Mini-Projeto 01 - Sentiment Analisys I

Franklin Ferreira

12 de fevereiro, 2020

% !TEX encoding = UTF-8 Unicode

Mini-Projeto 01 - Sentiment Analisys (Análise de sentimentos) I

O objetivo desta análise é explorar diferentes técnicas e ferramentas para a captura, manipulação e transformação de dados proveninentes da rede social Twitter. A análise a ser feita buscará entender os sentimentos que cada Tweet transmite para permitir a extração de informação, conhecimento e sabedoria.

Esta técnica visa auxilar os tomadores de decisão na análise dos sentimentos do seu público alvo em relação a um determinado tema. Como por exemplo, determinar se uma campanha de marketing apresenta uma aceitação positiva, negativa ou neutra.

Toda a análise foi construída em 4 etapas. O projeto completo, bem como todos os arquivos auxiliares utilizados para a criação deste projeto podem ser encontrados no link do github ao final desta análise.

Importando bibliotecas necessárias

```
# Importando bibliotecas necessárias para o uso do rmarkdown.

# install.packages("knitr")
# install.packages("rmarkdown")

library(knitr)
library(rmarkdown)
library(latexpdf)

## Etapas 1 e 2 - Pacotes para se conectar com o Twitter.

# install.packages("twitteR")
# install.packages("httr")

library(twitteR)
library(httr)

## Etapa 3 - Instalando o pacote para Text Mining.

# install.packages("tm")

library(tm)
```

```
## Etapa 4 - Instalando os pacotes necessários para a criação dos gráficos.

# install.packages("RColorBrewer")
# install.packages("wordcloud")
# install.packages("ggdendro")
# install.packages("dendextend")
# library(devtools)
# install_github("lchiffon/wordcloud2")
# install.packages("dplyr")
# install.packages("stringr")

library(RColorBrewer)
library(wordcloud)
library(ggdendro)
library(dendextend)
library(wordcloud2)
library(dplyr)
library(stringr)
```

Funções auxiliares

Define-se algumas funções auxiliares para automatizar as tarefas de Data Munging e o cálculo do scores de sentimentos de um Tweet.

```
####
## Definindo funções auxiliares.
####

# Funções que computa a polaridade de uma sentença (contabiliza o número de palavras
# positivas e negativas).

feelingsScore <- function(sentences, posWords, negWords) {

  # Criando um array de scores com lapply

  scores = lapply(sentences,
                  function(sentence, posWords, negWords) {

                    # Separa palavras presentes na sentença.

                    wordList = str_split(sentence, "\\s+")

                    # Converte a lista de palavras em um vetor.

                    words = unlist(wordList)

                    # Identifica o número de palavras positivas e negativas que foram
                    # encontradas na sentença. O valor NA é retornado caso a palavra não
                    # esteja presente dentro de uma das listas.

                    posMatches = match(words, posWords)
```

```
                    negMatches = match(words, negWords)

                    posMatches = !is.na(posMatches)
                    negMatches = !is.na(negMatches)

                    # Contabiliza o score total da sentença.

                    score = sum(posMatches) - sum(negMatches)

                    return(score)

                  }, posWords, negWords)

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

# Função que realiza uma limpeza nos textos capturados de tweets.

cleanData <- function(tweet) {

  # Remove links http

  tweet = gsub("(f|ht)(tp)(s?)(://)(.*)[.|/](.*)", " ", tweet)
  tweet = gsub("http\\w+", "", tweet)

  # Remove retweets

  tweet = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", " ", tweet)

  # Remove "#Hashtag"

  tweet = gsub("#\\w+", " ", tweet)

  # Remove nomes de usuarios "@people"

  tweet = gsub("@\\w+", " ", tweet)

  # Remove pontuacão

  tweet = gsub("[[:punct:]]", " ", tweet)

  # Remove os números

  tweet = gsub("[[:digit:]]", " ", tweet)

  # Remove espacos desnecessários

  tweet = gsub("[ \t]{2,}", " ", tweet)

  tweet = gsub("^\\s+|\\s+$", "", tweet)

  # Convertendo encoding de caracteres e convertendo para letra minúscula
```

```
  tweet = stringi::stri_trans_general(tweet, "latin-ascii")

  tweet = tryTolower(tweet)

  tweet = tweet[!is.na(tweet)]
}

# Converte caracateres maiúsculos para minúsculos.

tryTolower = function(x) {

  # Cria um dado missing (NA).

  y = NA

  # Executa um tramento de erro caso ocorra.

  try_error = tryCatch(tolower(x), error = function(e) e)

  # Se não houver erro, converte os caracteres.

  if (!inherits(try_error, "error"))
    y = tolower(x)

  return(y)
}
```

