Project - Social Network Sentiment Analysis

Edinor Junior

Feb 03, 2021

% !TEX encoding = UTF-8 Unicode

Sentiment Analysis on Social Networks

This project was carried out as one of the activities of the Big Data Analytics course with R and Microsoft Azure from Data Scientist Training. The goal is to capture data from the social network Twitter and perform sentiment analysis with the captured data. For this project to be carried out, several packages must be installed and loaded.

The entire project will be described according to its stages. First we will use the sentiment score calculation and then we will use a classifier with the Naive Bayes algorithm.

The business problem

Every company that has an online business or uses the internet to promote its business, knows (or should know) the importance of social networks. However, most still use it only as a way of carrying out marketing actions, without taking advantage of the possibility that the analysis of feelings brings to dynamically know the way people are reacting to the company's news. In this project, I will show one of the many possibilities to carry out the analysis of feelings, using the R language and demonstrating the results in graphs. For this I will use AltaML, a Canadian company, that works developing Machine Learning and Big Data solutions.

For this project, a library called *Utils* was created, which contains some functions for cleaning and processing data extracted from twitter.

```r
# install.packages("twitteR")
# install.packages("httr")
# install.packages("knitr")
# install.packages("rmarkdown")
library(twitteR)
library(httr)
library(knitr)
library(rmarkdown)

# load the librarie with the cleaning functions
source('utils.R')
options(warn=-1)
```

Step 1 - Authentication

The first step is to perform authentication on twitter to consume data via API, for this it is crucial that you have or create a developer account on the platform. I will not explain the step by step to accomplish this because it is not the scope of the script, however, you will find all the necessary details here

```r
# Create the twitter auth
key <- "ATLveMQbDPZKej3sDXmYhObrz"
secret <- "SsnyKZGV6axlNPq10gHF2n21bEP9Q8wU601YVVXXKFtQgkBJed"
token <- "2827834773-EZSfiVYeoclMVpAM8rTUsTRmWOMLqIqm2HN5Zt7"
tokensecret <- "8wxJyEjIGNzf4bfYr6K8jTI6ApKhQL97aVtAEWFH9jUFM"

# Answer 1 (Yes) when ask about the direct connection.
setup_twitter_oauth(key, secret, token, tokensecret)
```

```
## [1] "Using direct authentication"
```

Step 2 - Connection

Here we will test the connection and capture the tweets. The larger your sample, the more accurate your analysis. But collecting data can take time, depending on your internet connection. Start with 100 tweets, as as you increase the amount, it will require more resources from your computer. We will search for tweets with reference to the hashtag #MachineLearning.

```r
# Check user's timeline
userTimeline("altaml_com")
```

```
## [[1]]
## [1] "altaml_com: Interested in working for the AltaML #AppliedAILab? Applications for cohort 3 are op
##
## [[2]]
## [1] "altaml_com: Applications for Cohort 3 of #TheAILab are open! Want to know more about what it's l
##
## [[3]]
## [1] "altaml_com: We're so excited to be a part of this incredible project! #DL #AI #NARW https://t.co
##
## [[4]]
## [1] "altaml_com: Last week, @csa_asc announced the #smartWhales initiative to help detect &amp; monit
##
## [[5]]
## [1] "altaml_com: Applications for Cohort 3 of #TheAILab are open now! Curious what it's like to inter
##
## [[6]]
## [1] "altaml_com: We're looking for a Corporate Accountant! As a member of the Finance team, you play
##
## [[7]]
## [1] "altaml_com: We're hiring a Senior Software Dev! As a Senior Software Dev, you play a crucial rol
##
## [[8]]
## [1] "altaml_com: We are so excited to be a part of this incredible project to help monitor and detect
##
## [[9]]
## [1] "altaml_com: We are excited to be a part of @csa_asc's #smartWhales initiative to help detect &am
##
## [[10]]
## [1] "altaml_com: We're hiring a Sales Enablement Leader! As Sales Enablement Leader, you will enable
##
## [[11]]
## [1] "altaml_com: Our Co-CEO @NicoleLJanssen and TDA @rickdavidsonyeg sat down with @ABInnovates to di
##
## [[12]]
## [1] "altaml_com: #Edmonton is Young, Educated and Growing! We're so honoured to be included in @Edmon
##
## [[13]]
## [1] "altaml_com: Curious how #AI and #ML can really benefit health research and patient outcomes? Che
##
## [[14]]
## [1] "altaml_com: Our @AlphaLayerAI team is looking for a Machine Learning Developer! If you have a pa
##
## [[15]]
## [1] "altaml_com: Today is our live Q&amp;A session with AltaML #AppliedAILab Alumni! Have questions a
##
## [[16]]
## [1] "altaml_com: Our very first cohort of the AltaML #AppliedAILab wrapped up at the end of December
```

```r
# Capture the tweets
about <- "MachineLearning"
tweets <- 100
lang <- "en"
tweetdata = searchTwitter(about, n = tweets, lang = lang)

# Visualize the first few lines of the dataframe
head(tweetdata)
```

```
## [[1]]
## [1] "CoderRetweet: RT @BlackWomenInAI: 100 Days of Code/ Day 33- Registration Form #Tkinter  #Recipes
##
## [[2]]
## [1] "SynergyIT: How Many Axes Does Your Robotic Positioner Need? https://t.co/5UuXORiEdT #uipath #in
##
## [[3]]
## [1] "TheCuriousLuke: RT @KevinClarity: "Real world Machine Learning in #Fintech " by @deepakvraghavan
##
## [[4]]
## [1] "BotKoshur: RT @KevinClarity: "Real world Machine Learning in #Fintech " by @deepakvraghavan\nht
##
## [[5]]
## [1] "Women_who_code: RT @KevinClarity: "Real world Machine Learning in #Fintech " by @deepakvraghavan
##
## [[6]]
## [1] "100DaysOf2020: RT @KevinClarity: "Real world Machine Learning in #Fintech " by @deepakvraghavan
```

