Análise de Sentimentos - Twitter

Renato Jorge Dyszy

13/08/2020

Mini-Projeto - Análise de Sentimentos - Twitter

O objetivo desse trabalho é capturar dados da rede social Twitter e realizar análise de sentimentos com os dados capturados. Será usado um classificador com o algoritmo Naive Bayes.

Todo o projeto será descrito por etapas.

Etapa 1 - Pacotes e Autenticação

```
# Instalando e carregando o pacote twitter
#install.packages("twitteR")
#install.packages("httr")
#install.packages("knitr")
#install.packages("rmarkdown")
library(twitteR)
library(httr)
library(knitr)
library(rmarkdown)

# Carregando a biblioteca com funções de limpeza
source('utils.R')


# Chaves de autenticação no twitter
Key <- "4JXZrkiEnK4uG2XnuRVP5DpFx"
secret <- "QPaFjQ254F8IY3CJgnuJciyLraCl4aK5gezdcaciaDwrHIenE8"
token <- "1260033538746757128-SAhJ3lV2AKvjW26flqQq28MVq1Et3B"
tokenSecret <- "zGMgOFyf1jaSakHZbEAdIBILb8x0ugz5TbUHD5LmKrmK3"

# Autenticação. Responda 1 quando perguntado sobre utilizar direct connection
setup_twitter_oauth(Key, secret, token, tokenSecret)
```

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

Etapa 2 - Captura de tweets

Buscaremos tweets com referência a hashtag #MachineLearning

```
# Capturando os tweets
tema <- "MachineLearning"
qtd_tweets <- 1500
lingua <- "pt"
tweetdata <- searchTwitter(tema, n = qtd_tweets, lang = lingua)
```

```
## Warning in doRppAPICall("search/tweets", n, params = params, retryOnRateLimit =
## retryOnRateLimit, : 1500 tweets were requested but the API can only return 215
```

```
head(tweetdata)
```

```
## [[1]]
## [1] "Fabriciosx: RT @tmrolemberg: Vale a pena dar uma lida, GPT-3 é o modelo de IA que esta sendo bas
##
## [[2]]
## [1] "datasciencebot_: RT @tmrolemberg: Vale a pena dar uma lida, GPT-3 é o modelo de IA que esta send
##
## [[3]]
## [1] "tmrolemberg: Vale a pena dar uma lida, GPT-3 é o modelo de IA que esta sendo bastante comentado
##
## [[4]]
## [1] "prodest_iti: Você já deve ter ouvido falar de machine learning. Mas será que essa tecnologia fu
##
## [[5]]
## [1] "TheCuriousLuke: RT @GrupoTreinar: Novidades sobre Inteligência Artificial e Gestão da Informaçã
##
## [[6]]
## [1] "WebSecurityIT: RT @GrupoTreinar: Novidades sobre Inteligência Artificial e Gestão da Informação
```

Etapa 3 - Tratamento do dados

```
# Obtendo o texto
tweetlist = sapply(tweetdata, function(x) x$getText())

# Limpando, organizando e transformando os dados
tweetlist <- limpaTweets(tweetlist)

# Removendo os NAs
tweetlist = tweetlist[!is.na(tweetlist)]
names(tweetlist) = NULL
```

Etapa 4 - Wordcloud, associação entre palavras e dendograma

Criação de uma nuvem de palavras para verificar a relação entre as palavras que ocorrem com mais frequencia.

Criamos uma tabela com a frequencia das palavras e então geramos um dendograma, que mostra como as palavras se relacionam e se associam ao tema principal

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

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

```
##
## Attaching package: 'NLP'
```

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

```r
# Com o pacote tm, vamos converter os tweets coletados em um objeto do tipo Corpus, que armazena dado
tweetCorpus <- Corpus(VectorSource(tweetlist))

# Limpa Corpus
tweetCorpus <- limpaCorpus(tweetCorpus)
```

```
## Warning in tm_map.SimpleCorpus(myCorpus, tolower): transformation drops
## documents
```

```
## Warning in tm_map.SimpleCorpus(myCorpus, removePunctuation): transformation
## drops documents
```

```
## Warning in tm_map.SimpleCorpus(myCorpus, removeNumbers): transformation drops
## documents
```

```r
# gerando uma nuvem de palavras
pal2 <- brewer.pal(8, "Dark2")

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

```
## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : inteligencia could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : exploratory could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : conhece could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : livelabs could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : dados could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : artigo could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : design could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : cores could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : tensor could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : modelo could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : comecando could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : esse could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : pensando could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : produzimos could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : scraping could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : trading could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : web could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : projeto could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : por could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : aec could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : learn could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : machines could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : feito could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : mohammed could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : mohsin could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(tweetCorpus, min.freq = 2, scale = c(5, 1), random.color =
## F, : renato could not be fit on page. It will not be plotted.
```


```
  # Convertendo o objeto texto para o formato de matriz
  tweettdm <- TermDocumentMatrix(tweetCorpus)
  tweettdm
```

```
## <<TermDocumentMatrix (terms: 371, documents: 198)>>
## Non-/sparse entries: 1304/72154
```

