Predictive Text Modelling

Mughundhan Chandrasekar

2/15/2017

BRIEF INTRODUCTION

This is a simple paper which briefs out the introductory efforts putforth in-order to build a Shiny-App which accepts a phrase as input and a clever algorithm predicts the subsequent word and returns as output. Word suggestions are given by this shiny-app. Algorithm (or a combination of different machine learning algorithm) is to be selected carefully to produce an efficient result. Text-mining is a vital process which serves as pre-requisite condition in-order to achieve this ambitious task.

1. Setting-Up the Environment

- Several packages are required to be installed and loaded in-order to perform certain operations using R-Studio.
- A suitable environment is to be set (fresh environment without any pre-defined objects) and an appropriate path is set to access the required data-set with ease.
- The required data-sets are imported using appropriate R-code and stored under an easily understandable name as shown in the code chunk below :

```r
rm(list=ls())
library(NLP)
library(tm)
library(R.utils)
library(stringi)
library(RColorBrewer)
library(wordcloud)
library(data.table)
library(ggplot2)
library(SnowballC)
setwd("/Users/Mughundhan/Coursera/final/en_US")
blogs <- readLines("en_US.blogs.txt")
news <- readLines("en_US.news.txt")
twitter <- readLines("en_US.twitter.txt")
```

2. About the file

- We wish to have a look at the general information about the file before we proceed with the analysis.
- This helps us to understand the file better.
- The size of the file and additional information allows the data-scientist to sample a specific amount of data which could be managed by the system's memory.

```r
twitter_summary <- summary(twitter)
blogs_summary <- summary(blogs)
```

```r
news_summary <- summary(news)
cbind(twitter_summary, blogs_summary, news_summary)

##        twitter_summary blogs_summary news_summary
## Length "2360148"       "899288"      "1010242"   
## Class  "character"     "character"   "character" 
## Mode   "character"     "character"   "character" 

Twitter_Size <- file.info("en_US.twitter.txt")$size/1024^2
Blogs_Size <- file.info("en_US.blogs.txt")$size/1024^2
News_Size <- file.info("en_US.news.txt")$size/1024^2
cbind(Twitter_Size, Blogs_Size, News_Size)

##      Twitter_Size Blogs_Size News_Size
## [1,]     159.3641   200.4242  196.2775
```

3. Simple Operations

A. Find the Number of lines in each Text document

- The following chunk of code extracts the number of rows in each of the data-set which is considered as the number of lines in each of the data-sets.

```r
blogs_Lines <- NROW(blogs) #Number of Lines in blogs.txt
news_Lines <- NROW(news) #Number of Lines in news.txt
twitter_Lines <- NROW(twitter) #Number of Lines in twitter.txt
cbind(blogs_Lines, news_Lines, twitter_Lines)

##      blogs_Lines news_Lines twitter_Lines
## [1,]      899288    1010242       2360148
```

We shall infer that the Twitter data-set holds the most number of lines

B. Longest Line in the three data-sets

- A connection is established in-order to perform operations like traversing the data-set to find the number of characters in each line (to find the length of the line).
- Each line is extracted and the length of each line in the data-set is found.
- The line with the maximum length in each data-set is found

```r
conB <- file("en_US.blogs.txt","r")
conT <- file("en_US.twitter.txt","r")
conN <- file("en_US.news.txt","r")
lenB <- nchar(readLines(conB))
lenT <- nchar(readLines(conT))
lenN <- nchar(readLines(conN))
Lol <- cbind(lenB, lenT, lenN)
apply(Lol, 2, max, na.rm=TRUE)

##  lenB  lenT  lenN 
## 40835   213 11384
```

```
close(conB)
close(conT)
close(conN)
```

- We shall infer that the blogs data-set has the lengthiest line among all other given data-sets.
- For smooth functioning of the code we need to terminate the connections which we have opened. Each connection is closed on performing the desired action.

C. Search and match

C.1. Find the no. of occurrences

- To find the number of occurrences of a specific word, that specific word is searched using the *grep()* - with case-sensitivity turned off (in this case).
- A function is used for enhancing re-usability of code.
- In the following chunk of code a function is called to find the number of occurrences of each match (love, hate, biostatistics in this specific case).

```
love <- grep(".love.", twitter, ignore.case = FALSE)
hate <- grep(".hate.", twitter, ignore.case = FALSE)
biostats <- grep(".biostats.", twitter, ignore.case = FALSE)
trans <- function(x) {
  length(x)
}
No_of_love <- trans(love)
No_of_hate <- trans(hate)
No_of_biostats <- trans(biostats)
cbind(No_of_love, No_of_hate, No_of_biostats)

##      No_of_love No_of_hate No_of_biostats
## [1,]      86293      21764              1
```

We shall infer that the word "love" occur more frequently when compared with other two words.