Step 3 - Treatment of data collected through text mining

Here we will install the tm package, for text mining. We will convert the collected tweets into an object of the Corpus type, which stores data and metadata and then we will do some cleaning process, such as removing punctuation, converting the data to lowercase letters and removing stopwords (common words in the English language, in this case) .

```r
# install.packages("tm")
# install.packages("SnowballC")
library(SnowballC)
library(tm)
```

```
## Loading required package: NLP
```

```
##
```

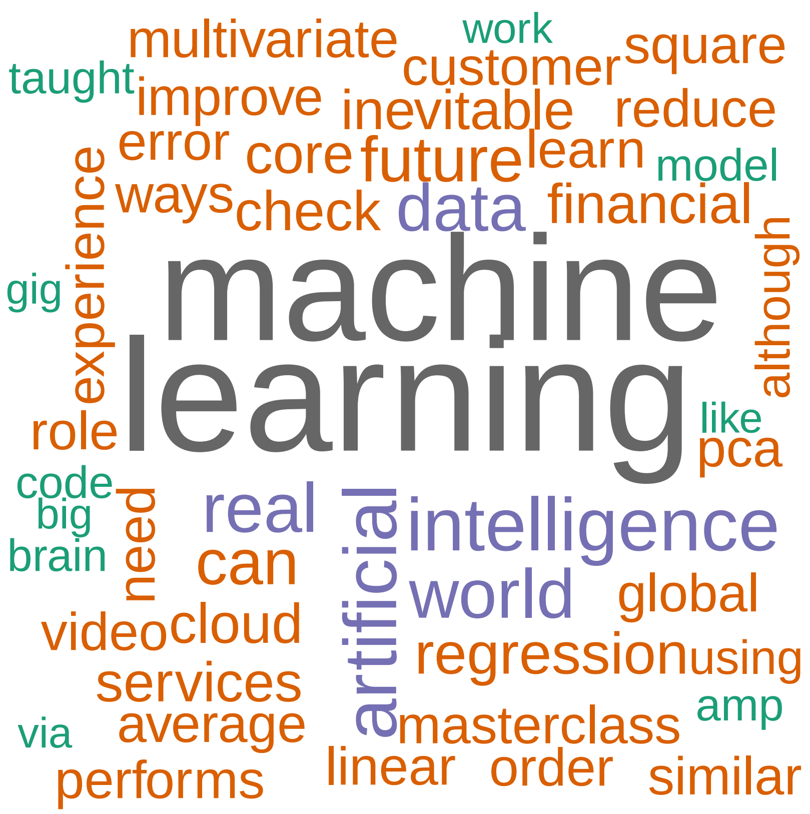
```
## Attaching package: 'NLP'

## The following object is masked from 'package:httr':
##
##     content
```

```
# Treatment of data (cleaning, organization e transformation).
tweetlist <- sapply(tweetdata, function(x) x$getText())
tweetlist <- iconv(tweetlist, to = "utf-8", sub="")
tweetlist <- cleanTweets(tweetlist)
tweetcorpus <- Corpus(VectorSource(tweetlist))
tweetcorpus <- tm_map(tweetcorpus, removePunctuation)
tweetcorpus <- tm_map(tweetcorpus, content_transformer(tolower))
tweetcorpus <- tm_map(tweetcorpus, function(x)removeWords(x, stopwords()))

# Convert the oject Corpus to a text
term_per_document = as.matrix(TermDocumentMatrix(tweetcorpus), control = list(stopwords = c(stopwords("
```

