Mini-Projeto 02 - Sentiment Analisys II

Franklin Ferreira

13 de fevereiro, 2020

% !TEX encoding = UTF-8 Unicode

Mini-Projeto 02 - Sentiment Analisys (Análise de sentimentos) II

O objetivo desta análise é explorar diferentes técnicas e ferramentas para a captura, manipulação e transformação de dados provenientes do Twitter. Buscaremos introduzir o processo de comparação dos sentimentos dos usuários desta rede social em relação a dois temas (duas palavras-chave).

Esta técnica visa auxilar os tomadores de decisão na compreensão dos sentimentos do seu público alvo em relação a múltiplos temas. Como por exemplo, determinar se uma campanha de marketing apresentou uma aceitação mais positiva ou negativa do que outra.

O projeto completo, bem como todos os arquivos auxiliares utilizados para sua criação 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)

## Pacotes para se conectar com o Twitter.

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

library(twitteR)
library(httr)

## Pacotes para Data Munging.

# install.packages("stringr")
# install.packages("plyr")
# install.packages("dplyr")
# install.packages("tm")
```

```
library(stringr)
library(plyr)
library(dplyr)
library(tm)

## Pacotes para a criação de gráficos.

# install.packages("lattice")
# install.packages("ggplot2")

library(lattice)
library(ggplot2)
```

Funções auxiliares

Antes de iniciar a análise, vamos definir algumas funções auxiliares para automatizar as tarefas de Data Munging e o cálculo da polaridade do sentimento de um Tweet.

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

# Função 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 usuários "@people".

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

  # Remove pontuação.

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

  # Remove números.

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

  # Remove espaços desnecessários.

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

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

  # Convertendo encoding de caracteres e letras maíusculas em minúsculas.

  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)
}
```

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, poderá 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)
```

Realizando a análise de Sentimento sobre os Tweets.

Para determinar a polaridade do sentimento em um texto, utilizamos dicionários de palavras classificadas previamente como positivas e negativas. Palavras que não existam dentro destes banco de dados são classificadas com polaridade neutra.

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

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

Para manter a padronização entre as palavras contidas nos dicionários e as palavras extraídas dos Tweets, utilizamos a mesma função de limpeza em cada conjunto de dados.

```
# Limpando dicionários de palavras com polaridade positivas e negativas.

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

Para avaliar o resultado do cálculo da polaridade do sentimento de uma determinada sentença, utilizaremos uma massa de dados de teste para analisar os resultados gerados.

Observe que o score gerado pode assumir os seguintes valores:

- **0** - Indica que a expressão não possui palavra em nossas listas de palavras positivas e negativas ou encontrou pares de palavras positivas e negativas na mesma sentença;
- **1** - Indica que a expressão possui uma palavra com conotação positiva e;
- **-1** - Indica que a expressão possui uma palavra com conotação negativa.

```
# Criando massa de dados para teste.

test <- c("Big Data is the future!", "...awesome experience...",
          "analytics could not be bad?", "Learn to use big data!")

# Limpando as senteças da massa de dados de teste.

test <- cleanData(test)

# Computando a polaridade de cada sentença com base nos dicionários de palavras
# positivas e negativas.

feelingsTest <- feelingsScore(test, pos, neg)

# Verificando o tipo do objeto criado.

class(feelingsTest)
```

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

```
## Visualizando os scores gerados.

feelingsTest$score
```

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

O resultado gerado indica que:

- a primeira e a última sentença da massa de dados de teste apresentam polaridade neutra;
- a segunda senteça apresenta polaridade positiva e;
- a terceira sentença apresenta polaridade negativa.

Análisando sentimentos entre diferentes palavras-chave

Nesta fase de análise desejamos capturar um conjunto de tweets que contenham as palavras-chave *Machine Learning* e *Data Science*. Após todas as etapas de limpeza e transformação dos textos, as porcentagens de tweets positivos, negativos e neutros são exibidas.

```
# Determina o número máximo e o idioma dos Tweets a serem capturados.

n <- 3000

lang <- 'en'

# Definindo as Key Words.

keyWord_1 <- "Machine Learning"
keyWord_2 <- "Data Science"

# Capturando tweets em inglês com as diferentes key words especificadas.

mlTweets <- searchTwitter(keyWord_1, n = n, lang = lang)
dsTweets <- searchTwitter(keyWord_2, n = n, lang = lang)

# Extraindo textos dos tweets.

mlText <- sapply(mlTweets, function(x) enc2native(x$getText()))
dsText <- sapply(dsTweets, function(x) enc2native(x$getText()))

# Definindo o número de tweets capturados para cada key word.

nTweets <- c(length(mlText), length(dsText))

# Unindo textos dos tweets.

allTextsTweets <- c(mlText, dsText)

# Limpando textos dos tweets.

allTextsTweets <- cleanData(allTextsTweets)
```