C.2 Return the lines with matching words

The line containing one among the three words (love, hate, biostats) is returned.

```
head(twitter[love])

## [1] "The new sundrop commercial ...hehe love at first sight"
## [2] "I love you, and I'm so proud of you. From sitting on those stairs on
The X Factor, to now. You boys are my inspiration.\342\231\245 :) xx"
## [3] "\"You will allow me to continue to do what I love.\" Big Show's
wife?"
## [4] "Hahahahaha, I love it. Thanks for the quick birthday lesson :)"
## [5] "And thank everyone of you who has taken the time out of your day to
listen to us, your all amazing, much love h2p"
## [6] "I love you brianana"
```

```
head(twitter[hate])

## [1] "Have a great evening/afternoon/whatever it is where you are, folks!"
## [2] "That's game we lost I can't hate on the Heat Good Game"
## [3] "Dallas slapped me on my red as cherries sunburn, and im still crying
aftr 10 mins., i hate him."
## [4] "I hate when i wear bhoops and people are like \" the bigger the hole
the bigger the hoe\" and im just like. O___o bitch try me."
## [5] "Just finished \"one thousand gifts\"....what a read. Just ordered 4
new....hate waiting!"
## [6] "I hate missing school"

twitter[biostats]

## [1] "i know how you feel.. i have biostats on tuesday and i have yet to
study =/"
```

C.3 Search for a phrase - return the no. of lines as well as the exact line

```
phrase <- grep("A computer once beat me at chess, but it was no match for me
at kickboxing", twitter)
twitter[phrase] #The lines

## [1] "A computer once beat me at chess, but it was no match for me at
kickboxing"
## [2] "A computer once beat me at chess, but it was no match for me at
kickboxing"
## [3] "A computer once beat me at chess, but it was no match for me at
kickboxing"

length(phrase) #Total No. of lines

## [1] 3
```

4. Making the data suitable for making analysis

A. Cleaning the data

- We have selected the twitter data-set for performing the text mining operations
- We remove weird symbols in-order to facilitate text mining operations
- Since the twitter data-set is pretty huge to be managed by the system's memory, a specific amount of data (length=10,000) is sampled out from twitter data-set.
- The data-set is stored in a corpus
- The characters in the data-set is converted to lower-case
- The punctuations, numerical values and **stop-words** (words such as the, this, that etc occurs several times in a data-set which adds no value in our text mining purposes and hence these words are to be removed in-order to fetch better results) are removed.
- **stemDocument** - Stemming is an essential step in which several words originate from the same root word and has the same meaning (Eg: walk, walks, walked, walking - all denotes the same thing). Thus, for efficient results, we need to classify these words into a single root-word and use it for text mining purposes

- Finally the whitespaces are removed and the data-set is processed as a plain text document

```
clean_twitter<- iconv(twitter, 'UTF-8', 'ASCII', "byte")
sample_twitter<-sample(clean_twitter, 10000)
docT <- VectorSource(sample_twitter)
corp_twitter <- Corpus(docT)
corp_twitter <- tm_map(corp_twitter, tolower)
#corp_twitter <- tm_map(corp_twitter, removeWords, stopwords("english"))
corp_twitter <- tm_map(corp_twitter, removePunctuation)
corp_twitter <- tm_map(corp_twitter, removeNumbers)
corp_twitter <- tm_map(corp_twitter, stemDocument)
corp_twitter <- tm_map(corp_twitter, stripWhitespace)
corp_twitter <- tm_map(corp_twitter, PlainTextDocument)
```

B. Building a Document Term Matrix

- Transforming the data-set into a document term matrix enables the data to be stored as : (rows, columns) = (document, term)
- The document term matrix is further converted into a matrix in-order to make computations pretty efficiently
- The number of terms and their corresponding frequencies shall be obtained by computing the column sum ansd its further sorted in descending order (based on frequencies)
- write.csv allows us to create a csv file holding the matrix

```
dtm <- DocumentTermMatrix(corp_twitter)
dim(dtm)

## [1] 10000 15737

freq_Mat <- as.matrix(dtm)
freq_Sum <- colSums(freq_Mat)
freq_twitter <- sort(freq_Sum, decreasing = TRUE)
#write.csv(freq_Mat, file="dtm.csv") #Creates a CSV file
```

