**Assignment-Text Mining**

1. Extract tweets for any user (try choosing a user who has more tweets)
2. Perform sentimental analysis on the tweets extracted from the above

|  |
| --- |
| ### Twitter Sentiment Analysis #### |
|  |  |
|  | install.packages("twitteR") |
|  | library("twitteR") |
|  |  |
|  | install.packages("ROAuth") |
|  | library("ROAuth") |
|  |  |
|  | install.packages("base64enc") |
|  | library(base64enc) |
|  |  |
|  | install.packages("httpuv") |
|  | library(httpuv) |
|  | ### https://apps.twitter.com/ |
|  |  |
|  | cred <- OAuthFactory$new(consumerKey='yPv2Z0W5ZTuiMVaZdAytqH8H0', |
|  | consumerSecret='E1UIuJczP1JRM0VFb36Z1BScMcJobPGtzqx9XUBU2nTkwJcIo7', |
|  | requestURL='https://api.twitter.com/oauth/request\_token', |
|  | accessURL='https://api.twitter.com/oauth/access\_token', |
|  | authURL='https://api.twitter.com/oauth/authorize') |
|  |  |
|  |  |
|  | setup\_twitter\_oauth("yPv2Z0W5ZTuiMVaZdAytqH8H0", |
|  | "E1UIuJczP1JRM0VFb36Z1BScMcJobPGtzqx9XUBU2nTkwJcIo7", |
|  | "995572125254782976-VRxUhTLGjBeUn53q0CJTeIhPliDXjnS", # Access token |
|  | "TPX35FMKDruUOukEaLOZhVE8uxvpBcrZXMnWs8pUwHaW2") # Access token secret key |
|  |  |
|  | tweets <- userTimeline('narendramodi', n = 1000) |
|  | tweetsDF <- twListToDF(tweets) |
|  | write.csv(TweetsDF, "Tweets\_modi.csv") |
|  | getwd() |
|  |  |
|  | word\_tweets<- searchTwitter('lockdown', n=1000, lang="en", resultType = "recent") |
|  | class(word\_tweets) |
|  | word\_tweets[1:20] |
|  |  |
|  | final <-sapply(tweets, function(x) x$getText()) |
|  |  |
|  | str(final) |
|  |  |
|  | #https://apps.twitter.com/ |
|  |  |
|  | library(tm) |
|  | final\_corpus<- Corpus(VectorSource(final)) |
|  | inspect(final\_corpus[100]) |
|  |  |
|  |  |
|  | fin\_clean<-tm\_map(final\_corpus, removePunctuation) |
|  | fin\_clean<-tm\_map(fin\_clean, content\_transformer(tolower)) |
|  | fin\_clean<-tm\_map(fin\_clean, removeWords, stopwords("english")) |
|  | fin\_clean<-tm\_map(fin\_clean,removeNumbers) |
|  | fin\_clean<-tm\_map(fin\_clean, stripWhitespace) |
|  |  |
|  | fin\_clean<-tm\_map(fin\_clean, removeWords, c("gameofthrones")) ## clean some words |
|  |  |
|  | library(wordcloud) |
|  | wordcloud(fin\_clean, random.order = F, max.words = 5,colors=rainbow(50)) |
|  | wordcloud(fin\_clean, rot.per=0.5, random.order=TRUE,colors=brewer.pal(8, "Dark2")) |
|  |  |
|  | # Term document matrix |
|  | # converting unstructured data to structured format using TDM |
|  |  |
|  | tdm <- TermDocumentMatrix(fin\_clean) |
|  | dtm <- t(tdm) |
|  | tdm <- as.matrix(tdm) |
|  |  |
|  | # Bar plot |
|  | w <- rowSums(tdm) |
|  | w |
|  | w\_sub <- subset(w, w >= 20) |
|  | w\_sub |
|  | windows() |
|  | barplot(w\_sub, las=3, col = rainbow(20)) |
|  |  |
|  | # Term mcdonalds repeats in almost all documents |
|  | fin\_clean <- tm\_map(fin\_clean, removeWords, c('apple','air','can','mcds','mcdonalds','macbook','product','windows')) |
|  | fin\_clean <- tm\_map(fin\_clean, stripWhitespace) |
|  | tdm <- TermDocumentMatrix(fin\_clean) |
|  | tdm <- as.matrix(tdm) |
|  |  |
|  | # Word cloud |
|  | install.packages("wordcloud") |
|  | library(wordcloud) |
|  | windows() |
|  | wordcloud(words = names(w\_sub), freq = w\_sub) # wordcloud with only subset of words |
|  |  |
|  | w\_sub1 <- sort(rowSums(tdm), decreasing = TRUE) |
|  | wordcloud(words = names(w\_sub1), freq = w\_sub1) # all words are considered |