Stage 4 - Wordcloud, association between words and dendrogram

Let's create a word cloud to check the relationship between the words that occur most often. We create a table with the frequency of the words and then generate a dendrogram, which shows how the words relate and are associated with the main theme (in our case, the term MachineLearning).

```
# install.packages("RColorBrewer")
# install.packages("wordcloud")
library(RColorBrewer)
library(wordcloud)

# Create wordcloud
pal2 <- brewer.pal(8,"Dark2")

wordcloud(tweetcorpus,
          min.freq = 2,
          scale = c(5,1),
          random.color = F,
          max.word = 60,
          random.order = F,
          colors = pal2)
```



```r
# Converting the oject text to a matrix format
tweettdm <- TermDocumentMatrix(tweetcorpus)
tweettdm
```

```
## <<TermDocumentMatrix (terms: 179, documents: 100)>>
## Non-/sparse entries: 587/17313
## Sparsity           : 97%
## Maximal term length: 14
## Weighting          : term frequency (tf)
```

```r
# Finding the words has more frequency
findFreqTerms(tweettdm, lowfreq = 11)
```

```
## [1] "learning"     "machine"      "real"         "world"        "artificial"
## [6] "intelligence"
```

```r
# Finding associations
findAssocs(tweettdm, 'datascience', 0.60)
```

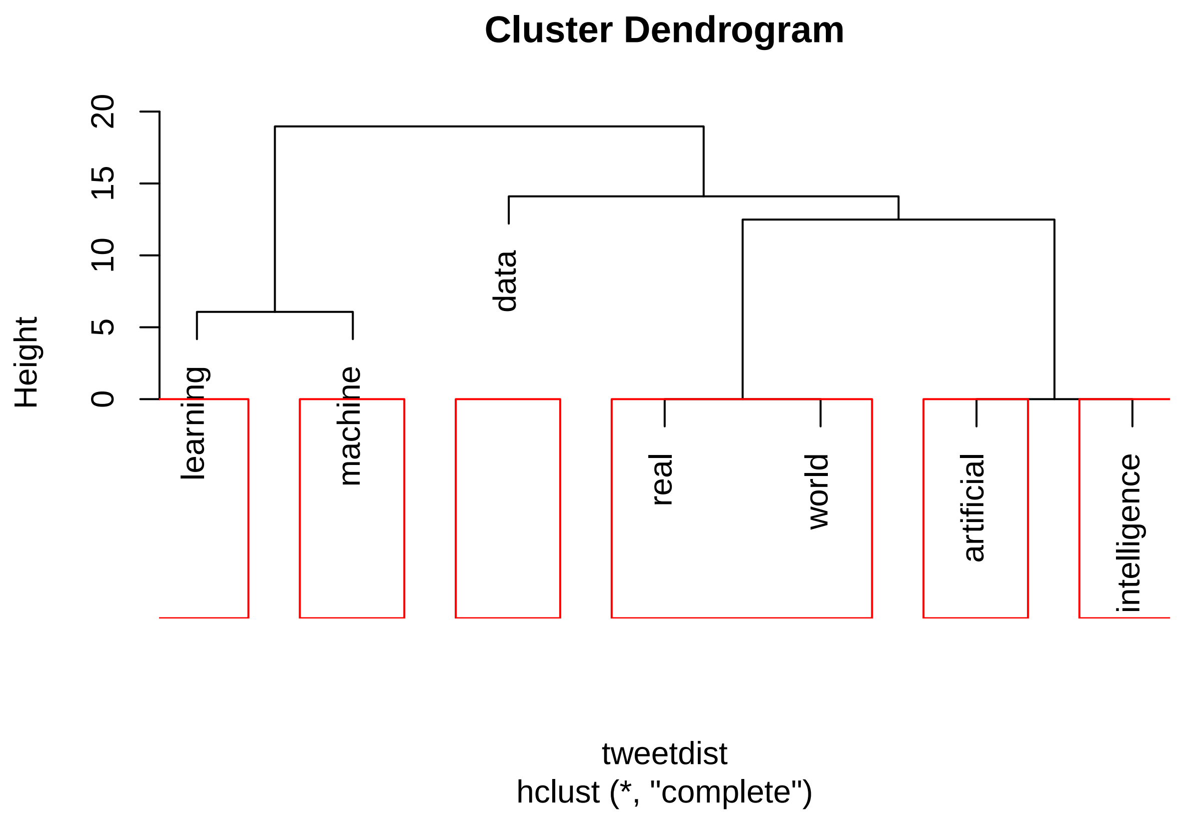
```
## $datascience
## numeric(0)
```

```r
# Removing sparse terms (not used often)
tweet2tdm <-removeSparseTerms(tweettdm, sparse = 0.9)

# Scaling the data
tweet2tdmscale <- scale(tweet2tdm)

# Distance Matrix
```

```r
tweetdist <- dist(tweet2tdmscale, method = "euclidean")

# Prepare the dendrogram
tweetfit <- hclust(tweetdist)

# Creating the dendrogram (checking how words are grouped together)
plot(tweetfit)

# Verify groups
cutree(tweetfit, k = 6)
```

```
##     learning      machine         real        world   artificial intelligence
##            1            2            3            3            4            5
##         data
##            6
```

```r
# Viewing the word groups in the dendrogram
rect.hclust(tweetfit, k = 6, border = "red")
```