Vamos visualizar os textos dos primeiros tweets capturados.

```
# Exibe o texto dos 5 primeiros tweets capturados.

allTextsTweets[1:5]
```

[1] “please rt looking for a postdoc to join an exciting collaboration between and applying data analytics and...”
[2] “new "smart surface" from mit’s csail uses in excess of antennas to boost signal strength for wireless devices...” [3] “drone blimp free ports powered by machine learning blockchain ai”
[4] “drone blimp free ports powered by machine learning blockchain ai”
[5] “machine learning glossary google developers”

Antes de iniciarmos a análise exploratória dos dados, precisamos remover as palavras que são irrelevantes para o estudo. Estes termos são chamados de **Stop words**.

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

tweetCorpus <- Corpus(VectorSource(allTextsTweets))

# Removendo stopwords dos textos dos tweets.

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

# Convertendo o objeto da classe Corpus para um vetor.

tweets <- unname(unlist(tweetCorpus))

# Eliminando elemento que identifica o idioma das sentenças manipuladas.

allTextsTweets <- tweets[- length(tweets)]

# Exibe o texto dos 5 primeiros tweets capturados.

allTextsTweets[1:5]
```

[1] “please rt looking postdoc join exciting collaboration applying data analytics . . . ”
[2] “new "smart surface" mit’s csail uses excess antennas boost signal strength wireless devices. . . ” [3] “drone blimp free ports powered machine learning blockchain ai”
[4] “drone blimp free ports powered machine learning blockchain ai”
[5] “machine learning glossary google developers”

Compare os resultados antes e após a remoção das *stop words*. Perceberá que palavras como *in*, *for*, *is* e *a* foram removidas das sentenças.

Calculando scores

Vamos calcular o score de cada tweet e definir as proporções de polaridade dos textos.

```
# Aplicando função para calcular a polaridade dos tweets.

scores <- feelingsScore(allTextsTweets, pos, neg)

# Organizando o dataframe com os scores de polaridade calculados.

scores$keyWord <- factor(rep(c(keyWord_1, keyWord_2), nTweets))

# Classificando scores obtidos em polaridades negativas ou positivas.

scores$veryPos <- as.numeric(scores$score >= 1)
scores$veryNeg <- as.numeric(scores$score <= -1)

# Exibindo as primeiras linhas do dataset.

glimpse(scores)
```

```
## Observations: 6,000
## Variables: 5
```

```
## $ text    <fct> please rt looking   postdoc  join  exciting collaboration  ...
## $ score   <int> 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 2, 1, 1, 1, 0, -1, 0, 0, -2, ...
## $ keyWord <fct> Machine Learning, Machine Learning, Machine Learning, Machi...
## $ veryPos <dbl> 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,...
## $ veryNeg <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,...
```

```
# Calculando o total de Tweet positivos, negativos e neutros por palavra chave.

nMlPos <- sum(scores[scores$keyWord == keyWord_1, 'veryPos'])
nMlNeg <- sum(scores[scores$keyWord == keyWord_1, 'veryNeg'])
nMlNet <- length(mlText) - (nMlPos + nMlNeg)

nDsPos <- sum(scores[scores$keyWord == keyWord_2, 'veryPos'])
nDsNeg <- sum(scores[scores$keyWord == keyWord_2, 'veryNeg'])
nDsNet <- length(dsText) - (nDsPos + nDsNeg)

# Calculando a porcentagem de tweets com polaridade negativa, positiva e neutra
# por palavra chave.

mlScorePos <- round( 100 * nMlPos / (nMlPos + nMlNeg + nMlNet))
dsScorePos <- round( 100 * nDsPos / (nDsPos + nDsNeg + nDsNet))

mlScoreNeg <- round( 100 * nMlNeg / (nMlPos + nMlNeg + nMlNet))
dsScoreNeg <- round( 100 * nDsNeg / (nDsPos + nDsNeg + nDsNet))

mlScoreNet <- round( 100 * nMlNet / (nMlPos + nMlNeg + nMlNet))
dsScoreNet <- round( 100 * nDsNet / (nDsPos + nDsNeg + nDsNet))

# Criando um dataframe com os resultados obtidos para facilitar a visualização dos resultados.

data.frame(positve  = c(DataScience = dsScorePos, MachineLearning = mlScorePos),
           negative = c(DataScience = dsScoreNeg, MachineLearning = mlScoreNeg),
           neutral  = c(DataScience = dsScoreNet, MachineLearning = mlScoreNet))
```