|  |  |
|  | windows() |
|  | wordcloud(words = names(w\_sub1), freq = w\_sub1, random.order = F, colors = rainbow(20), scale=c(3,1), rot.per = 0.3) |
|  |  |
|  | # lOADING +VE AND -VE dictonaries |
|  | pos.words = scan(file.choose(), what="character", comment.char=";") # read-in positive-words.txt |
|  | neg.words = scan(file.choose(), what="character", comment.char=";") # read-in negative-words.txt |
|  | pos.words = c(pos.words,"wow", "kudos", "hurray") # including our own positive words to the existing list |
|  |  |
|  | # Positive wordcloud |
|  | pos.matches = match(names(w\_sub1), c(pos.words)) |
|  | pos.matches = !is.na(pos.matches) |
|  | freq\_pos <- w\_sub1[pos.matches] |
|  | p\_names <- names(freq\_pos) |
|  | windows() |
|  | wordcloud(p\_names,freq\_pos,scale=c(4,1),colors = rainbow(20)) |
|  |  |
|  | # Negative wordcloud |
|  | neg.matches = match(names(w\_sub1), c(neg.words)) |
|  | neg.matches = !is.na(neg.matches) |
|  | freq\_neg <- w\_sub1[neg.matches] |
|  | n\_names <- names(freq\_neg) |
|  | windows() |
|  | wordcloud(n\_names,freq\_neg,scale=c(5,1),colors = brewer.pal(8,"Dark2")) |
|  |  |
|  | #### emotion mining #### |
|  | install.packages("syuzhet") |
|  | library("syuzhet") |
|  | library(lubridate,ggplot2) |
|  | library(ggplot2) |
|  | library(scales) |
|  | library(dplyr) |
|  | library(reshape2) |
|  |  |
|  | x <- get\_nrc\_sentiment(fin\_txt) |
|  | head(x,n=5) |
|  |  |
|  | fin\_txt[4] |
|  | get\_nrc\_sentiment('happy') |
|  | get\_nrc\_sentiment('boring') |
|  |  |
|  | get\_sentiment('boring',method="afinn") |
|  | get\_sentiment('happy',method="afinn") |
|  |  |
|  | #each sentences by weight |
|  | example<-get\_sentences(fin\_txt) |
|  | nrc\_data<-get\_nrc\_sentiment(example) |
|  |  |
|  |  |
|  | # Bar plot for emotion mining |
|  | windows() |
|  | barplot(colSums(nrc\_data), las = 1, col = rainbow(10), ylab = 'Count', main = 'Emotion scores') |
|  |  |
|  |  |
|  |  |
|  | sentiment\_vector<-get\_sentiment(example,method="bing") |
|  | sentiment\_afinn<-get\_sentiment(example,method="afinn") |
|  | sentiment\_nrc<-get\_sentiment(example,method="nrc") |
|  |  |
|  | sum(sentiment\_afinn) |
|  | mean(sentiment\_afinn) |
|  | summary(sentiment\_afinn) |
|  |  |
|  | windows() |
|  | plot(sentiment\_vector,type='l',main ='Plot trajectory',xlab='Narative time', ylab='Emotional valence') |
|  | abline(h=0,color='red') |
|  |  |
|  | plot( |
|  | sentiment\_vector, |
|  | type="h", |
|  | main="Example Plot Trajectory", |
|  | xlab = "Narrative Time", |
|  | ylab= "Emotional Valence" |
|  | ) |
|  |  |
|  | ##Shape smoothing and normalization using a Fourier based transformation and low pass filtering is achieved |
|  | #using the get\_transformed\_values function as shown below. |
|  | ft\_values <- get\_transformed\_values( |
|  | sentiment\_vector, |
|  | low\_pass\_size = 3, |
|  | x\_reverse\_len = 100, |
|  | padding\_factor = 2, |
|  | scale\_vals = TRUE, |
|  | scale\_range = FALSE |
|  | ) |
|  | plot( |
|  | ft\_values, |
|  | type ="l", |
|  | main ="Narendra Modi Tweets using Transformed values", |
|  | xlab = "Narrative Time", |
|  | ylab = "Emotional Valence", |
|  | col = "red" |
|  | ) |
|  | #Most Negative and Positive reviews |
|  | negative<-example[which.min(sentiment\_vector)] |
|  | positive<-example[which.max(sentiment\_vector)] |