Step 5 - Sentiment Analysis

Now we can proceed with the sentiment analysis. We built a function (called sentiment.score) and a list of positive and negative words (these lists are part of this project). Our function checks each item in the data set and compares it with the provided word lists and from there calculates the feeling score, being positive, negative or neutral.

```r
# install.packages("stringr")
# install.packages("plyr")
library(stringr)
library(plyr)
```

```
##
## Attaching package: 'plyr'

## The following object is masked from 'package:twitteR':
##
##     id
```

```r
sentiment.score = function(sentences, pos.words, neg.words, .progress = 'none')
{

  # Build a array from scores with lapply
  scores = laply(sentences,
               function(sentence, pos.words, neg.words)
               {
                 sentence = gsub("[[:punct:]]", "", sentence)
                 sentence = gsub("[[:cntrl:]]", "", sentence)
                 sentence =gsub('\\d+', '', sentence)
                 tryTolower = function(x)
                 {
                   y = NA

                   # Treatment error
                   try_error = tryCatch(tolower(x), error=function(e) e)
                   if (!inherits(try_error, "error"))
                     y = tolower(x)
                   return(y)
                 }

                 sentence = sapply(sentence, tryTolower)
                 word.list = str_split(sentence, "\\s+")
                 words = unlist(word.list)
                 pos.matches = match(words, pos.words)
                 neg.matches = match(words, neg.words)
                 pos.matches = !is.na(pos.matches)
                 neg.matches = !is.na(neg.matches)
                 score = sum(pos.matches) - sum(neg.matches)
                 return(score)
               }, pos.words, neg.words, .progress = .progress )

  scores.df = data.frame(text = sentences, score = scores)
  return(scores.df)
}

# Mapping the positives and negatives words
pos = readLines("positives_words.txt")
neg = readLines("negatives_words.txt")

# Create data from test
test = c("Big Data is the future", "awesome experience",
         "analytics could not be bad", "learn to use big data")
```

```r
# Test our function with our dummy data
testsentiment = sentiment.score(test, pos, neg)
class(testsentiment)
```

```
## [1] "data.frame"
```

```r
# Checking the score
# 0 - expression has no word in our positive and negative word lists or
# found a negative and a positive word in the same sentence
# 1 - expression has a word with a positive connotation
# -1 - expression has a negative connotation
testsentiment$score
```

```
## [1]  0  1 -1  0
```

Step 6 - Generating a Sentiment Analysis Score

With the calculated score, we will separate by country, in this case Canada and the USA, as a way to compare sentiment in different regions. We then generate a boxplot and histogram using the lattice package.