Etapa 1 - Executando a autenticação para se conectar com o Twitter

Utiliza-se o pacote *twitterR* para estabelecer uma conexão com o Twitter. Note que ao efetuar o acesso, é necessário que se tenha uma conta nesta rede social e que possua as chaves de autenticação solicitadas para o estabelecimento da conexão. Caso não tenha as chaves, pode obtê-las aqui: https://apps.twitter.com/.

```
# Definindo as chaves de autenticação no Twitter.

key         <- "Insert your key here!"
secret      <- "Insert your secret here!"
token       <- "Insert your token here!"
tokenSecret <- "Insert your token secret here!"

# Realizando o processo de autenticação para iniciar uma sesssão com o twitteR.
#
#> Digite 1 quando for solicitado a utilização da direct connection.

setup_twitter_oauth(key, secret, token, tokenSecret)
```

Etapa 2 - Efetuando a conexão e captura dos tweets

O modo de captura dos tweets irá variar de acordo com a finalidade do projeto. Por exemplo, caso desejasse capturar os tweets de uma determinada timeline, poderia executar as funções a seguir:

```
# Capturando tweets de uma timeline específica.

user <- "dsacademybr"

tweets <- userTimeline(user = user, n = 100)
```

Note que 100 mensagens da timeline 'dsacademybr' foram capturadas.

Mesmo que a análise de uma timeline específica renda valiosas informações, para este projeto optamos por capturar as mensagens diretamente do stream de Tweets da rede social. Com isso, desejamos aumentar a variedade e a pluradidade das informações obtidas para o estudo.

Os primeiros 700 tweets que contiverem a palavra-chave 'Machine Learning' e que forem proveninentes da língua inglesa serão capturados.

```
# Iniciando uma busca por tweets que contenham a string tema definida.

theme    <- "Machine Learning"
language <- "en"

nTweets <- 700

tweetData <- searchTwitter(searchString = theme, n = nTweets, lang = language)

# Visualizando as primeiras linhas do objeto tweetData.

head(tweetData, 2)
```

[[1]] [1] "burucu_osman: RT:@DataScienceCtrlGartner and Forrester Weigh in on Automated Machine Learning https://t.co/Q19w5EFBuX"

[[2]] [1] "fatimapaschal4: RT @arxiv_cshc: Feeling Anxious? Perceiving Anxiety in Tweets using Machine Learning https://t.co/qHHsLInbd9"

Etapa 3 - Realizando o tratamento dos dados coletados através de text mining

As etapas a seguir limpam, organizam e transformam os textos de cada tweet.

```
# Extraindo os textos de cada Tweet e aplicando um enconding para evitar que palavras
# acentuadas sejam distorcidas.

tweetList <- sapply(tweetData, function(tweet){ enc2native(tweet$getText()) })

# Executando a limpeza dos textos de cada Tweet (remoção de links, retweets, #Hashtag,
# etc).

tweetList <- cleanData(tweetList)

# Exibindo os dois primeiros tweets da lista após o processo de limpeza.

tweetList[1:2]
```

[1] “and forrester weigh in on automated machine learning”
[2] “feeling anxious perceiving anxiety in tweets using machine learning”

```
# Convertendo a lista de textos dos tweets para o Classe Corpus.

tweetCorpus <- Corpus(VectorSource(tweetList))

# Removendo as pontuações dos textos.

tweetCorpus <- tm_map(tweetCorpus, removePunctuation)

# Removendo stopwords dos textos dos tweets.

tweetCorpus <- tm_map(tweetCorpus, function(x){ removeWords(x, stopwords()) })
```

Etapa 4 - Criando Wordclouds, associação entre as palavras e um dendograma

Nesta etapa, busca-se identificar através de elementos visuais, os realcionamentos entre as palavras mais recorrentes em tweets que contenham a palavra-chave ‘Machine Learning’.

Wordcloud I

Quanto maior for o número de ocorrências de uma determinda palavra, maior será seu tamanho dentro da wordcloud. Da mesma forma, as cores e o posicionamento das palavras mais recorrentes também se diferenciam.

```
#
## Criando uma Wordcloud.
#

# Definindo a palheta de cores a ser utilizada na wordcloud.

pallete <- brewer.pal(n = 8, name = "Dark2")

# Criando uma wordcloud.

par(mar = c(0,0,0,0))

wordcloud(words        = tweetCorpus,
          min.freq     = 0,
          scale        = c(3,1),
          random.color = F,
          random.order = F,
          colors       = pallete)
```

Wordcloud II

Visualizando as palavras em uma wordcloud interativa.

```
# Criando uma wordcloud interativa.

freqWords <- as.matrix(TermDocumentMatrix(tweetCorpus))

freqWords <- as.data.frame(apply(freqWords, 1, sum))

freqWords <- data.frame(word = rownames(freqWords), freq = freqWords[ ,1])

wordcloud2(freqWords)
```

