Elaborado com o Notebook Python Google Colab

```
# Load the Drive helper and mount
from google.colab import drive

# This will prompt for authorization.
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
!ls "/content/drive/My Drive/Databases"
```

```
'Arvore Piratininga 2017.xlsx'   NoThemeTweets.csv
 blogset-br.csv.gz               'Sentiment Analysis Dataset.csv'
 imdb-reviews-pt-br.csv
```

```
import pandas as pd

dados = pd.read_csv("/content/drive/My Drive/Databases/NoThemeTweets.csv")
```

```
dados
```

| | id | tweet_text | tweet_date | sentiment | query_used |
|---|---|---|---|---|---|
| **0** | 1031761728445530112 | @Tixaa23 14 para eu ir :) | Tue Aug 21 04:35:39 +0000 2018 | Positivo | :) |
| **1** | 1031761040462278656 | @drexalvarez O meu like eu já dei na época :) | Tue Aug 21 04:32:55 +0000 2018 | Positivo | :) |
| **2** | 1031760962372689920 | Eu só queria conseguir comer alguma coisa pra ... | Tue Aug 21 04:32:37 +0000 2018 | Positivo | :) |
| **3** | 1031760948250456066 | :D que lindo dia ! | Tue Aug 21 04:32:33 +0000 2018 | Positivo | :) |
| **4** | 1031760895985246208 | @Primo_Resmungao Pq da pr jeito!!é uma "oferta... | Tue Aug 21 04:32:21 +0000 2018 | Positivo | :) |
| **...** | ... | ... | ... | ... | ... |
| | | Acordar 8 horas é tão | Fri Oct 12 | | |

```
classificacao = dados["sentiment"].replace(["Negativo","Positivo"],[0,1])
```

```
dados["classificacao"] = classificacao
```

```
dados.groupby('classificacao').count()
```

| classificacao | id | tweet_text | tweet_date | sentiment | query_used |
|---|---|---|---|---|---|
| 0 | 522707 | 522707 | 522707 | 522707 | 522707 |
| 1 | 263107 | 263107 | 263107 | 263107 | 263107 |

```
dados
```

| | id | tweet_text | tweet_date | sentiment | query_used | c |
|---|---|---|---|---|---|---|
| 0 | 1031761728445530112 | @Tixaa23 14 para eu ir :) | Tue Aug 21 04:35:39 +0000 2018 | Positivo | :) | |
| 1 | 1031761040462278656 | @drexalvarez O meu like eu já dei na época :) | Tue Aug 21 04:32:55 +0000 2018 | Positivo | :) | |
| 2 | 1031760962372689920 | Eu só queria conseguir comer alguma coisa pra ... | Tue Aug 21 04:32:37 +0000 2018 | Positivo | :) | |
| 3 | 1031760948250456066 | :D que lindo dia ! | Tue Aug 21 04:32:33 +0000 2018 | Positivo | :) | |
| 4 | 1031760895985246208 | @Primo_Resmungao Pq da pr jeito!!é uma "oferta... | Tue Aug 21 04:32:21 +0000 2018 | Positivo | :) | |
| ... | ... | ... | ... | ... | ... | ... |
| | | Acordar 8 horas é tão | Fri Oct 12 | | | |

```
import nltk
from nltk import tokenize

nltk.download('stopwords')

token_espaco = tokenize.WhitespaceTokenizer()

token_pontuacao = tokenize.WordPunctTokenizer()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
palavras_irrelevantes = nltk.corpus.stopwords.words("portuguese")

from string import punctuation

pontuacao = list()
for ponto in punctuation:
  pontuacao.append(ponto)

excluir = [":)",":(","://","https"]
```

```
excluir = [ '.', '(', ')', ',', 'https']

pontuacao_stopwords = pontuacao + palavras_irrelevantes + excluir

frase_processada = list()
for tweet in dados.tweet_text:
  nova_frase = list()
  tweet = tweet.lower()
  palavras_texto = token_espaco.tokenize(tweet)
  for palavra in palavras_texto:
    if '@' not in palavra:
      if palavra not in pontuacao_stopwords:
        nova_frase.append(palavra)
  frase_processada.append(' '.join(nova_frase))

dados["tratamento_1"] = frase_processada
```

```
dados.head()
```