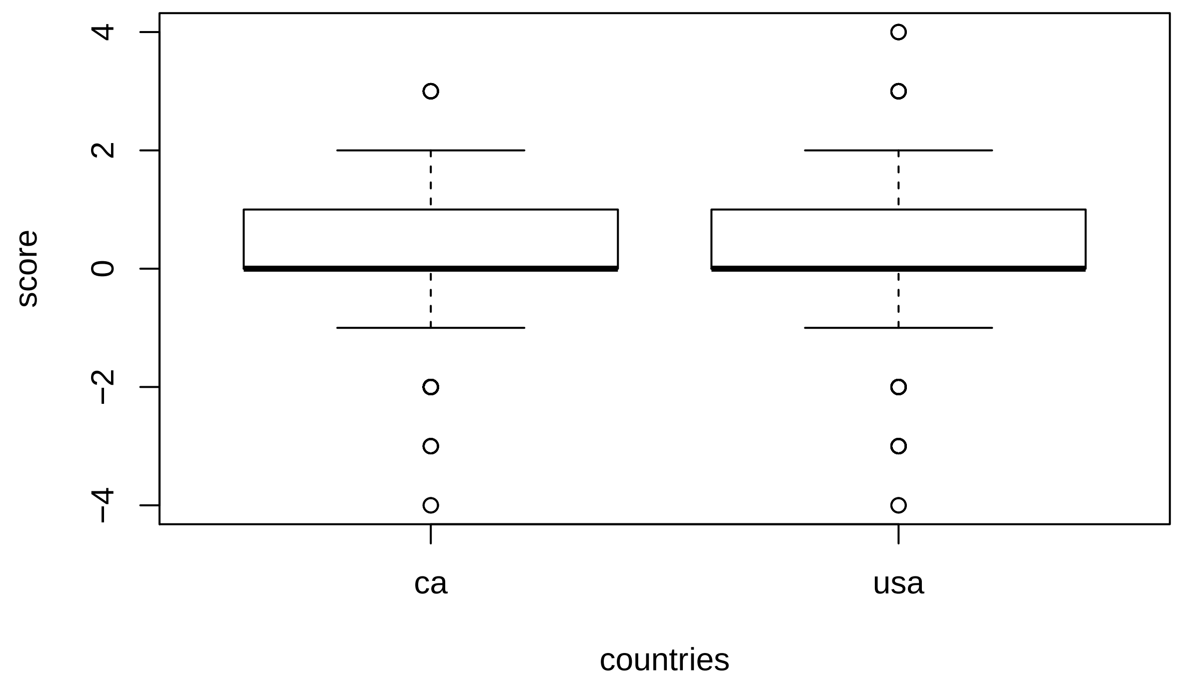
```r
# Tweets by country
catweets = searchTwitter("ca", n = 500, lang = "en")
usatweets = searchTwitter("usa", n = 500, lang = "en")

# Capture text
catxt = sapply(catweets, function(x) x$getText())
usatxt = sapply(usatweets, function(x) x$getText())

# Text vector by country
countryTweet = c(length(catxt), length(usatxt))

# Join the texts
countries = c(catxt, usatxt)

# Apply the function to calculate sentiment score
scores = sentiment.score(countries, pos, neg, .progress = 'text')
```

```
##   |                                                                             |
```