```
##                 positve negative neutral
## DataScience          37       12      51
## MachineLearning      36        9      55
```

Os resultados demonstram que a sentenças que contenham a palavra-chave *Machine Learning* apresentam polaridade neutra em 55% dos Tweets. Para a palavra-chave *Data Science*, a proporção é menor tendo 51% dos Tweets com polaridade Neutra.

Vemos que Tweets relacionados a *Data Science* são mais positivos ou negativos do que os relacionados a *Machine Learning*. Ou seja, há um posicionamento mais claro sobre os sentimentos dos indivíduos em Tweets relacionados a *Data Science*.

Vamos calcular as medidas estatísticas de centralidade e dispersão para analisar os scores obtidos para cada palavra-chave.

```
scores %>%
  group_by(keyWord) %>%
  select(score) %>%
  summarise (
    min    = min(score),
```

```
    Q1     = quantile(score, probs = c(0.25)),
    median = median(score),
    mean   = mean(score),
    Q2     = quantile(score, probs = c(0.75)),
    max    = max(score),
    SD     = sd(score)
  )
```

```
## # A tibble: 2 x 8
##   keyWord            min    Q1 median  mean    Q2   max    SD
##   <fct>            <int> <dbl>  <dbl> <dbl> <dbl> <int> <dbl>
## 1 Data Science        -4     0      0 0.313     1     4 0.996
## 2 Machine Learning    -3     0      0 0.327     1     4 0.911
```

O valor da mediana das palavras-chave corrobora com as porcentagens obtidas anteriormente. *Data Science* é a sentença que apresenta o maior desvio padrão. Ou seja, seus scores estão mais dispersos.

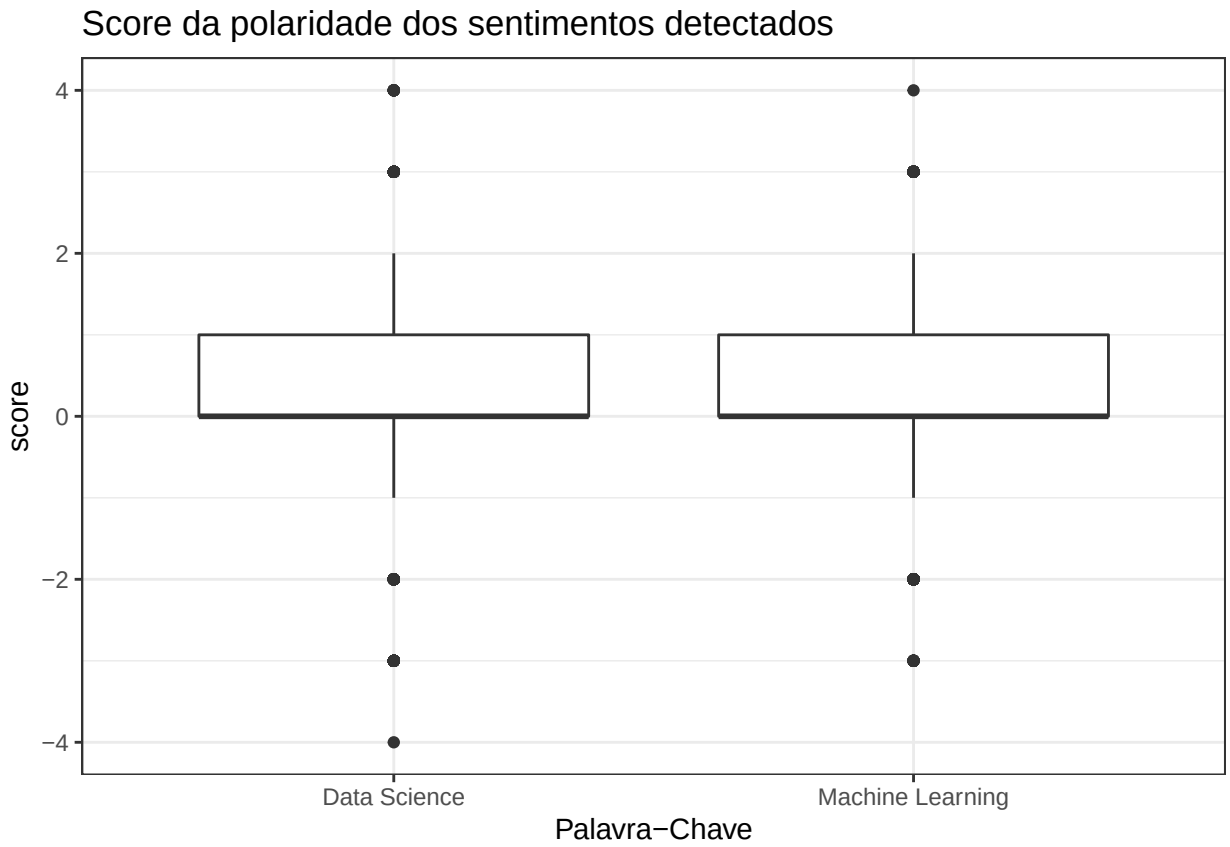
Gráficos

Boxplot

Para facilitar a compreensão das estatísticas dos Tweets, criaremos alguns gráficos que exibam visualmente os resultados anteriormente calculados.

