Mini-Projeto 03 - Sentiment Analisys III

Franklin Ferreira

14 de fevereiro, 2020

% !TEX encoding = UTF-8 Unicode

Mini-Projeto 03 - Sentiment Analisys (Análise de sentimentos) III

O objetivo desta análise é explorar diferentes técnicas e ferramentas para a captura, manipulação e transformação de dados provenientes do Twitter. Buscaremos classificar os sentimentos que cada Tweet transmite e determinar sua polaridade.

Esta técnica visa auxilar os tomadores de decisão na compreensão dos sentimentos do seu público alvo em relação a um determinado tema. Como por exemplo, determinar se uma campanha de marketing gerou surpresa, raiva, medo, desgosto, alegria, etc.

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

Os pacotes *Rstem* e *sentiment*, podem ser econtrados no diretório Archive do CRAN: https://cran.r-project.org/

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

```
library(plyr)
library(dplyr)

## Pacotes para a criação de gráficos.

# install.packages("ggplot2")

library(ggplot2)

# Pacotes para executar a análise de sentimentos.

# install.packages("Rstem_0.4-1.tar.gz", sep = "", repos = NULL, type = "source")
# install.packages("sentiment_0.2.tar.gz",sep = "", repos = NULL, type = "source")

library(Rstem)
library(sentiment)
```

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)
```

Capturando Tweets

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

n <- 1500
```

```r
lang <- 'en'

# Definindo a Key Word.

keyWord <- "Big Data"

# Capturando tweets que contenham a palavra chave especificada.

tweets <- searchTwitter(keyWord, n = n, lang = lang)

# Extraindo os textos dos tweets.

textTweets <- sapply(tweets, function(tweet) { enc2native(tweet$getText()) })

# Limpando textos dos tweets.

textTweets <- cleanData(textTweets)

# Visualizando os textos dos 5 primeiros tweets capturados.

textTweets[1:5]
```

[1] “new book alert my team contributed a chapter titled when big data meet supportive communication advancing inte...”
[2] “eye tracking facial recognition sensors and digital displays gather data that retailers can use to fine tune the...”
[3] “the dikw pyramid is an often used method with roots in knowledge management to explain the ways we move from data d to in...” [4] “these ai and big data companies are entering dcode’s accelerator program”
[5] “difference between big data and big lore”

Classificando as emoções dos Tweets

O pacote *sentiment* fornece a função *classfy_emotion()* que permite utilizar um classificador treinado com o algoritmo Naive Bayes para identificar 6 tipos de sentimentos em um texto:

- *anger* (raiva);
- *disgust* (nojo);
- *fear* (medo);
- *joy* (alegria);
- *sadness* (tristeza) e;
- *surprise* (surpresa).

As sentenças que não puderem ser classificadas pelo algoritmo, serão rotuladas como tendo um sentimento neutro.

```r
# Classficando as emoções identificadas nos textos.

class_emo <- classify_emotion(textTweets, algorithm = "bayes")

# Extraindo emoções identificadas.
```

```r
emotion <- class_emo[, 'BEST_FIT']

# Substituindo valores NA por "Neutro".

emotion[is.na(emotion)] = "neutral"

sort(table(emotion), decreasing = T)
```

```
## emotion
##  neutral      joy  sadness surprise    anger  disgust     fear
##     2248      444       84       78       71       52       23
```

Note que a maior parte dos Tweets são classificados como tendo um sentimento neutro. O que nos indica que a maior parte dos sentimentos referentes a palavra-chave *Big Data* é diferente daqueles que o algoritmo utilizado é capaz de classificar.

Determinando a polaridade dos Tweets

O pacote *sentiment* também fornece a função *classify_polarity()* para determinar a polaridade de uma sentença a partir da utilização de um clasificador treinado com o algortimo Naive Bayes.

```r
# Classificando a polaridade de cada um dos textos dos tweets.

class_pol <- classify_polarity(textTweets, algorithm = "bayes")

polarity <- class_pol[, 'BEST_FIT']

# Gerando um dataframe com os resultados das classificações.

tweetsFeelings = data.frame(text = textTweets, emotion = emotion,
                            polarity = polarity, stringsAsFactors = FALSE)

# Ordenando o dataframe segundo o tipo de emoção identificada.

tweetsFeelings = within(tweetsFeelings,
                emotion <- factor(emotion,
                                  levels = names(sort(table(emotion),
                                                      decreasing = TRUE)))
                )
```

