_twitter)
```

```
##  chr [1:2360148] "How are you? Btw thanks for the RT. You gonna be in DC anytime soon? Lov
e to see you. Been way, way too long." ...
```

8.2.3.1 View few lines from the file `us_twitter` :

Hide

```
us_twitter[3:5]
```

```
## [1] "they've decided its more fun if I don't."
## [2] "So Tired D; Played Lazer Tag & Ran A LOT D; Ughh Going To Sleep Like In 5 Minutes ;)"
## [3] "Words from a complete stranger! Made my birthday even better :)"
```

8.2.4 Longest lines in the files

Hide

```
longest_lines<- (c(max(nchar(us_blogs)), max(nchar(us_news)), max(nchar(us_twitter))))
type <- c("US-Blogs", "US-News", "US-Twitter")
longest_lines <- as.data.frame(cbind(type,longest_lines))
colnames(longest_lines) <- c("file.name", "line.length")
longest_lines$line.length <- as.numeric(longest_lines$line.length)
print(longest_lines)
```

```
##    file.name line.length
## 1   US-Blogs           2
## 2    US-News           3
## 3 US-Twitter           1
```

8.2.5 Shortest lines in the files

Hide

```
shortest_lines<-c(min(nchar(us_blogs)), min(nchar(us_news)), min(nchar(us_twitter)))
type <- c("US-Blogs", "US-News", "US-Twitter")
shortest_lines<-as.data.frame(cbind(type, shortest_lines))
colnames(shortest_lines) <- c("file.name", "line.length")
print(shortest_lines)
```

```
##    file.name line.length
## 1   US-Blogs           1
## 2    US-News           2
## 3 US-Twitter           2
```

8.3 Sampling Data and Data Preparation

As we have large size of datasets to work with, we will use a representative sample for this project to improve performance of Shiny app and predictive models. I have randomly selected representative samples of the data for analysis and modeling using `sample()` function. This will help to reduce usage of memory and increase performance of the Shiny application.

Hide

```
# Set random seed so that samples do not change
set.seed(12345)

# Sample from US-Blogs
# Help ?sample
us_blogs_sample <- sample(us_blogs, length(us_blogs)*0.01)

# Sample from US-News
us_news_sample <- sample(us_news, length(us_news)*0.10)

# Sample from US Twitter
us_twitter_sample <- sample(us_twitter, length(us_twitter)*0.01)

# Combine all the sample
us_data_sample <- c(us_blogs_sample,us_news_sample,us_twitter_sample)

# remove data to make some space
rm(list=c("type", "us_blogs", "us_news", "us_twitter", "us_blogs_sample", "us_news_sample",
"us_twitter_sample"))
```

Hide

```
# Quick check on data structure
us_data_sample[1:3]
```

```
## [1] "And now I<U+0092>m home. Older, wiser, a little slimmer and hopefully secure in the k
nowledge of what I<U+0092>m good at, and what I want to do with my time."



## [2] "I turned the Today show on to catch up on the morning's news and immediately I knew s
omething terrible had happened in New York City."



## [3] "I<U+0092>ll take this opportunity to diverge from the usual Take Three path, and, ins
tead of focusing on one last role, offer up an Arkin Remix - a concisely-potted overview. Ark
in has long been seen as one of the exemplary supporting actors. So many of his roles before
his resurgence in popularity during the <U+0091>90s and <U+0091>00s, to present day, were mem
orable; it<U+0092>s hard to single one last role out. He added charm and a studious commitmen
t to characterising a range of films from his debut (That<U+0092>s Me, in 1963) onwards."
```

Hide

```
summary(us_data_sample)
```

```
##    Length     Class      Mode
##     40318 character character
```

8.3.1 Preprocessing of Text Data

Before performing analysis on Text data, we will preprocess the data parse, clean and transform as necessary for better results and predictions.

8.3.1.1 Remove special characters

- Remove non-English characters, letters etc. using `iconv()` and option `latin1`
- Remove special characters with spaces using `gsub()` and regular expression `[^0-9a-z]`
- Remove duplicate characters using `gsub()` and regular expression
- Remove special numbers with spaces using `gsub()` and regular expression
- Remove multiple spaces to one using `gsub()` and regular expression

To know more on `incon()` click here (https://www.rdocumentation.org/packages/base/versions/3.4.1/topics/iconv), for `gsub()` click here (https://www.r-bloggers.com/regular-expression-and-associated-functions-in-r/) and click here (https://en.wikipedia.org/wiki/Regular_expression) to know more on `regular expression` , also view `regex` (https://stat.ethz.ch/R-manual/R-devel/library/base/html/regex.html).

Hide