| | id | tweet_text | tweet_date | sentiment | query_used | classi |
|---|---|---|---|---|---|---|
| **0** | 1031761728445530112 | @Tixaa23 14 para eu ir :) | Tue Aug 21 04:35:39 +0000 2018 | Positivo | :) | |
| **1** | 1031761040462278656 | @drexalvarez O meu like eu já dei na época :) | Tue Aug 21 04:32:55 +0000 2018 | Positivo | :) | |
| **2** | 1031760962372689920 | Eu só queria conseguir comer alguma coisa pra ... | Tue Aug 21 04:32:37 +0000 2018 | Positivo | :) | |

```
!pip install unidecode
```

```
Requirement already satisfied: unidecode in /usr/local/lib/python3.6/dist-packages (1
```

```
import unidecode

sem_acentos = [unidecode.unidecode(tweet) for tweet in dados["tratamento_1"]]
```

```
dados["tratamento_2"] = sem_acentos
```

```
dados.head()
```

| | id | tweet_text | tweet_date | sentiment | query_used | classi |
|---|---|---|---|---|---|---|
| **0** | 1031761728445530112 | @Tixaa23 14 para eu ir :) | Tue Aug 21 04:35:39 +0000 2018 | Positivo | :) | |
| **1** | 1031761040462278656 | @drexalvarez O meu like eu já dei na época :) | Tue Aug 21 04:32:55 +0000 2018 | Positivo | :) | |

```
nltk.download('rslp')
```

```
[nltk_data] Downloading package rslp to /root/nltk_data...
[nltk_data]   Package rslp is already up-to-date!
True
```

```
stopwords_sem_acento = [unidecode.unidecode(texto) for texto in pontuacao_stopwords]
```

```
stemer = nltk.RSLPStemmer()

frase_processada = list()
for tweet in dados["tratamento_2"]:
  nova_frase = list()
  palavras_texto = token_pontuacao.tokenize(tweet)
  for palavra in palavras_texto:
    if palavra not in stopwords_sem_acento:
      nova_frase.append(stemer.stem(palavra))
  frase_processada.append(' '.join(nova_frase))

dados["tratamento_3"] = frase_processada
```

```
dados
```

| | id | tweet_text | tweet_date | sentiment | query_used | c |
|---|---|---|---|---|---|---|
| **0** | 1031761728445530112 | @Tixaa23 14 para eu ir :) | Tue Aug 21 04:35:39 +0000 2018 | Positivo | :) | |
| **1** | 1031761040462278656 | @drexalvarez O meu like eu já dei na época :) | Tue Aug 21 04:32:55 +0000 2018 | Positivo | :) | |
| **2** | 1031760962372689920 | Eu só queria conseguir comer alguma coisa pra ... | Tue Aug 21 04:32:37 +0000 2018 | Positivo | :) | |
| **3** | 1031760948250456066 | :D que lindo dia ! | Tue Aug 21 04:32:33 +0000 2018 | Positivo | :) | |
| **4** | 1031760895985246208 | @Primo_Resmungao Pq da pr jeito!!é uma | Tue Aug 21 04:32:21 | Positivo | :) | |

```
1  import matplotlib.pyplot as plt
2  from wordcloud import WordCloud
3  import seaborn as sns
```

| | | | | | | |
|---|---|---|---|---|---|---|
| **785800** | 1050705144207267680 | Acordar 8 horas é tão | | | | |

```
1  def nuvem_palavras_neg(texto, coluna_texto):
2    texto_negativo = texto.query("sentiment == 'Negativo'")
3    todas_palavras = ' '.join([texto for texto in texto_negativo[coluna_texto]])
4
5    nuvem_palavras = WordCloud(width=800, height=500, max_font_size=110, collocations=F
6
7    plt.figure(figsize=(10,7))
8    plt.imshow(nuvem_palavras, interpolation='bilinear')
9    plt.axis("off")
10   plt.show()
```

| | | | | | | |
|---|---|---|---|---|---|---|
| | | @andrebraga2806 | Fri Oct 12 | | | |

