**A complete guide to text processing using Twitter data and R.**

# Why Text Processing using R?

With the increasing importance of computational text analysis in research , many researchers face the challenge of learning how to use advanced software that enables this text analysis. Currently, one of the most popular environments for computational methods and the emerging field of “data science” is the R statistical software. However, for researchers that are not well-versed in programming, learning how to use R can be a challenge, and performing text analysis in particular can seem daunting.

# **Why R?**

R was specifically designed for statistical analysis, which makes it highly suitable for data science applications. Although the learning curve for programming with R can be steep, especially for people without prior programming experience, the tools now available for carrying out text analysis in R make it easy to perform powerful, cutting-edge text analytics using only a few simple commands. One of the keys to R’s explosive growth has been its densely populated collection of extension software libraries, known in R terminology as packages, supplied and maintained by R’s extensive user community. Each package extends the functionality of the base R language and core packages, and in addition to functions and data must include documentation and examples, often in the form of vignettes demonstrating the use of the package. The best-known package repository, the Comprehensive R Archive Network (CRAN), currently has over 10,000 packages that are published.

Twitter("#technology", n=1000,lang = "en")

Convert this extracted data to a dataframe which makes it more readable and easier to work with.

amazon\_tweets <- twListToDF(tweets\_a)

google\_tweets <- twListToDF(tweets\_g)

facebook\_tweets <- twListToDF(tweets\_f)

tech\_tweets <- twListToDF(tweets\_tech)

View(amazon\_tweets)

View(google\_tweets)

View(facebook\_tweets)

View(tech\_tweets)

Below is a code to pre-process the data and remove tabs, blank spaces, links etc. This section can be modified according to one’s requirements.

google\_text<- google\_tweets$text

amazon\_text<- amazon\_tweets$text

facebook\_text<- facebook\_tweets$text

tech\_text<- tech\_tweets$text

#convert all text to lower case

google\_text<- tolower(google\_text)

amazon\_text<- tolower(amazon\_text)

facebook\_text<- tolower(facebook\_text)

tech\_text<- tolower(tech\_text)

# Replace blank space (“rt”)

google\_text <- gsub("rt", "", google\_text)

amazon\_text <- gsub("rt", "", amazon\_text)

facebook\_text <- gsub("rt", "", facebook\_text)

tech\_text <- gsub("rt", "", tech\_text)

# Replace @UserName

google\_text <- gsub("@\\w+", "", google\_text)

amazon\_text <- gsub("@\\w+", "", amazon\_text)

facebook\_text <- gsub("@\\w+", "", facebook\_text)

tech\_text <- gsub("@\\w+", "", tech\_text)

# Remove punctuation

google\_text <- gsub("[[:punct:]]", "", google\_text)

amazon\_text <- gsub("[[:punct:]]", "", amazon\_text)

facebook\_text <- gsub("[[:punct:]]", "", facebook\_text)

tech\_text <- gsub("[[:punct:]]", "", tech\_text)

# Remove links

google\_text <- gsub("http\\w+", "", google\_text)

amazon\_text <- gsub("http\\w+", "", amazon\_text)

facebook\_text <- gsub("http\\w+", "", facebook\_text)

tech\_text <- gsub("http\\w+", "", tech\_text)

# Remove tabs

google\_text <- gsub("[ |\t]{2,}", "", google\_text)

amazon\_text <- gsub("[ |\t]{2,}", "", amazon\_text)

facebook\_text <- gsub("[ |\t]{2,}", "", facebook\_text)

tech\_text <- gsub("[ |\t]{2,}", "", tech\_text)

# Remove blank spaces at the beginning

google\_text <- gsub("^ ", "", google\_text)

amazon\_text <- gsub("^ ", "", amazon\_text)

facebook\_text <- gsub("^ ", "", facebook\_text)

tech\_text <- gsub("^ ", "", tech\_text)

# Remove blank spaces at the end

google\_text <- gsub(" $", "", google\_text)

amazon\_text <- gsub(" $", "", amazon\_text)

facebook\_text <- gsub(" $", "", facebook\_text)

tech\_text <- gsub(" $", "", tech\_text)

# What are Stop Words?

When working with text mining applications, we often hear of the term “stop words” or “stop word list” or even “stop list”. Stop words are basically a set of commonly used words in any language, not just English. The reason why stop words are critical to many applications is that, if we remove the words that are very commonly used in a given language, we can focus on the important words instead.

Stop words are generally thought to be a **“single set of words”**. It really can mean different things to different applications.

#clean up by removing stop words

google\_tweets.text.corpus <- tm\_map(google\_tweets.text.corpus, function(x)removeWords(x,stopwords()))

amazon\_tweets.text.corpus <- tm\_map(amazon\_tweets.text.corpus, function(x)removeWords(x,stopwords()))

facebook\_tweets.text.corpus <- tm\_map(facebook\_tweets.text.corpus, function(x)removeWords(x,stopwords()))

tech\_tweets.text.corpus <- tm\_map(tech\_tweets.text.corpus, function(x)removeWords(x,stopwords()))

We are done pre-processing our data, and are ready to do some analysis.

# **What are Word Clouds?**

Word clouds (also known as text clouds or tag clouds) work in a simple way: the more a specific word appears in a source of textual data (such as a speech, blog post, or database), the bigger and bolder it appears in the word cloud.

So let’s generate some word clouds and find out some of the frequent and important terms being used in the tweets we have extracted.

library("wordcloud")

#generate wordcloud

wordcloud(google\_tweets.text.corpus,min.freq = 10,colors=brewer.pal(8, "Dark2"),random.color = TRUE,max.words = 500)

wordcloud(amazon\_tweets.text.corpus,min.freq = 10,colors=brewer.pal(8, "Dark2"),random.color = TRUE,max.words = 500)

wordcloud(facebook\_tweets.text.corpus,min.freq = 10,colors=brewer.pal(8, "Dark2"),random.color = TRUE,max.words = 500)

wordcloud(tech\_tweets.text.corpus,min.freq = 10,colors=brewer.pal(8, "Dark2"),random.color = TRUE,max.words = 500)