```
# Remove non-English characters, letters etc.
# Help ?inconv
us_data_sample<-iconv(us_data_sample, "latin1", "ASCII", sub="")

# Remove special characters with spaces
# Help ?gsub
us_data_sample_1 <- gsub("[^0-9a-z]", " ", us_data_sample, ignore.case = TRUE)
rm(us_data_sample)

# Remove duplicate characters
us_data_sample_1 <- gsub('([[:alpha:]])\\1+', '\\1\\1', us_data_sample_1)

# Remove special numbers with spaces
us_data_sample_1 <- gsub("[^a-z]", " ", us_data_sample_1, ignore.case = TRUE)

# Remove multiple spaces to one
us_data_sample_1 <- gsub("\\s+", " ", us_data_sample_1)
us_data_sample_1 <- gsub("^\\s", "", us_data_sample_1)
us_data_sample_1 <- gsub("\\s$", "", us_data_sample_1)
```

Hide

```
# Summary
summary(us_data_sample_1)
```

```
##    Length     Class      Mode
##     40318 character character
```

Hide

```
str(us_data_sample_1)
```

```
##  chr [1:40318] "And now Im home Older wiser a little slimmer and hopefully secure in the k
nowledge of what Im good at and what "| __truncated__ ...
```

8.3.1.2 Create corpus

Create a virtual corpus using `Vcorpus()` function.

Hide

```
# create Corpus
# Help ??VCorpus
myCorpus <- VCorpus(VectorSource(us_data_sample_1))
rm(us_data_sample_1)
```

8.3.1.3 Apply Transformations and further processing of text

Perform necessary transformation/preprocessing activities using `tm_map()` from `tm` package. The objective is to have clean texts by removing stop words, punctuation, multiple white spaces etc. We will perform the following transformations

- **tolower:** (https://www.rdocumentation.org/packages/quanteda/versions/0.99.12/topics/toLower) normalize text to small cases. For example `My name Is Ariful` will be converted to small case `my name is ariful`.
- **stopwords:** (https://www.rdocumentation.org/packages/tm/versions/0.7-1/topics/stopwords) remove stop words using English dictionary. For example remove words like “a”, “and”, “but”, “how”, “or”, and “what”
- **removePunctuation:** (https://www.rdocumentation.org/packages/tm/versions/0.7-1/topics/removePunctuation) remove punctuation marks such as ! " # $ % & ’ ( ) * + , - . / : ; < = > ? @ [ ] ^ _ ` { | } ~.

- **stripWhitespace:** (https://www.rdocumentation.org/packages/tm/versions/0.7-1/topics/stripWhitespace) Strip extra white space from a text document. Multiple white space characters are collapsed to a single blank
- **PlainTextDocument:** (https://www.rdocumentation.org/packages/tm/versions/0.7-1/topics/PlainTextDocument) convert document to plain text format

Hide

```
# Help ??tm_map'

# Normalize to small cases
myCorpus <- tm_map(myCorpus, content_transformer(tolower))

# Remove Stop Words
myCorpus <- tm_map(myCorpus, removeWords, stopwords("english"))

# Remove Punctuation
myCorpus <- tm_map(myCorpus, removePunctuation)

# Remove Numbers
 myCorpus <- tm_map(myCorpus, removeNumbers)

# Create plain text documents
myCorpus <- tm_map(myCorpus, PlainTextDocument)

# Stem words in a text document using Porter's stemming algorithm.
myCorpus <- tm_map(myCorpus, stemDocument, "english")

# Strip White Spaces
myCorpus <- tm_map(myCorpus, stripWhitespace)
```

8.3.2 Building Term Document Matrix

Now we will use `TermDocumentMatrix()` to create a `document-term matrix` or `term-document matrix` which is a mathematical matrix that describes the frequency of terms/words/strings that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms. There are various schemes for determining the value that each entry in the matrix should take. Read more on wiki (https://en.wikipedia.org/wiki/Document-term_matrix).

8.3.3 Find most frequent items - N-gram analysis

Most frequently occurred words (uni-gram, bi-gram and tri-gram) are shown in the plot.

Hide