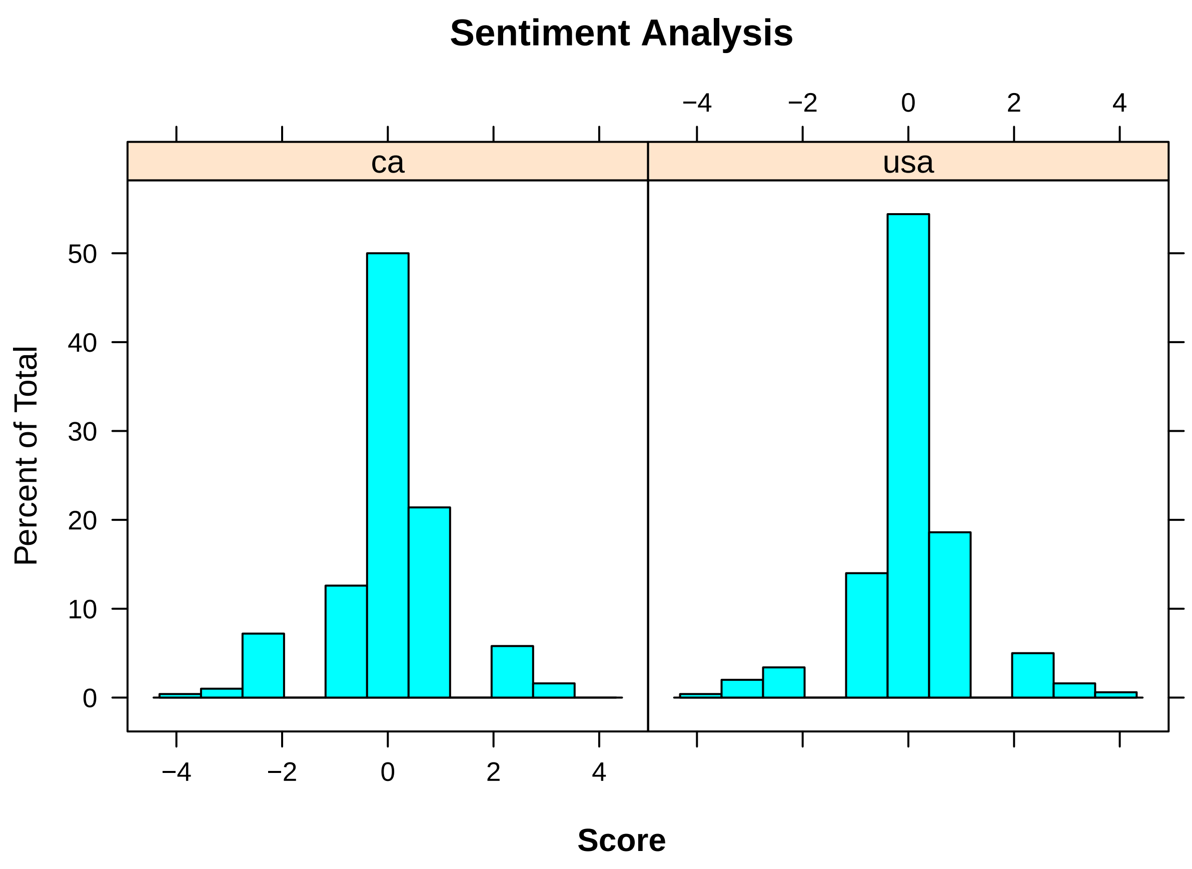
```r
# Calculate the score by country
scores$countries = factor(rep(c("ca", "usa"), countryTweet))
scores$very.pos = as.numeric(scores$score >= 1)
scores$very.neg = as.numeric(scores$score <= -1)

# Calculate the total
numpos = sum(scores$very.pos)
numneg = sum(scores$very.neg)

# Score global
global_score = round( 100 * numpos / (numpos + numneg) )
head(scores)
```

```
##
## 1          @JJWeisman @marissenmark @ACCRES_ca @itsupinsmoke @rebeccaleebligh Indeed, not if they do
## 2                      @2Duefer I was there.  It was off the charts.  5000 + like minded people no BS
## 3  @AliceFromQueens @KevinMKruse @HoosierNative86 You would think a CA senator would have some inter
## 4                                          Dollar Tree is hiring in #UnionCity, CA! Click the link i
## 5 RT @PierrePoilievre: .@erinotoole will fight for our energy sector and secure our future. \n\n"Erin
## 6                                            RT @sheiswarmth: midsummer eve (details), ca. 1908. edv
##   score countries very.pos very.neg
## 1     0        ca        0        0
## 2    -1        ca        0        1
## 3     0        ca        0        0
## 4     0        ca        0        0
## 5     1        ca        1        0
## 6     0        ca        0        0
```

```r
boxplot(score ~ countries, data = scores)

# Generate a histogram with lattice
# install.packages("lattice")
library("lattice")
```



```r
histogram(data = scores, ~score|countries, main = "Sentiment Analysis", xlab = "", sub = "Score")
```



Using Naive Bayes Classifier for sentiment analysis

Here we will do the sentiment analysis in a similar way as seen previously, but using the sentiment package. This package has been discontinued from CRAN, as it will no longer be updated, but can still be obtained through the CRAN archives link. The packages are available together with the project files and the installation procedure is described below.

```
# install.packages("~/FCD/BigDataRAzure/Mini-Projeto01/Rstem_0.4-1.tar.gz", repos = NULL, type = "sourc
# install.packages("~/FCD/BigDataRAzure/Mini-Projeto01/sentiment_0.2.tar.gz", repos = NULL, type = "sou
# install.packages("ggplot2")
library(Rstem)
```

```
##
## Attaching package: 'Rstem'

## The following objects are masked from 'package:SnowballC':
##
##     getStemLanguages, wordStem
```

```
library(sentiment)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':
##
##     annotate
```

Collecting Tweets

The collection of tweets is done using the searchTwitter () function of the twitteR package.

```r
# Collecting tweets
tweets = searchTwitter("MachineLearning", n = 1500, lang = "en")

# Obtained the text
tweets = sapply(tweets, function(x) x$getText())
```