```
## Sparsity           : 98%
## Maximal term length: 15
## Weighting          : term frequency (tf)
```

```
  # Encontrando as palavras que aparecem com mais frequencia
  findFreqTerms(tweettdm, lowfreq = 11)
```

```
##  [1] "que"          "learning"     "machine"      "voce"         "artificial"
##  [6] "gestao"       "informacao"   "inteligencia" "novidades"    "sobre"
## [11] "com"          "analysis"     "data"         "eda"          "exploratory"
## [16] "resources"    "novo"         "for"          "quantum"
```

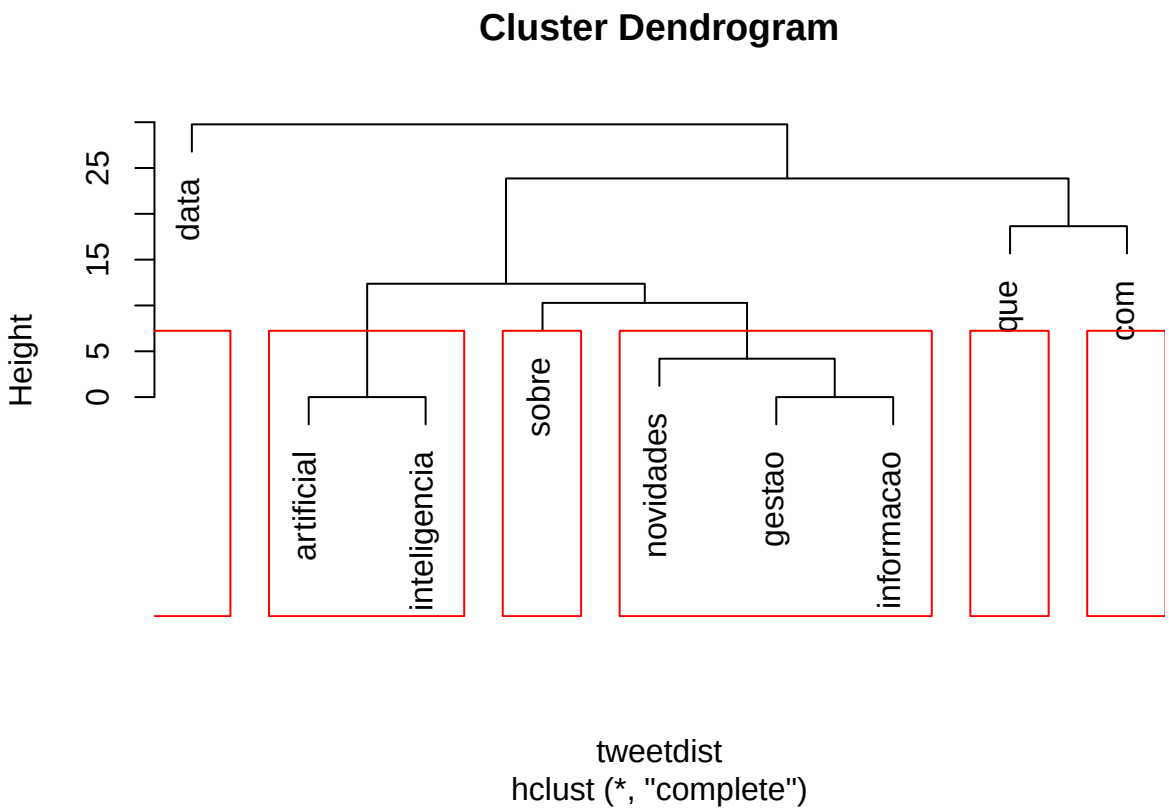
```
  # Buscando associações
  findAssocs(tweettdm, 'inteligencia', 0.6)
```

```
## $inteligencia
## artificial     gestao informacao      sobre  novidades
##       1.00       0.76       0.76       0.76       0.73
```

```
  # Removendo termos esparsos (não utilizados frequentemente)
  tweet2tdm <- removeSparseTerms(tweettdm, sparse = 0.9)

  # Criando escala nos dados
  tweet2tdmscale <- scale(tweet2tdm)

  # Distance Matrix
  tweetdist <- dist(tweet2tdmscale, method = "euclidean")

  # Preparando o dendograma
  tweetfit <- hclust(tweetdist)

  # Criando o dendograma (verificando como as palavras se agrupam)
  plot(tweetfit)

  # Verificando os grupos
  cutree(tweetfit, k = 6)
```

```
##          que   artificial       gestao   informacao inteligencia    novidades
##            1            2            3            3            2            3
##        sobre          com         data
##            4            5            6
```

```
  # Visualizando os grupos de palavras no dendograma
  rect.hclust(tweetfit, k = 6, border = "red")
```



Etapa 5 - Classificador Naive Bayes

Utilizamos as funções classify_emotion() e classify_polarity() do pacote sentiment, que utilizam o algotimo Naive Bayes para a análise de sentimento. Neste caso, o próprio algoritmo faz a classificação das palavras e não precisamos criar listas de palavras positivas e negativas.