```
uni_token <- function(x) {NGramTokenizer(x, Weka_control(min = 1, max = 1))}
bi_token <- function(x) {NGramTokenizer(x, Weka_control(min = 2, max = 2))}
tri_token <- function(x) {NGramTokenizer(x, Weka_control(min = 3, max = 3))}

uni_tdm <- TermDocumentMatrix(myCorpus, control = list(tokenize = uni_token))
uni_tdm <- removeSparseTerms(uni_tdm, 0.95)
bi_tdm <- TermDocumentMatrix(myCorpus, control = list(tokenize = bi_token))
tri_tdm <- TermDocumentMatrix(myCorpus, control = list(tokenize = tri_token))


uni_corpus <- findFreqTerms(uni_tdm,lowfreq = 20)
bi_corpus <- findFreqTerms(bi_tdm,lowfreq = 10)
tri_corpus <- findFreqTerms(tri_tdm,lowfreq = 5)

uni_corpus_freq <- rowSums(as.matrix(uni_tdm[uni_corpus,]))
uni_corpus_freq <- data.frame(word=names(uni_corpus_freq), frequency=uni_corpus_freq)
bi_corpus_freq <- rowSums(as.matrix(bi_tdm[bi_corpus,]))
bi_corpus_freq <- data.frame(word=names(bi_corpus_freq), frequency=bi_corpus_freq)
tri_corpus_freq <- rowSums(as.matrix(tri_tdm[tri_corpus,]))
tri_corpus_freq <- data.frame(word=names(tri_corpus_freq), frequency=tri_corpus_freq)
```

Hide

```
df1<- uni_corpus_freq[order(-uni_corpus_freq$frequency),][1:20,]

df2 <- bi_corpus_freq[order(-bi_corpus_freq$frequency),][1:20,]

df3<-tri_corpus_freq[order(-tri_corpus_freq$frequency),][1:20,]
```

8.3.4 Unigram: Most Frequently Occured Words

Top 100 frequenty occured words are..

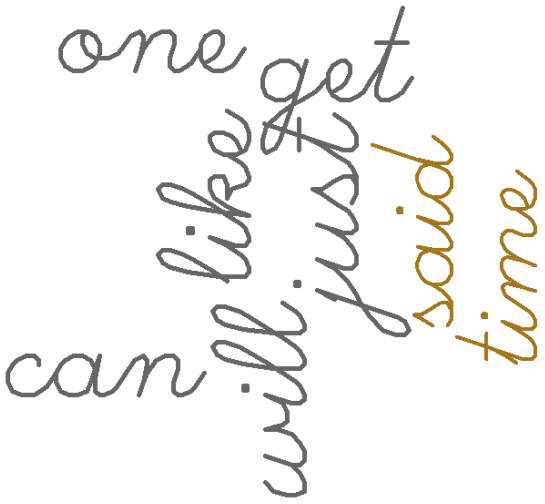
Hide

```
uni_corpus[1:100]
```

```
##   [1] "can"  "get"  "just" "like" "one"  "said" "time" "will" NA     NA
##  [11] NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
##  [21] NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
##  [31] NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
##  [41] NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
##  [51] NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
##  [61] NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
##  [71] NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
##  [81] NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
##  [91] NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
```

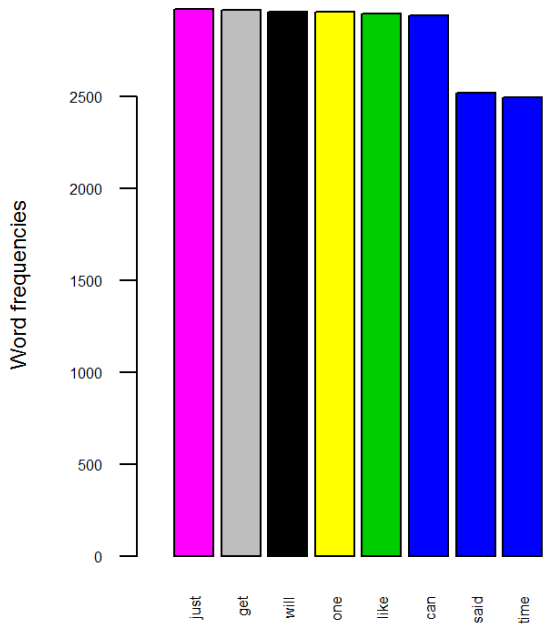
Hide

```
#We have built a word cloud using top 50 frequent words from sample data.
wordcloud(words = uni_corpus_freq$word, freq = uni_corpus_freq$frequency, min.freq = 1,
        max.words=50, random.order=TRUE, rot.per=0.75,
        colors=brewer.pal(8, "Dark2"), c(5,.5), vfont=c("script","plain"))
```



Hide