5. Exploratory Data Analysis

A. Top 10 most frequent words

The top ten words with highest frequencies are displayed in descending order with their corresponding frequencies.

```
head(freq_twitter,10)

##  the  you  and  for that your with have  are just
## 3855 2282 1791 1582  976  720  684  664  645  643

wf <- data.frame(word=names(freq_twitter), freq=freq_twitter)
head(wf,10)

##      word freq
## the   the 3855
## you   you 2282
## and   and 1791
## for   for 1582
## that that  976
## your your  720
## with with  684
## have have  664
## are   are  645
## just just  643
```

We shall infer that the words "just", "like" and "get" occur more frequently than other words (on ignoring the stop words)

B. Top 10 least frequent words

The top ten words with least frequencies are displayed in descending order with their corresponding frequencies.

```
tail(freq_twitter,10)
```

```
##       zinczenko            zine            zipp        zipsters 
##               1               1               1               1 
##         zombies zombiesvampires        zonamaco          zoning 
##               1               1               1               1 
##           zumba          zynga 
##               1               1
```

```
tail(wf,10)
```

```
##                            word freq
## zinczenko             zinczenko    1
## zine                       zine    1
## zipp                       zipp    1
## zipsters               zipsters    1
## zombies                 zombies    1
## zombiesvampires zombiesvampires    1
## zonamaco               zonamaco    1
## zoning                   zoning    1
## zumba                     zumba    1
## zynga                     zynga    1
```

Several words occurs not more than once and they are displayed above (to name a few - eg: zuri", "zumba" and "zone" occur only once)

C. Words occuring 300 times or more

Displays all the words which has a frequency of at-least 300.

```
findFreqTerms(dtm, lowfreq=300)
```

```
##  [1] "about"  "all"    "and"    "are"    "but"    "can"    "day"   
##  [8] "dont"   "for"    "from"   "get"    "good"   "have"   "how"
## [15] "its"    "just"   "know"   "like"   "love"   "not"    "now"
## [22] "out"    "thanks" "that"   "the"    "this"   "was"    "what"
## [29] "when"   "will"   "with"   "you"    "your"
```

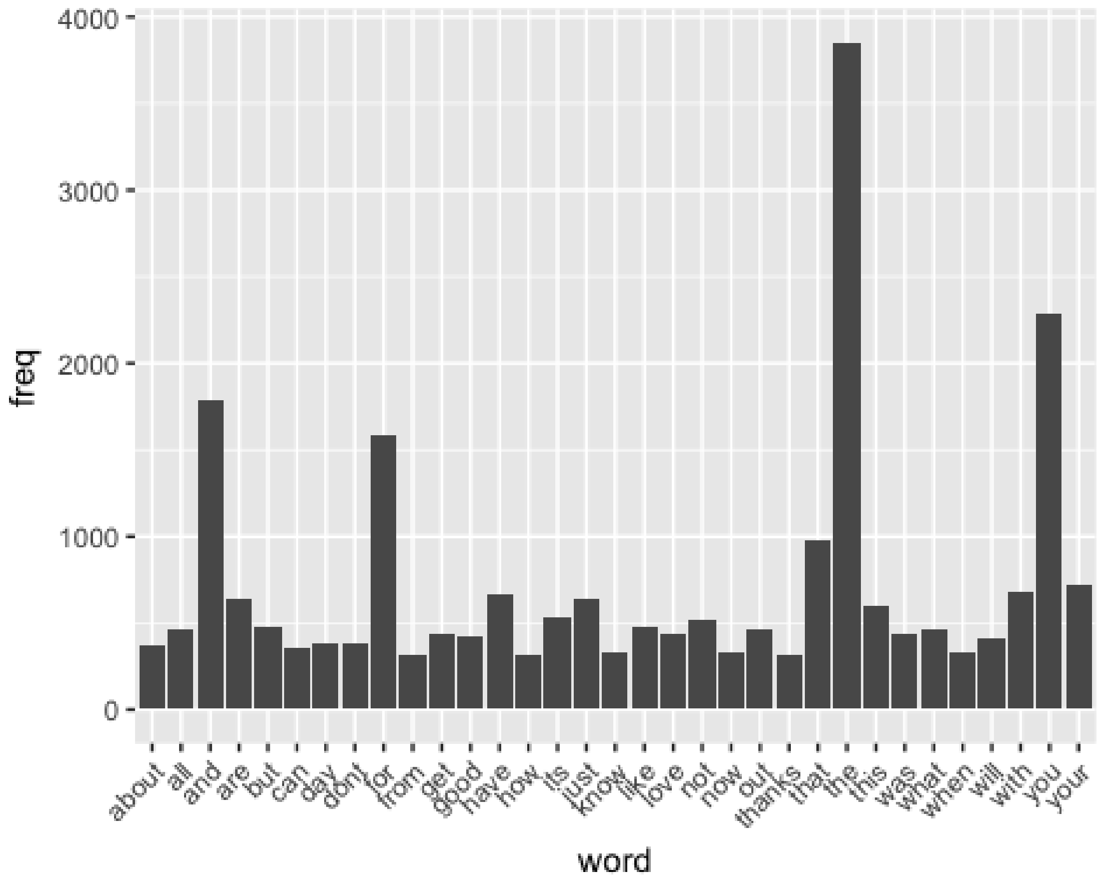
These words occur pretty frequently and shall be used for plotting and correlation purposes in-order to get better understanding about our data-set.