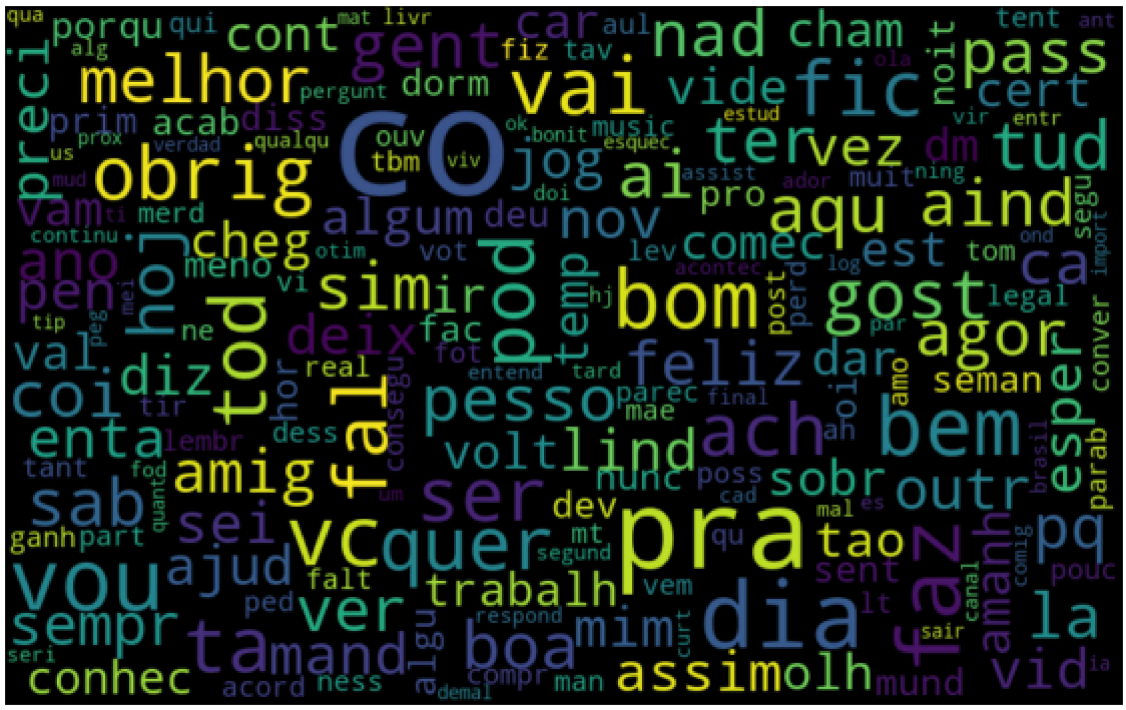
```
1  def nuvem_palavras_pos(texto, coluna_texto):
2    texto_positivo = texto.query("sentiment == 'Positivo'")
3    todas_palavras = ' '.join([texto for texto in texto_positivo[coluna_texto]])
4
5    nuvem_palavras = WordCloud(width=800, height=500, max_font_size=110, collocations=F
6
7    plt.figure(figsize=(10,7))
8    plt.imshow(nuvem_palavras, interpolation='bilinear')
9    plt.axis("off")
10   plt.show()
```

```
1  def pareto(texto, coluna_texto, quantidade):
2    todas_palavras = ' '.join([texto for texto in texto[coluna_texto]])
3
4    token_frase = token_espaco.tokenize(todas_palavras)
5    frequencia = nltk.FreqDist(token_frase)
6    df_frequencia = pd.DataFrame({"Palavra":list(frequencia.keys()),"Frequência":list(f
7    df_frequencia = df_frequencia.nlargest(quantidade, "Frequência")
```