```
barplot(df1[1:20,]$freq, las = 2, names.arg = df1[1:20,]$word,
        col =df1[1:20,]$freq, main ="",
        ylab = "Word frequencies", cex.axis=.5, cex = .5, cex.lab=0.75, cex.main=.75)
```



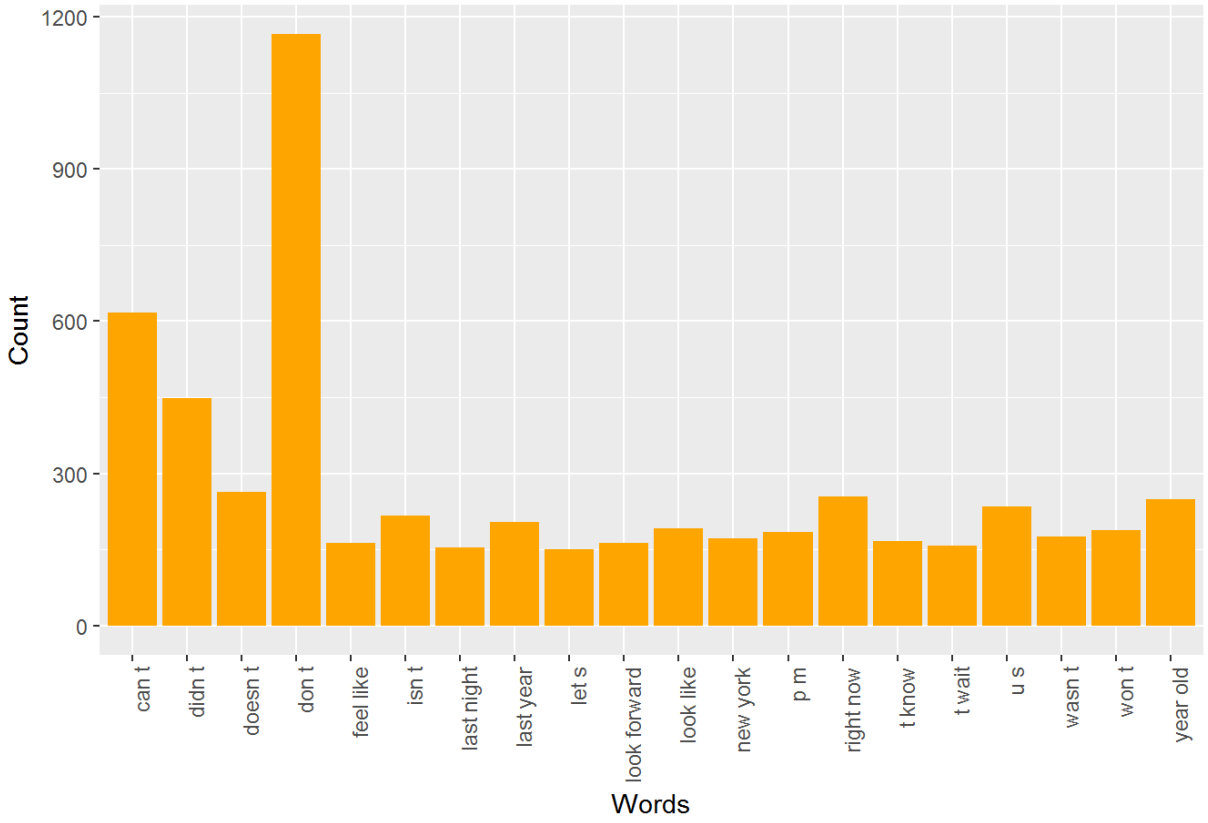
8.3.5 Bigram: Most Frequently occured “sequence of two adjacent elements”

Hide

```
wordcloud(words = bi_corpus_freq$word, freq = bi_corpus_freq$frequency, min.freq = 1,
        max.words=100, random.order=FALSE, rot.per=0.75,
        colors=brewer.pal(8, "Dark2"), c(5,.5), vfont=c("script","plain"))
```

Hide

```
ggplot(df2, aes(x = df2$word, y = frequency)) +
    geom_bar(stat = "identity", fill = "orange") +
    labs(title = " ") +
    xlab("Words") +
    ylab("Count")+
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