Cleaning, Organizing and Transforming Data

Here regular expressions, using the gsub () function to remove characters that can hinder the analysis process.

```r
# Remove characters
tweets = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", tweets)
# Remove @
tweets = gsub("@\\w+", "", tweets)
# Remove punctuation
tweets = gsub("[[:punct:]]", "", tweets)
# Remove digits
tweets = gsub("[[:digit:]]", "", tweets)
# Remove links html
tweets = gsub("http\\w+", "", tweets)
# Remove unnecessary spaces
tweets = gsub("[ \t]{2,}", "", tweets)
tweets = gsub("^\\s+|\\s+$", "", tweets)

# Create function to tolower
try.error = function(x)
{
  # Create missing value
  y = NA
  try_error = tryCatch(tolower(x), error=function(e) e)
  if (!inherits(try_error, "error"))
    y = tolower(x)
  return(y)
}

# Lower case
tweets = sapply(tweets, try.error)

# Remove the NAs values
tweets = tweets[!is.na(tweets)]
names(tweets) = NULL
```

Naive Bayes Classifier

We use the classify_emotion() and classify_polarity() functions from the sentiment package, which use the algebraic Naive Bayes for sentiment analysis. In this case, the algorithm itself classifies the words and we do not need to create lists of positive and negative words.

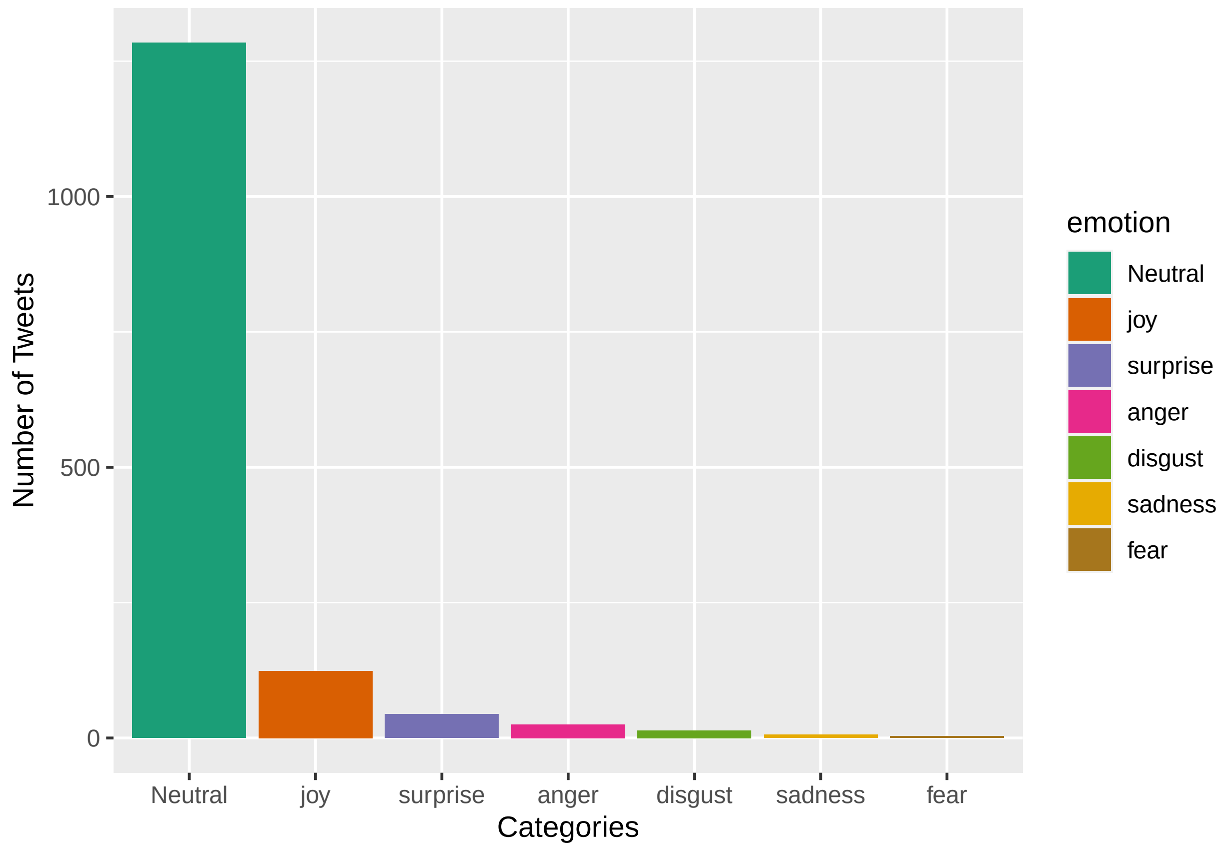
```r
# Classify emotions
class_emo = classify_emotion(tweets, algorithm = "bayes", prior = 1.0)
emotion = class_emo[,7]
```

```r
# Replacing NA's with "Neutral"
emotion[is.na(emotion)] = "Neutral"

# Classify Polarity
class_pol = classify_polarity(tweets, algorithm = "bayes")
polarity = class_pol[,4]

# Build a dataframe with the result
sent_df = data.frame(text = tweets, emotion = emotion,
                     polarity = polarity, stringsAsFactors = FALSE)

# Ordering dataframe
sent_df = within(sent_df,
                 emotion <- factor(emotion, levels = names(sort(table(emotion),
                                                          decreasing=TRUE))))
```