```
# Gerando um boxplot para os scores de polaridade obtidos.

qplot(y = score, x = keyWord, data = scores, geom = 'boxplot') +
  xlab('Palavra-Chave') +
  labs(title = 'Score da polaridade dos sentimentos detectados') +
  theme_bw()
```

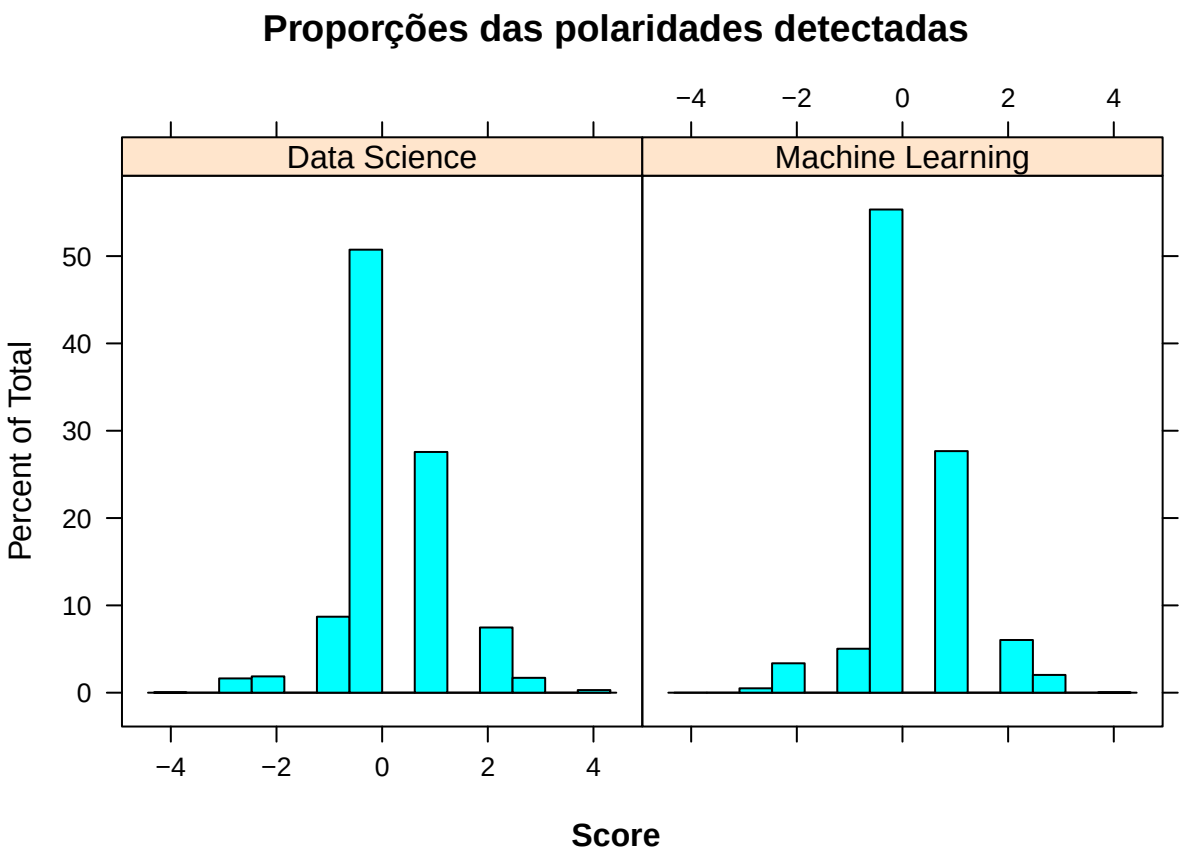


Note que a palavra-chave *Data Science* apresenta outliers com valores de score mais extremos do que *Machine Learning*.

Histograma

```
# Gerando um histograma com o lattice para os scores de polaridade obtidos.

histogram(data = scores, ~score|keyWord,
          main = "Proporções das polaridades detectadas",
          xlab = "", sub = "Score")
```



Note que as proporções de sentenças com polaridade neutra são maiores para a palavra-chave *Machine Learning*.

Contato

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