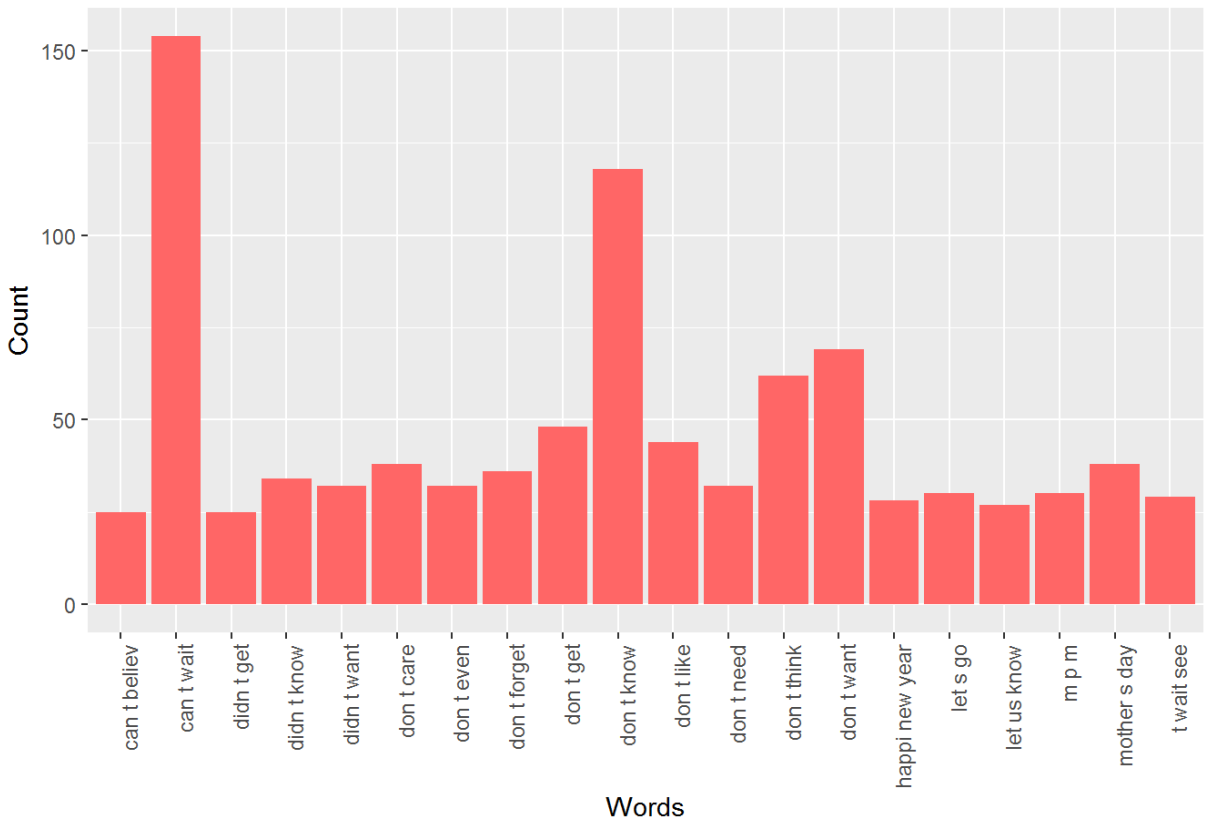
8.3.6 Trigram: Most Frequently occured “a group of three consecutive written units such as letters, syllables, or words”

Hide

```
wordcloud(words = tri_corpus_freq$word, freq = tri_corpus_freq$frequency, min.freq = 1,
        max.words=100, random.order=FALSE, rot.per=0.60,
        colors=brewer.pal(8, "Dark2"), c(5,.5), vfont=c("script","plain"))
```

Hide

```
ggplot(df3, aes(x = df3$word, y = frequency)) +
    geom_bar(stat = "identity", fill = "#FF6666") +
    labs(title = " ") +
    xlab("Words") +
    ylab("Count")+
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



9 Next Steps

- Improve the text analysis
- Develop predictive models to predict text
- Improve model performance
- Implement the predictive models on **R-Shiny** (https://shiny.rstudio.com/)

---

10 R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

10.1 Output Formating options used

```
output:
  html_document:
    toc: true
    toc_depth: 5
    toc_float: true
    number_sections: true
    code_folding: hide
```

To know more on formatting click here http://rmarkdown.rstudio.com/html_document_format.html#table_of_contents (http://rmarkdown.rstudio.com/html_document_format.html#table_of_contents).