TWO:

1) Extract reviews of any product from ecommerce website amazon

2) Perform sentimental analysis

|  |
| --- |
| #mining amazon product review |
|  | library(rvest) |
|  | library(XML) |
|  | library(magrittr) |
|  |  |
|  | # Amazon Reviews ############################# |
|  | aurl <- "https://www.amazon.in/Apple-MacBook-Air-13-3-inch-Integrated/product-reviews/B073Q5R6VR/ref=cm\_cr\_arp\_d\_paging\_btm\_3?showViewpoints=1&pageNumber" |
|  | amazon\_reviews <- NULL |
|  | for (i in 1:10){ |
|  | murl <- read\_html(as.character(paste(aurl,i,sep="="))) |
|  | rev <- murl %>% html\_nodes(".review-text") %>% html\_text() |
|  | amazon\_reviews <- c(amazon\_reviews,rev) |
|  | } |
|  | length(amazon\_reviews) |
|  | str(amazon\_reviews) |
|  | txt <- amazon\_reviews |
|  |  |
|  | library(tm) |
|  | corpus<- Corpus(VectorSource(txt)) |
|  | inspect(corpus[100]) |
|  |  |
|  |  |
|  | corp\_clean<-tm\_map(corpus, removePunctuation) |
|  | corp\_clean<-tm\_map(corp\_clean, content\_transformer(tolower)) |
|  | corp\_clean<-tm\_map(corp\_clean, removeWords, stopwords("english")) |
|  | corp\_clean<-tm\_map(corp\_clean,removeNumbers) |
|  | corp\_clean<-tm\_map(corp\_clean, stripWhitespace) |
|  |  |
|  | corp\_clean<-tm\_map(corp\_clean, removeWords, c("gameofthrones")) ## clean some words |
|  |  |
|  | library(wordcloud) |
|  | wordcloud(fin\_clean, random.order = F, max.words = 5,colors=rainbow(50)) |
|  | wordcloud(fin\_clean, rot.per=0.5, random.order=TRUE,colors=brewer.pal(8, "Dark2")) |

THREE:

1. Extract movie reviews for any movie from IMDB and perform sentimental analysis

|  |
| --- |
| library(rvest) |
|  | library(XML) |
|  | library(magrittr) |
|  |  |
|  | # IMDB Reviews ############################# |
|  | aurl <- "https://www.imdb.com/title/tt4154796/reviews?ref\_=tt\_urv" |
|  | imdb\_reviews <- NULL |
|  | for (i in 1:10){ |
|  | murl <- read\_html(as.character(paste(aurl,i,sep="="))) |
|  | rev <- murl %>% |
|  | html\_nodes(".show-more\_\_control") %>% |
|  | html\_text() |
|  | imdb\_reviews <- c(imdb\_reviews,rev) |
|  | } |
|  | length(imdb\_reviews) |
|  | write.table(imdb\_reviews,"endgame.txt",row.names = F) |
|  | getwd() |
|  |  |
|  | str(imdb\_reviews) |
|  | txt <- imdb\_reviews |
|  |  |
|  | library(tm) |
|  | corpus<- Corpus(VectorSource(txt)) |
|  | inspect(corpus[100]) |
|  |  |
|  |  |
|  | corp\_clean<-tm\_map(corpus, removePunctuation) |
|  | corp\_clean<-tm\_map(corp\_clean, content\_transformer(tolower)) |
|  | corp\_clean<-tm\_map(corp\_clean, removeWords, stopwords("english")) |
|  | corp\_clean<-tm\_map(corp\_clean,removeNumbers) |
|  | corp\_clean<-tm\_map(corp\_clean, stripWhitespace) |
|  |  |
|  | corp\_clean<-tm\_map(corp\_clean, removeWords, c("gameofthrones")) ## clean some words |
|  |  |
|  | library(wordcloud) |
|  | wordcloud(fin\_clean, random.order = F, max.words = 5,colors=rainbow(50)) |
|  | wordcloud(fin\_clean, rot.per=0.5, random.order=TRUE,colors=brewer.pal(8, "Dark2")) |
|  |  |
|  | # Term document matrix |
|  | # converting unstructured data to structured format using TDM |
|  |  |
|  | tdm <- TermDocumentMatrix(corp\_clean) |
|  | dtm <- t(tdm) |
|  | tdm <- as.matrix(tdm) |
|  | # Bar plot |
|  | w <- rowSums(tdm) |
|  | w |
|  | w\_sub <- subset(w, w >= 20) |
|  | w\_sub |
|  | windows() |
|  | barplot(w\_sub, las=3, col = rainbow(20)) |
|  |  |
|  | corp\_clean <- tm\_map(corp\_clean, removeWords, c('apple','air','can','mcds','macbook','product','windows')) |
|  | corp\_clean <- tm\_map(corp\_clean, stripWhitespace) |
|  | tdm <- TermDocumentMatrix(corp\_clean) |
|  | tdm <- as.matrix(tdm) |
|  |  |
|  | # Word cloud |
|  | install.packages("wordcloud") |
|  | library(wordcloud) |
|  | windows() |
|  | wordcloud(words = names(w\_sub), freq = w\_sub) # wordcloud with only subset of words |
|  |  |
|  | w\_sub1 <- sort(rowSums(tdm), decreasing = TRUE) |
|  | wordcloud(words = names(w\_sub1), freq = w\_sub1) # all words are considered |
|  |  |
|  | windows() |
|  | wordcloud(words = names(w\_sub1), freq = w\_sub1, random.order = F, colors = rainbow(20), scale=c(3,1), rot.per = 0.3) |
|  |  |
|  | # lOADING +VE AND -VE dictonaries |
|  | pos.words = scan(file.choose(), what="character", comment.char=";") # read-in positive-words.txt |
|  | neg.words = scan(file.choose(), what="character", comment.char=";") # read-in negative-words.txt |
|  | pos.words = c(pos.words,"wow", "kudos", "hurray") # including our own positive words to the existing list |
|  |  |
|  | # Positive wordcloud |
|  | pos.matches = match(names(w\_sub1), c(pos.words)) |
|  | pos.matches = !is.na(pos.matches) |
|  | freq\_pos <- w\_sub1[pos.matches] |
|  | p\_names <- names(freq\_pos) |
|  | windows() |
|  | wordcloud(p\_names,freq\_pos,scale=c(4,1),colors = rainbow(20)) |
|  |  |
|  | # Negative wordcloud |
|  | neg.matches = match(names(w\_sub1), c(neg.words)) |
|  | neg.matches = !is.na(neg.matches) |
|  | freq\_neg <- w\_sub1[neg.matches] |
|  | n\_names <- names(freq\_neg) |
|  | windows() |
|  | wordcloud(n\_names,freq\_neg,scale=c(5,1),colors = brewer.pal(8,"Dark2")) |
|  |  |
|  | #### emotion mining #### |
|  | install.packages("syuzhet") |
|  | library("syuzhet") |
|  | library(lubridate,ggplot2) |
|  | library(ggplot2) |
|  | library(scales) |
|  | library(dplyr) |
|  | library(reshape2) |
|  |  |
|  | x <- get\_nrc\_sentiment(txt) |
|  | head(x,n=5) |
|  |  |
|  | txt[4] |
|  | get\_nrc\_sentiment('happy') |
|  | get\_nrc\_sentiment('boring') |
|  |  |
|  | get\_sentiment('boring',method="afinn") |
|  | get\_sentiment('happy',method="afinn") |
|  |  |
|  | #each sentences by weight |
|  | ex<-get\_sentences(txt) |
|  | nrc\_data<-get\_nrc\_sentiment(ex) |
|  |  |
|  |  |
|  | # Bar plot for emotion mining |
|  | windows() |
|  | barplot(colSums(nrc\_data), las = 1, col = rainbow(10), ylab = 'Count', main = 'Emotion scores') |
|  |  |
|  |  |
|  |  |
|  | sentiment\_vector<-get\_sentiment(ex,method="bing") |
|  | sentiment\_afinn<-get\_sentiment(ex,method="afinn") |
|  | sentiment\_nrc<-get\_sentiment(ex,method="nrc") |
|  |  |
|  | sum(sentiment\_afinn) |
|  | mean(sentiment\_afinn) |
|  | summary(sentiment\_afinn) |
|  |  |
|  | windows() |
|  | plot(sentiment\_vector,type='l',main='Plot trajectory',xlab='Narative time',ylab='Emotional valence') |
|  | abline(h=0,color='red') |
|  |  |
|  | plot( |
|  | sentiment\_vector, |
|  | type="h", |
|  | main="Example Plot Trajectory", |
|  | xlab = "Narrative Time", |
|  | ylab= "Emotional Valence" |
|  | ) |
|  |  |
|  | ##Shape smoothing and normalization using a Fourier based transformation and low pass filtering is achieved |
|  | #using the get\_transformed\_values function as shown below. |
|  | ft\_values <- get\_transformed\_values( |
|  | sentiment\_vector, |
|  | low\_pass\_size = 3, |
|  | x\_reverse\_len = 100, |
|  | padding\_factor = 2, |
|  | scale\_vals = TRUE, |
|  | scale\_range = FALSE |
|  | ) |
|  | plot( |
|  | ft\_values, |
|  | type ="l", |
|  | main ="IMDB Endgame Movie Reviews using Transformed values", |
|  | xlab = "Narrative Time", |
|  | ylab = "Emotional Valence", |
|  | col = "red" |
|  | ) |
|  | #Most Negative and Positive reviews |
|  | negative<-example[which.min(sentiment\_vector)] |
|  | positive<-example[which.max(sentiment\_vector)] |