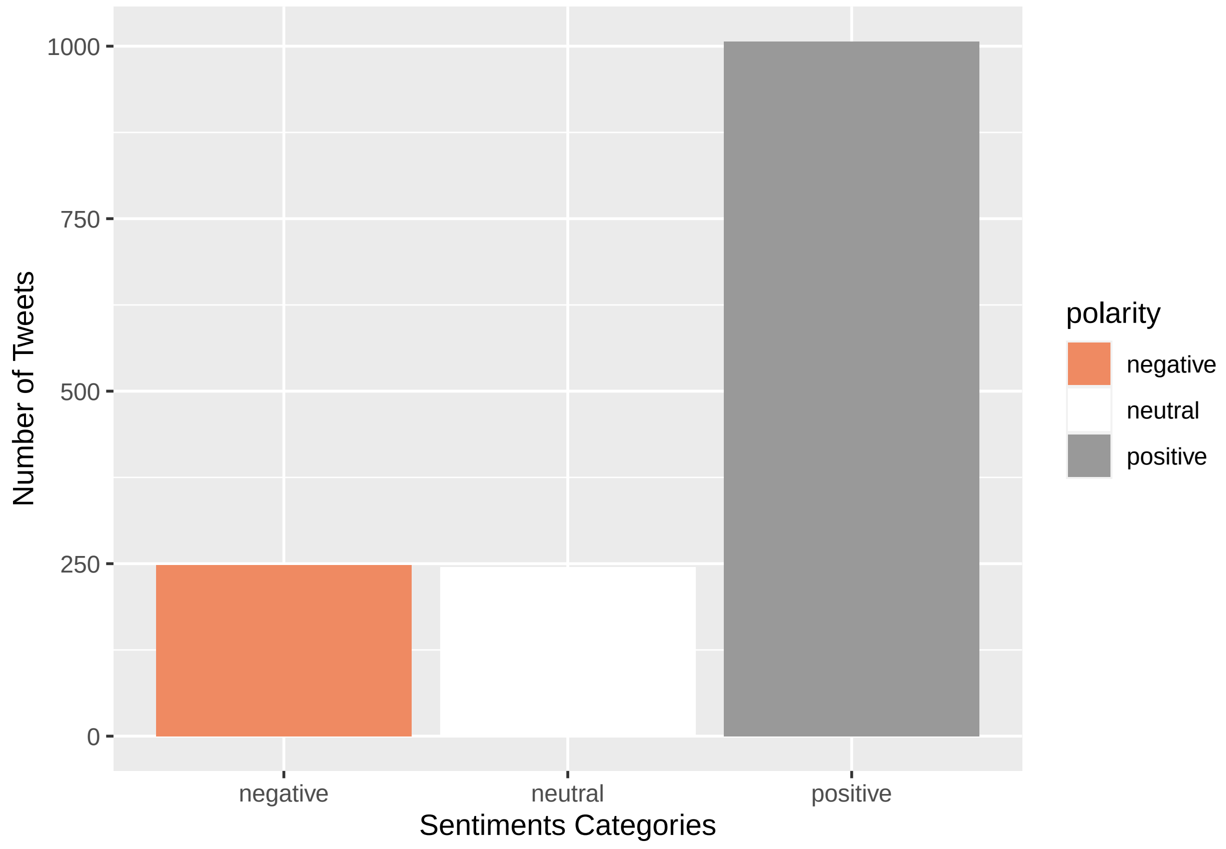
Preview

Finally, we use ggplot2 to view the results.

```r
# Emotions found
ggplot(sent_df, aes(x = emotion)) +
  geom_bar(aes(y = ..count.., fill = emotion)) +
  scale_fill_brewer(palette = "Dark2") +
  labs(x = "Categories", y = "Number of Tweets")
```



```r
# Polarity
ggplot(sent_df, aes(x = polarity)) +
  geom_bar(aes(y = ..count.., fill = polarity)) +
  scale_fill_brewer(palette = "RdGy") +
  labs(x = "Sentiments Categories", y = "Number of Tweets")
```


Number of Tweets
negative
neutral
positive
Sentiments Categories
polarity
negative
neutral
positive