Computando algumas estatísticas.

```
# Convertendo o objeto corpus com os tweets para um objeto do tipo TermDocumentMatrix.

tweetTDM <- TermDocumentMatrix(tweetCorpus)

tweetTDM
```

```
## <<TermDocumentMatrix (terms: 1374, documents: 700)>>
```

```
## Non-/sparse entries: 5849/955951
## Sparsity           : 99%
## Maximal term length: 16
## Weighting          : term frequency (tf)
```

Podemos visualizar as palavras mais recorrentes de acordo com sua frequência.

```
# Visualizando a matriz de termos por documento.
#
#> Esta matriz exibe o número de vezes que uma determinada palavra apareceu dentro do
#  texto de um tweet.

termPerDocument <- as.matrix(tweetTDM)

# Identificando as palavaras que aparecem com frequência igual ou maior do que a
# frequência especificada dentro dos textos ddos tweets capturados.

findFreqTerms(tweetTDM, lowfreq = 50)
```

```
##  [1] "learning"     "machine"      "using"        "science"      "artificial"
##  [6] "data"         "available"    "courses"      "deep"         "free"
## [11] "intelligence" "stanford"     "learn"
```

Podemos calcular o quão associadas duas palavras estão dentro do conjunto de dados gerado.

```
# Computando as correlações de todas as palavras identificadas com a palavra 'data' e
# exibindo aquelas que apresentaram uma corelação maior do que o limite especificado.

assoc <- findAssocs(tweetTDM, terms = 'data', corlimit = 0.1)

# Exibindo as 5 primeiras associações calculadas.

assoc$data[1:5]
```

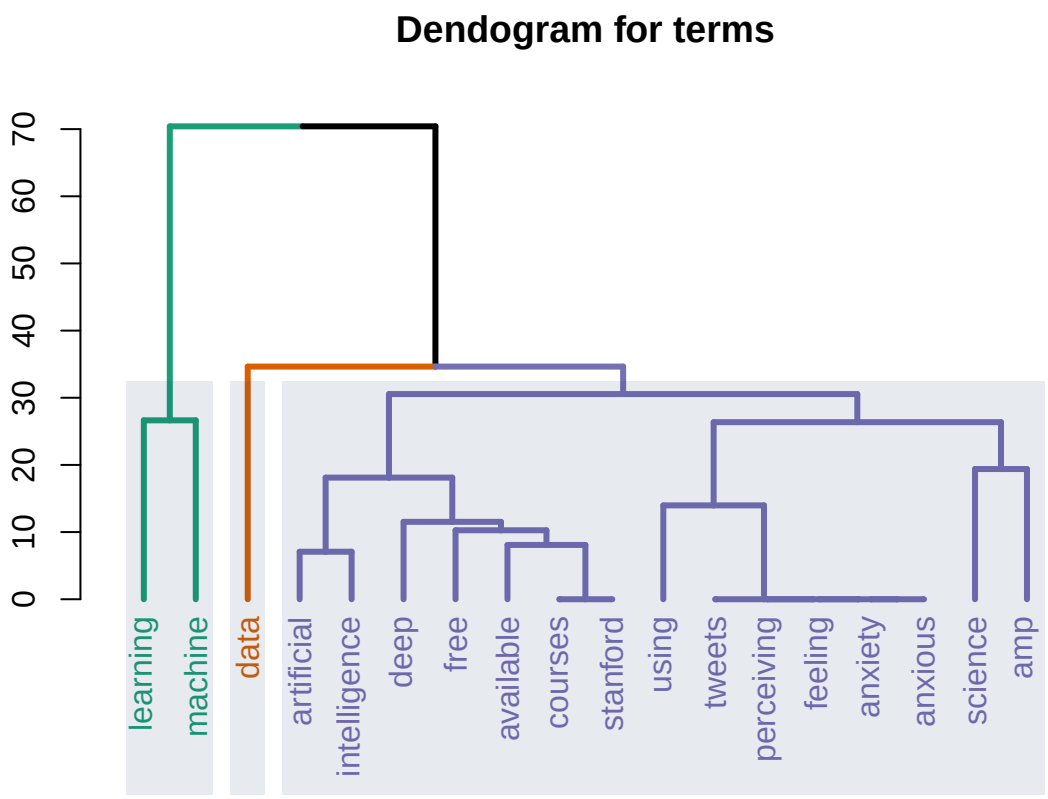
```
##   science    making  decision platforms  scalable
##      0.70      0.45      0.44      0.44      0.44
```

