Análise de Sentimentos

Lucas Braga

06/03/2022

Esse é um pequeno projeto sobre análise de sentimentos usando a linguagem R. A proposta é coletar 200 tweets que contenham a palavra "russia" e analisar se eles contém palavras positivas ou negativas. Em seguida veremos um classificador com o algoritmo Naive Bayes

```
library(rtweet)
library(stringr)
library(ggplot2)
library(tm)
```

```
## Carregando pacotes exigidos: NLP
```

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

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(SnowballC)
library(stringi)
library(RColorBrewer)
library(wordcloud)
library(stringr)
library(plyr)
suppressMessages(library(dplyr))
library(lattice)
library(Rstem)
```

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

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

```
library(SnowballC)
library(sentiment)
```

Coletando os Tweets

```
tweet_df <-
  search_tweets("russia",
                n = 200,
                lang = "en",
                include_rts = FALSE)
names(tweet_df)
```

```
##  [1] "user_id"                "status_id"
##  [3] "created_at"             "screen_name"
##  [5] "text"                   "source"
##  [7] "display_text_width"     "reply_to_status_id"
##  [9] "reply_to_user_id"       "reply_to_screen_name"
## [11] "is_quote"               "is_retweet"
## [13] "favorite_count"         "retweet_count"
## [15] "quote_count"            "reply_count"
## [17] "hashtags"               "symbols"
## [19] "urls_url"               "urls_t.co"
## [21] "urls_expanded_url"      "media_url"
## [23] "media_t.co"             "media_expanded_url"
## [25] "media_type"             "ext_media_url"
## [27] "ext_media_t.co"         "ext_media_expanded_url"
## [29] "ext_media_type"         "mentions_user_id"
## [31] "mentions_screen_name"   "lang"
## [33] "quoted_status_id"       "quoted_text"
## [35] "quoted_created_at"      "quoted_source"
## [37] "quoted_favorite_count"  "quoted_retweet_count"
## [39] "quoted_user_id"         "quoted_screen_name"
## [41] "quoted_name"            "quoted_followers_count"
## [43] "quoted_friends_count"   "quoted_statuses_count"
## [45] "quoted_location"        "quoted_description"
## [47] "quoted_verified"        "retweet_status_id"
## [49] "retweet_text"           "retweet_created_at"
## [51] "retweet_source"         "retweet_favorite_count"
## [53] "retweet_retweet_count"  "retweet_user_id"
## [55] "retweet_screen_name"    "retweet_name"
## [57] "retweet_followers_count" "retweet_friends_count"
## [59] "retweet_statuses_count" "retweet_location"
## [61] "retweet_description"    "retweet_verified"
## [63] "place_url"              "place_name"
## [65] "place_full_name"        "place_type"
## [67] "country"                "country_code"
## [69] "geo_coords"             "coords_coords"
## [71] "bbox_coords"            "status_url"
## [73] "name"                   "location"
## [75] "description"            "url"
## [77] "protected"              "followers_count"
## [79] "friends_count"          "listed_count"
## [81] "statuses_count"         "favourites_count"
## [83] "account_created_at"     "verified"
## [85] "profile_url"            "profile_expanded_url"
## [87] "account_lang"           "profile_banner_url"
```

```
## [89] "profile_background_url"  "profile_image_url"
```

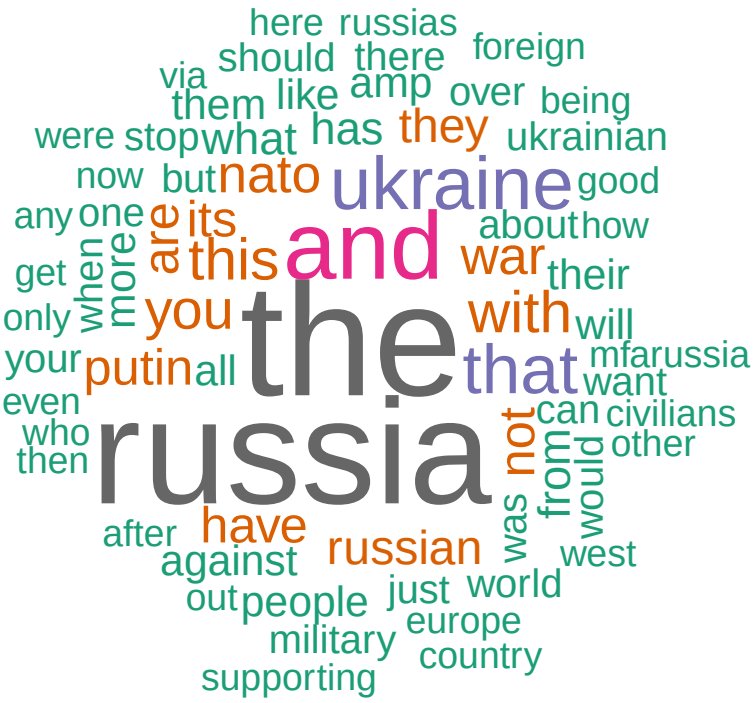
```
tweets <- tweet_df %>%
  select(
    user_id,
    status_id,
    created_at,
    screen_name,
    text,
    favorite_count,
    retweet_count,
    urls_expanded_url
  )
```

Tratando os dados coletados