6. Plotting - for better understanding

A. Barplot

- The graphy displays all the words which has a frequency of at-least 300
- Relative strength of each word shall be clearly examined using this barplot

```
p <- ggplot(subset(wf, freq_twitter>300), aes(word, freq))
p <- p + geom_bar(stat="identity")
p <- p + theme(axis.text.x=element_text(angle=45, hjust=1))
p
```



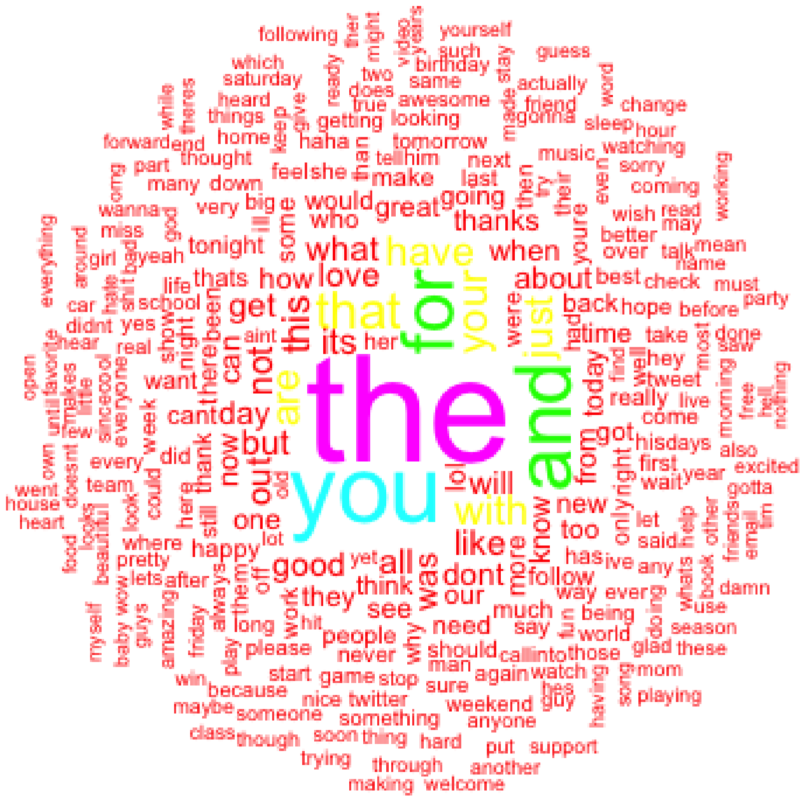
Few aesthetics are used for providing a visual appeal which enhances the understandability

B. Wordcloud

- The top 300 most frequently occuring words - given in rainbow colour
- Domination of a specific word shall be clearly seen with this mode of view

- A seed value is set. This is to make use of the pseudo randomization process involved in R. In future if the same seed value is called, the same set of randomization is simply replicated.

```
set.seed(123)
wordcloud(corp_twitter, max.words = 300, random.order = FALSE,rot.per=0.35,
use.r.layout=FALSE,colors=rainbow(6))
```



SUMMARY

- "just", "like" and "get" occur more frequently.
- "zuri", "zumba" and "zone" occur only once.
- The size of the files along with its summary is given.
- Basic operations like - finding the matching word, search and replace etc are done.
- The length of the lines, number of words and characters in each dataset is found.
- Plots are displayed for better understanding of this milestone report.

FUTURE WORK

- Correlation between words are to be analysed
- Combination of machine learning algorithms and the best combination is to be found (ideally the combination which meets our condition would be used in our future work)
- An efficient shinyApp will be deployed using all the techniques as mentioned earlier