Dendograma

O dendograma a seguir permite visualizar como as palavras estão hierarquicamente relacionadas e como poderiam ser agrupadas em 3 clusters.

```
# Removendo termos esparsos (não utilizados frequentemente) do term-document.

tweetTDMNonSparse <- removeSparseTerms(tweetTDM, sparse = 0.95)

# Criando escala nos dados.

tweetTDMScale <- scale(tweetTDMNonSparse)

# Computando as distâncias euclidianas entre as palavras presentes no term-document.
```

```r
tweetDist <- dist(tweetTDMScale, method = "euclidean")

#
## Criando dendogram.
#

# Executando uma análise hierárquica de cluster sobre os dados de distância dos termos
# selecionados.

tweetFit <- hclust(tweetDist)

# Convertendo os resultados  da análise para um objeto do tipo dendogram.

dend <- as.dendrogram(tweetFit)

# Definindo o número de clusters que devem ser segmentados.

k <- 3

# Definindo a palheta de cores para os clusters a serem plotados.

pallete <- brewer.pal(n = k, name = "Dark2")

# Plotando dendograma.

dend %>%
  set(what = "labels_col", value = pallete, k = k) %>%
  set(what ="branches_k_color", value = pallete, k = k) %>%
  set(what ="branches_lwd", value = 3) %>%
  plot(horiz = F, axes = T, main = 'Dendogram for terms')

# Adicionando um retângulo sobre cada cluster gerado.

rect.dendrogram(dend, k = k, col = rgb(0.1, 0.2, 0.4, 0.1), border = 0, which = 1:k)
```

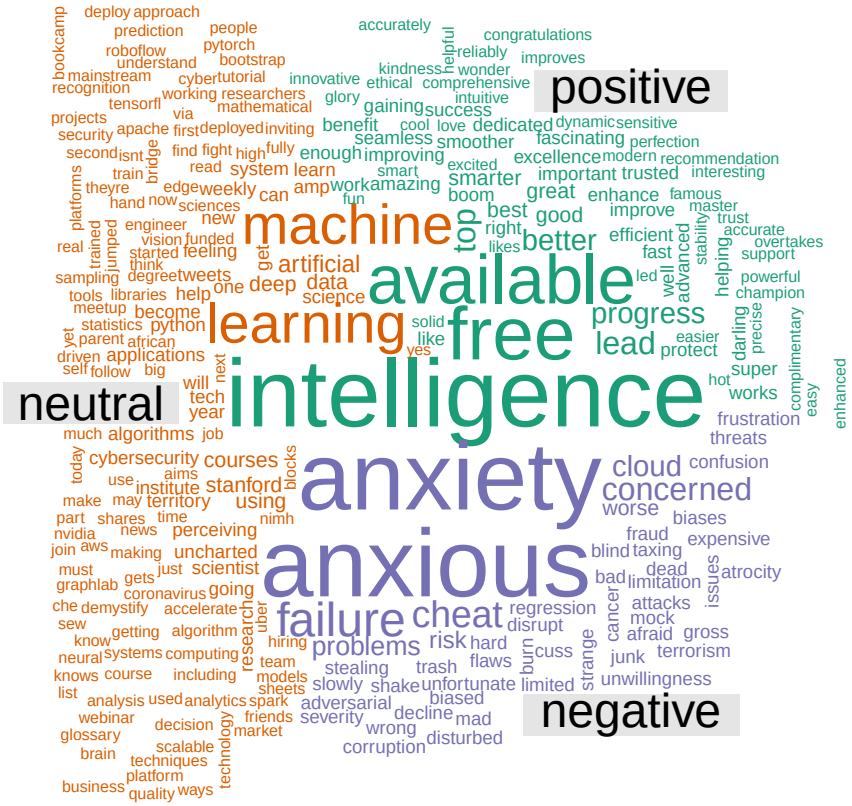


Wordcloud II

A wordcloud a seguir visa exibir as palavras classificadas com conotação positiva, negativa e neutra mais recorrentes nos tweets capturados.

```
# Carregando palavras previamente classificadas como positivas e negativas.

pos <- readLines("positiveWords.txt")
neg <- readLines("negativeWords.txt")

# Limpando conjunto de palavras positivas e negativas.

pos <- cleanData(pos)
neg <- cleanData(neg)

# Computando a polaridade de cada termo do conjunto de dados.

feelingsWords <- feelingsScore(freqWords$word, pos, neg)

feelingsWords$freq <- freqWords$freq

feelingsWords <- feelingsWords %>%
  mutate(
    positive = ifelse(score == 1, freq, 0),
    neutral = ifelse(score == 0, freq, 0),
```

```
    negative = ifelse(score == -1, freq, 0),
  ) %>%
  select(positive, neutral, negative)

rownames(feelingsWords) <- freqWords$word

par(mar = c(0,0,0,0))

comparison.cloud(feelingsWords,
                 scale = c(3,.5),
                 title.size = 1.5)
```



Contato

- **E-mail:** franklinfs390@gmail.com
- **Linkedin:** https://www.linkedin.com/in/franklinfs390/
- **Github:** https://github.com/franklin390