```
library(tm)
library(SnowballC)
library(stringi)


tweetcorpus <- stri_trans_tolower(tweets$text)
tweetcorpus <- VCorpus(VectorSource(tweetcorpus))
tweetcorpus <- tm_map(tweetcorpus, removePunctuation)
tweetcorpus <-
  tm_map(tweetcorpus, removeWords, stopwords("portuguese"))
```

Gerando nuvem de palavras

```
library(RColorBrewer)
library(wordcloud)

wordcloud(
  tweetcorpus,
  min.freq = 4,
  scale = c(5, 1),
  random.color = F,
  max.word = 70,
  random.order = F,
  colors = brewer.pal(8, "Dark2")
)
```



Algmas visualizações

```
tweetdm <- TermDocumentMatrix(tweetcorpus)
findFreqTerms(tweetdm, lowfreq = 11)
```

```
##  [1] "about"     "against"   "all"       "amp"       "and"       "are"
##  [7] "but"       "can"       "from"      "has"       "have"      "its"
## [13] "just"      "like"      "military"  "more"      "nato"      "not"
## [19] "one"       "over"      "people"    "putin"     "russia"    "russian"
## [25] "should"    "stop"      "that"      "the"       "their"     "them"
## [31] "there"     "they"      "this"      "ukraine"   "ukrainian" "want"
## [37] "war"       "was"       "what"      "when"      "will"      "with"
## [43] "world"     "would"     "you"       "your"
```

```
# Buscando associações
findAssocs(tweetdm, 'russia', 0.60)
```

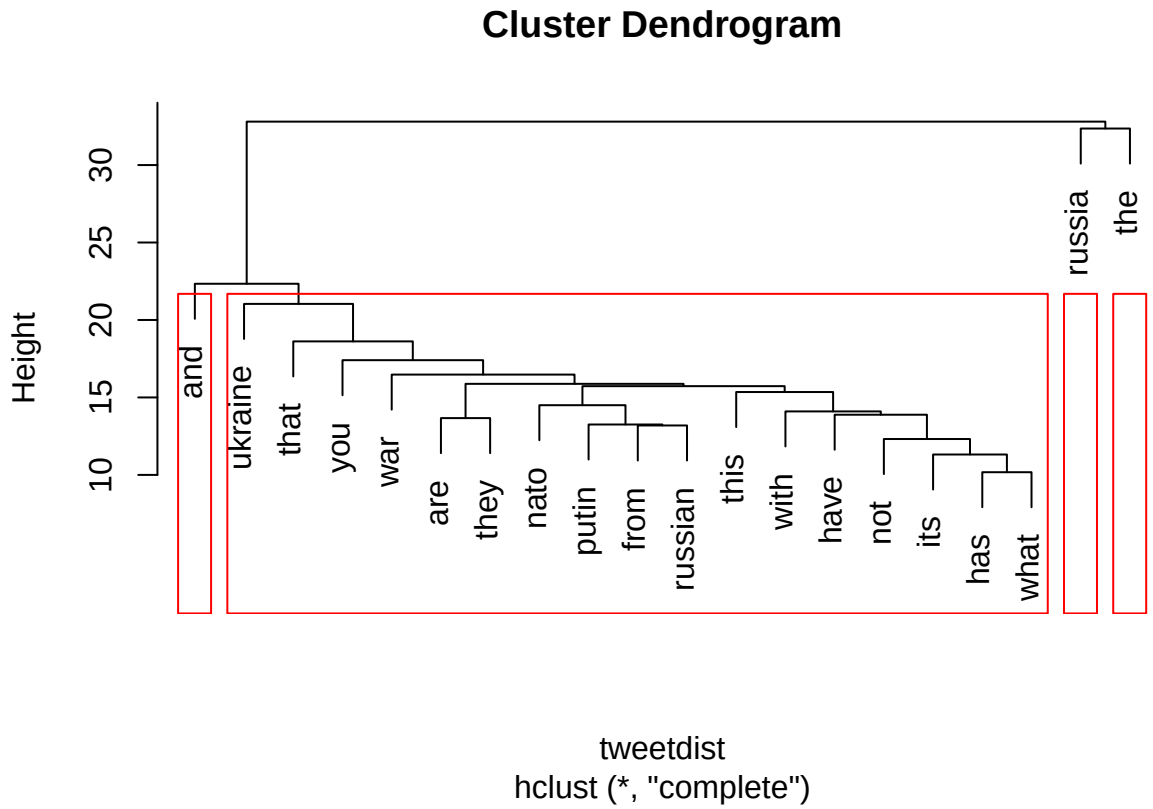
```
## $russia
## numeric(0)
```

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

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


# Matriz de distância
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 = 5)
```

```
##     and     are    from     has    have     its    nato     not   putin  russia
##       1       2       2       2       2       2       2       2       2       3
## russian    that     the    they    this ukraine     war    what    with     you
##       2       2       4       2       2       5       2       2       2       2
```

```r
# Visualizando os grupos de palavras no dendograma
rect.hclust(tweetfit, k = 4, border = "red")
```