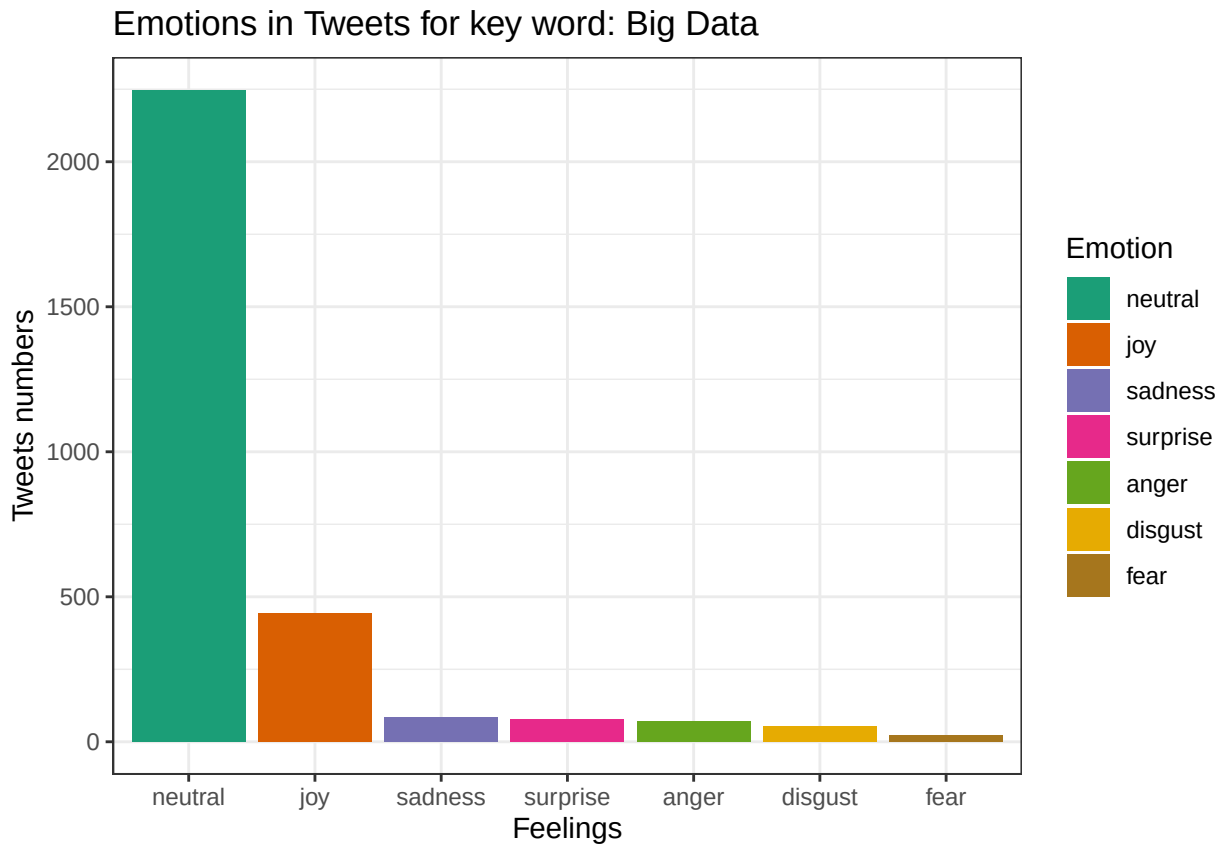
Gráficos

Gráfico de barras para as emoções identificadas nos Tweets

```r
# Plotando um gráfico de barras para as emoções encontradas.

tweetsFeelings %>%
  ggplot(aes(x = emotion, fill = emotion)) +
```

```
  geom_bar() +
  scale_fill_brewer(palette = "Dark2") +
  xlab("Feelings") +
  ylab("Tweets numbers") +
  labs(title = paste('Emotions in Tweets for key word:', keyWord), fill = 'Emotion') +
  theme_bw()
```