```
#install.packages("C:/DataScience/Projetos/Local/AnaliseSentimentos/Rstem_0.4-1.tar.gz", repos = NULL
#install.packages("C:/DataScience/Projetos/Local/AnaliseSentimentos/sentiment_0.2.tar.gz", repos = NU
#install.packages("ggplot2")
library(Rstem)
library(sentiment)
library(ggplot2)
```

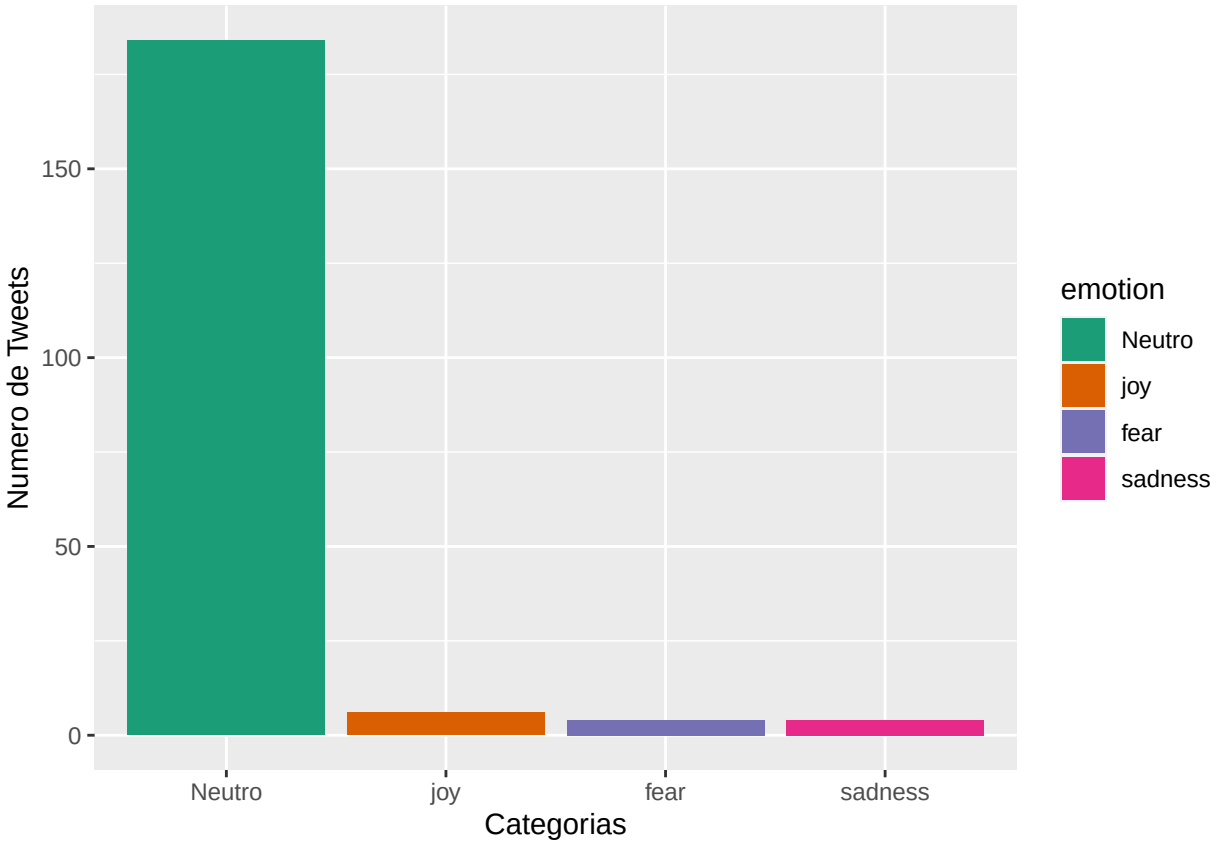
```
##
## Attaching package: 'ggplot2'

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

```
# Classificando emoção
class_emo = classify_emotion(tweetlist, algorithm = "bayes", prior = 1.0)
emotion = class_emo[,7]
```

```
# Substituindo NA's por "Neutro"
emotion[is.na(emotion)] = "Neutro"

# Classificando polaridade
class_pol = classify_polarity(tweetlist, algorithm = "bayes")
polarity = class_pol[,4]

# Gerando um dataframe com o resultado
sent_df = data.frame(text = tweetlist, emotion = emotion,
                     polarity = polarity, stringsAsFactors = FALSE)

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

Etapa 6 - Visualização

```
# Emoções encontradas
ggplot(sent_df, aes(x = emotion)) +
       geom_bar(aes(y = ..count.., fill = emotion)) +
       scale_fill_brewer(palette = "Dark2") +
       labs(x = "Categorias", y = "Numero de Tweets")
```



```r
# Polaridade
ggplot(sent_df, aes(x = polarity)) +
       geom_bar(aes(y = ..count.., fill = polarity)) +
       scale_fill_brewer(palette = "RdGy") +
       labs(x = "Categorias de Sentimento", y = "Numero de Tweets")
```