Análise de Sentimento

```
# Criando uma função para avaliar o sentimento
sentimento.score = function(sentences,
                            pos.words,
                            neg.words,
                            .progress = 'none')
{
  # Criando um array de scores com 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
                     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)
}

# Mapeando as palavras positivas e negativas
pos = readLines("palavras_positivas.txt")
neg = readLines("palavras_negativas.txt")

# Criando uma massa de dados para teste
teste = c("Russia is the future",
          "Ukraine is awesome",
          "War could not be bad",
          "learn about war")

# Testando a função em nossa massa de dados dummy
```

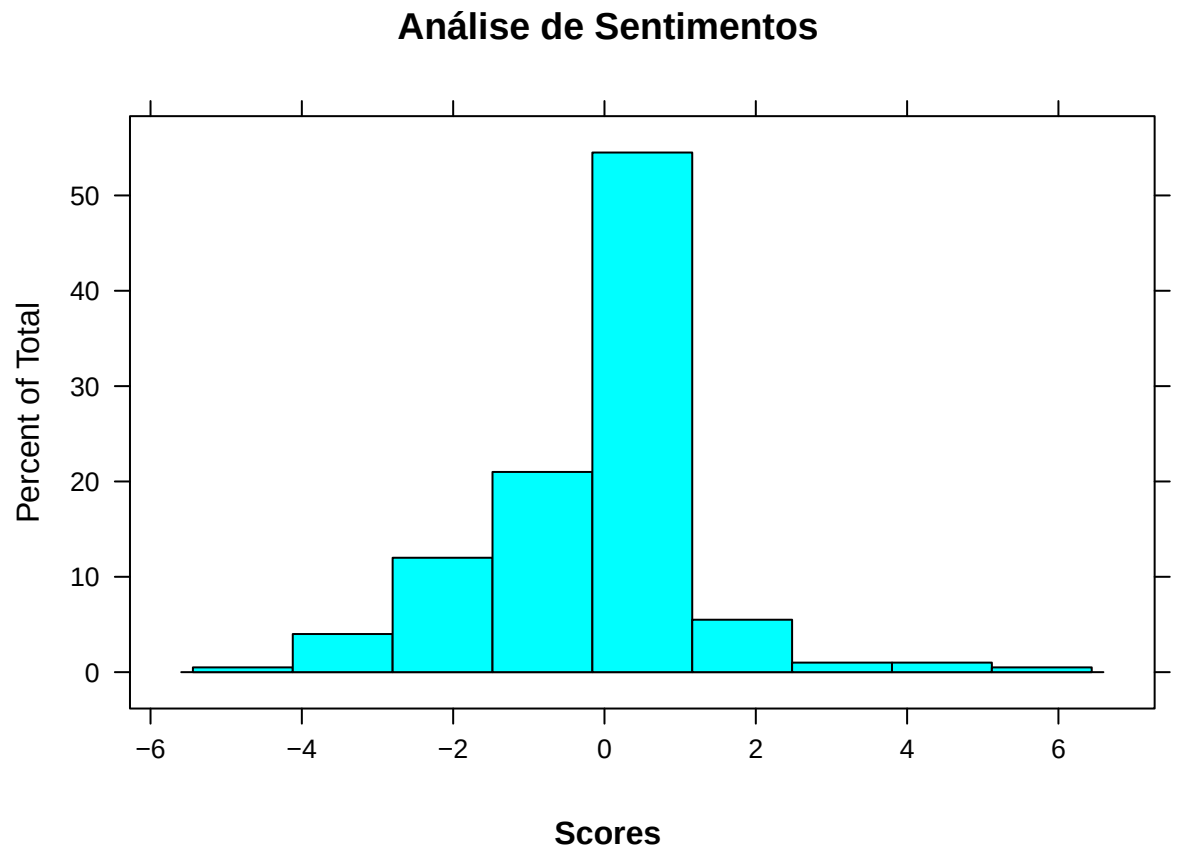
```
testeSentimento = sentimento.score(teste, pos, neg)
scores = sentimento.score(tweets$text, pos, neg, .progress = 'text')
```

```
##   |                                                                      |
```

```
scores$muito.pos = as.numeric(scores$score >= 1)
scores$muito.neg = as.numeric(scores$score <= -1)

# Calculando total
numpos = sum(scores$muito.pos)
numneg = sum(scores$muito.neg)

totalScore = round(100 * numpos / (numpos + numneg))
```