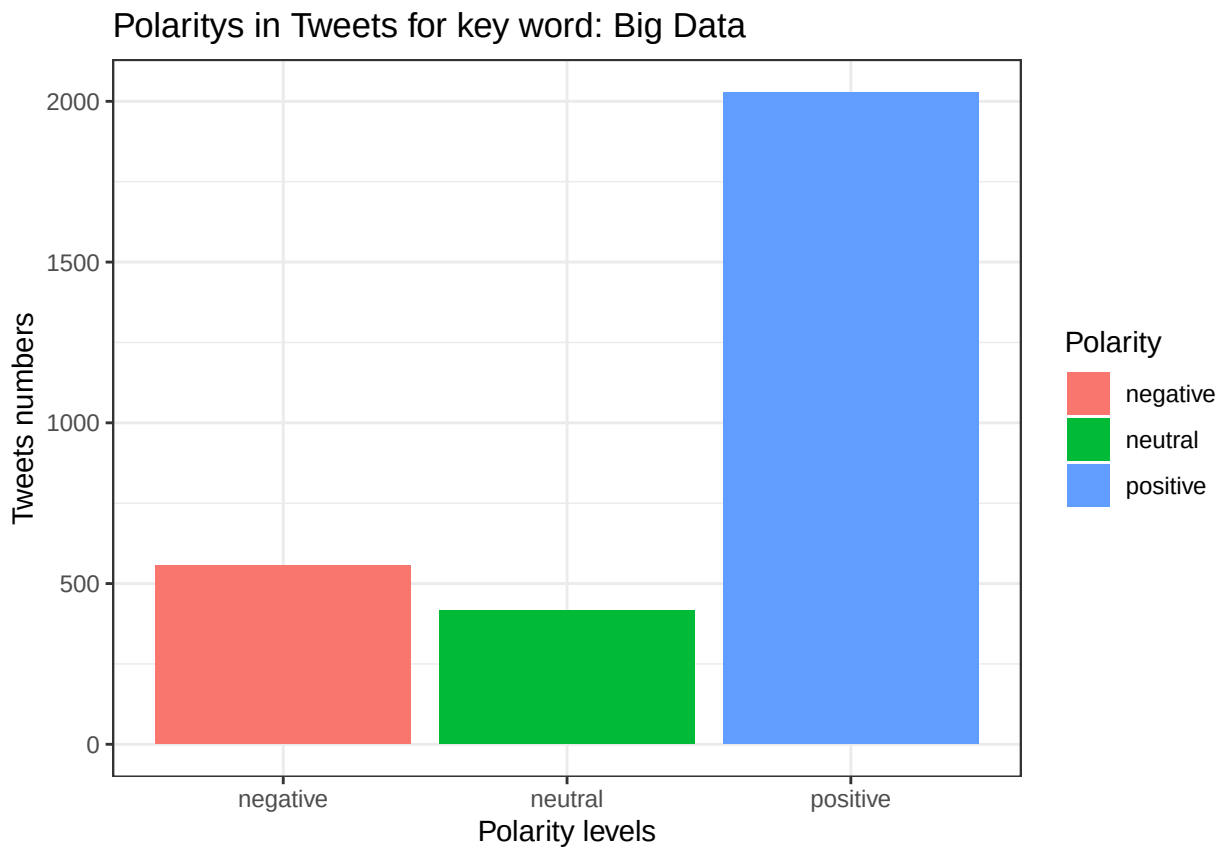


Observe que após o sentimento neutro (neutral), a alegria (joy), a tristeza (sadness) e a surpresa (surprise) são os mais recorrentes.

Gráfico de barras para as polaridades identificadas nos Tweets

```
# Plotando um gráfico de barras para a polaridade dos Tweets.

tweetsFeelings %>%
  ggplot(aes(x = polarity, fill = polarity)) +
  geom_bar() +
  xlab("Polarity levels") +
  ylab("Tweets numbers") +
  labs(title = paste('Polaritys in Tweets for key word:', keyWord), fill = 'Polarity') +
  theme_bw()
```



Podemos observar a partir do gráfico que a maior parte dos Twees tem polaridade positiva. Ou seja, apresenta um número maior de palavras com conotação positiva do que palavras com conotação negativa.

Contato

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