Histograma

```
histogram(
  data = scores,
  ~ scores$score,
  main = "Análise de Sentimentos",
  xlab = "",
  sub = "Scores"
)
```



Usando o classificador Naive Bayes

```
# Classificando emoção
class_emo = classify_emotion(tweets$text, algorithm = "bayes", prior = 1.0)
```

```
## Warning in TermDocumentMatrix.SimpleCorpus(x, control): custom functions are
## ignored
```

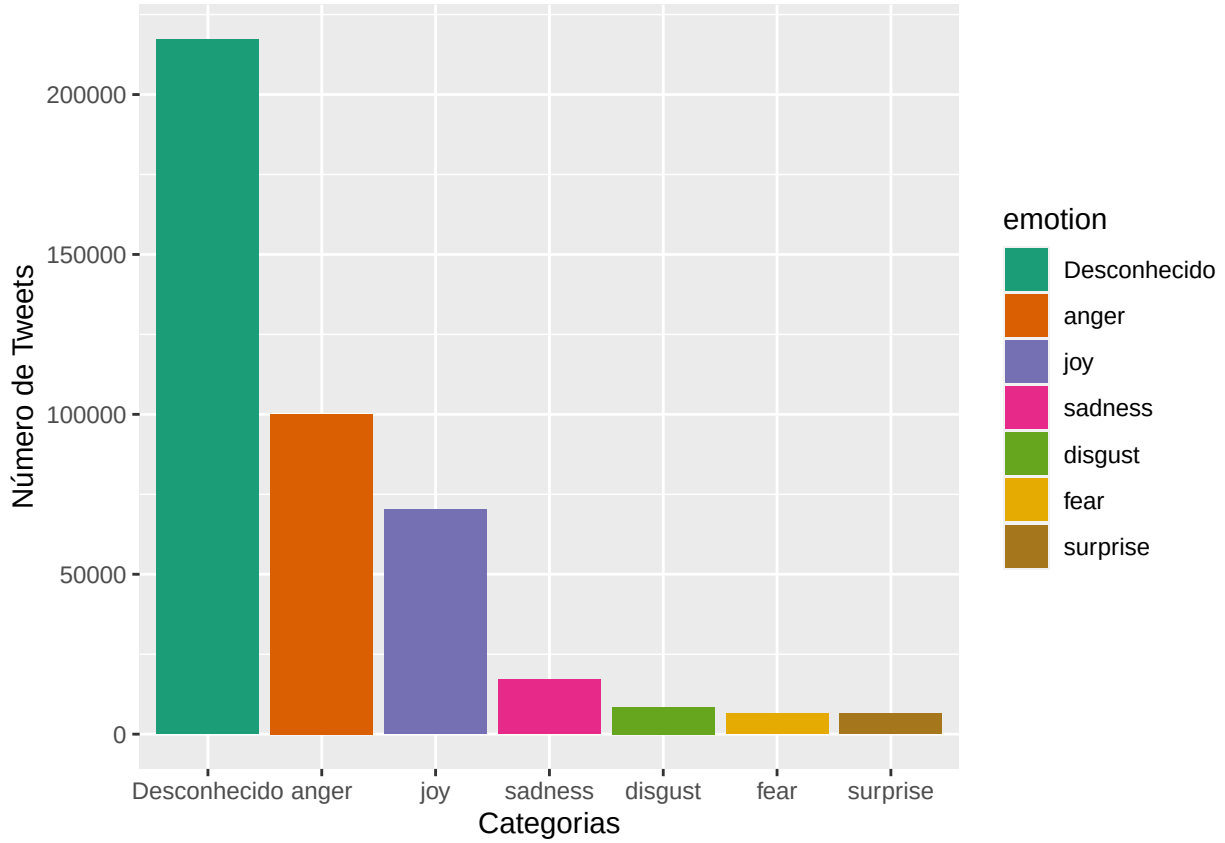
```
emotion = class_emo[, 7]

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

# Classificando polaridade
class_pol = classify_polarity(tweets$text, algorithm = "bayes")
```

```
## Warning in TermDocumentMatrix.SimpleCorpus(x, control): custom functions are
## ignored
```

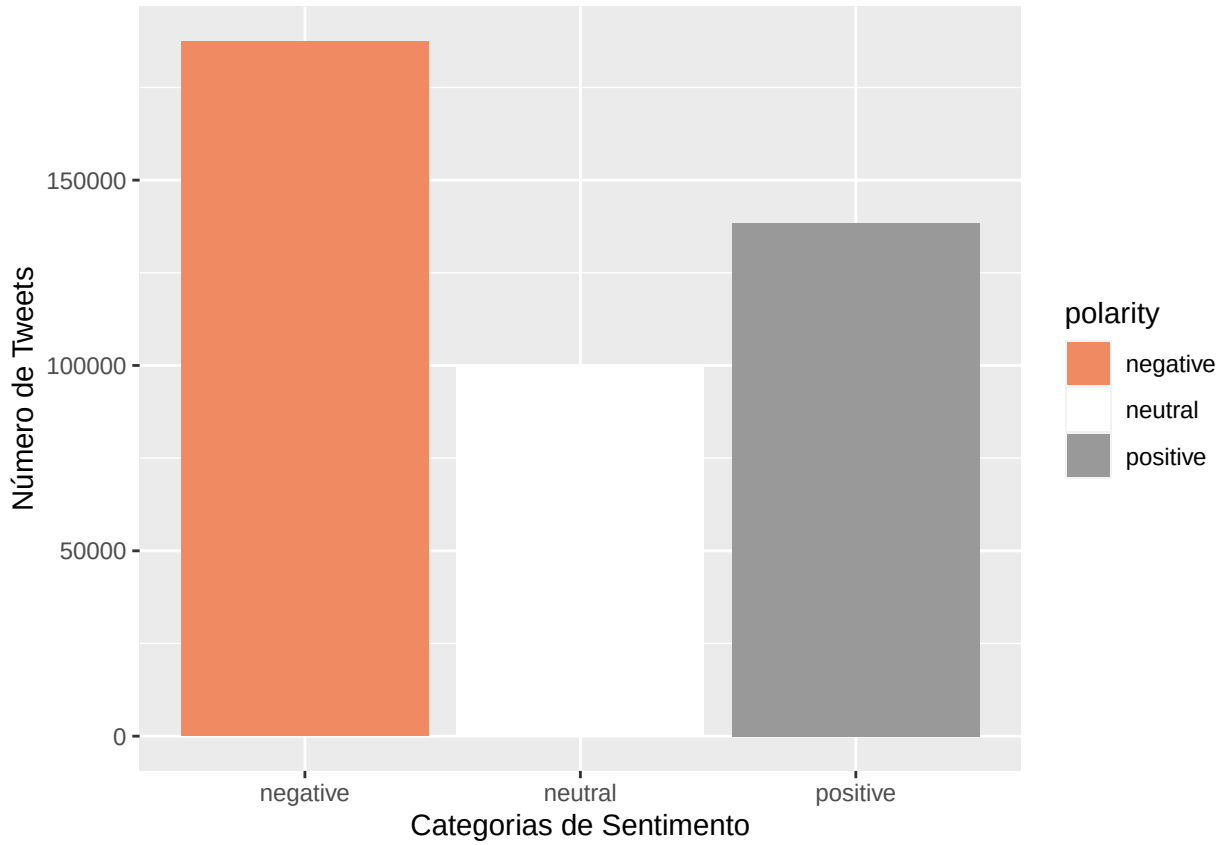
```
polarity = class_pol[, 4]

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

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

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 = "Número de Tweets")
```




```
# 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 = "Número de Tweets")
```


150000
100000
50000
0
Número de Tweets
negative
neutral
positive
Categorias de Sentimento
polarity
negative
neutral
positive