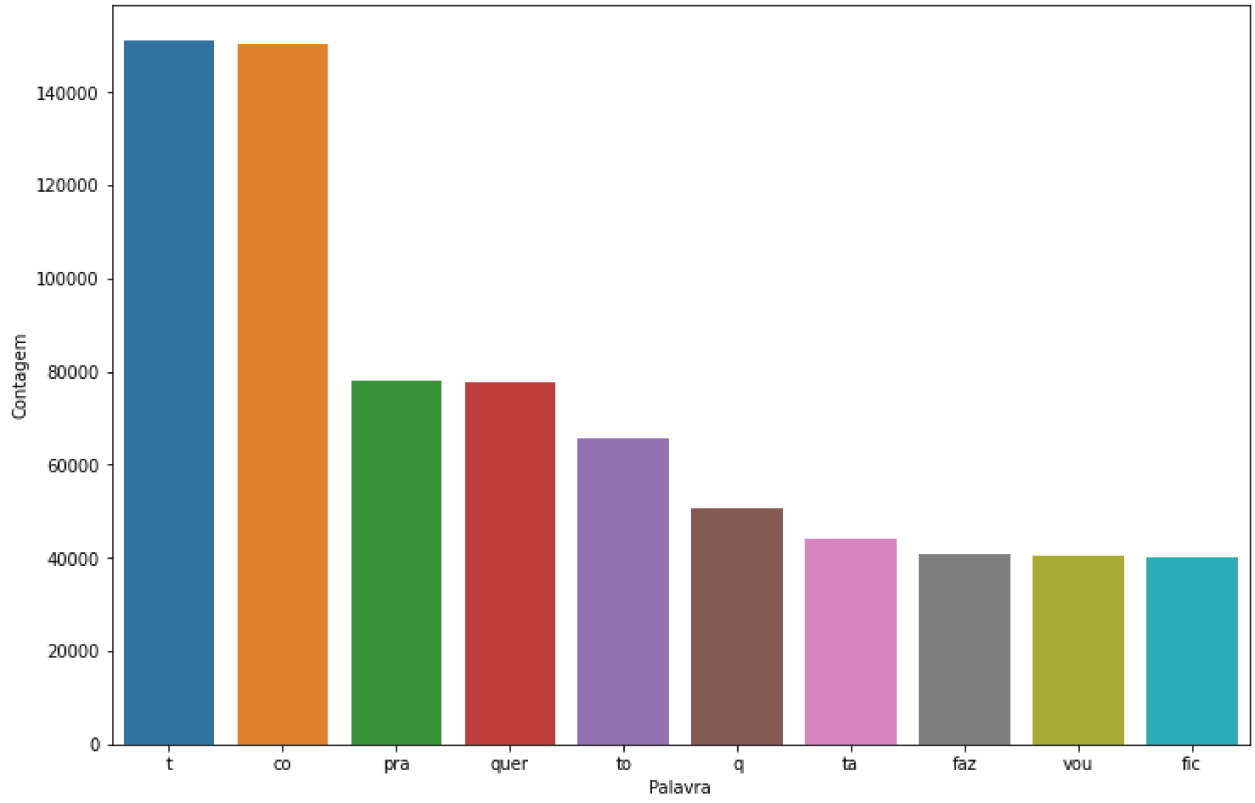
```
    plt.figure(figsize=(12,8))
    ax = sns.barplot(data= df_frequencia, x = "Palavra", y = "Frequência")
    ax.set(ylabel= "Contagem")
    plt.show()
```

```
nuvem_palavras_neg(dados, "tratamento_3")
```


```
nuvem_palavras_pos(dados, "tratamento_3")
```




```
pareto(dados, "tratamento_3", 10)
```




```
def trata_frase(frase):
  frase_tratada = list()
  nova_frase = list()
  frase = frase.lower()
  palavras_texto = token_espaco.tokenize(frase)
  for palavra in palavras_texto:
    if '@' not in palavra:
      if palavra not in pontuacao_stopwords:
        nova_frase.append(stemer.stem(palavra))
  frase_tratada.append(' '.join(nova_frase))
  frase_tratada = [unidecode.unidecode(frase) for frase in frase_tratada]

  return frase_tratada
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split

tfidf1 = TfidfVectorizer(lowercase = False, max_features=50)

tfidf_bruto = tfidf1.fit_transform(dados["tweet_text"])

treino, teste, classe_treino, classe_teste = train_test_split(tfidf_bruto, dados["cla

regressao_logistica1 = LogisticRegression(solver = "lbfgs")

regressao_logistica1.fit(treino, classe_treino)
```

```
acuracia_tfidf_bruto = regressao_logistica1.score(teste, classe_teste)

print(acuracia_tfidf_bruto)
```

```
0.6881763669866737
```

```
tfidf2 = TfidfVectorizer(lowercase = False, max_features=50)

tfidf_tratado = tfidf2.fit_transform(dados["tratamento_3"])

treino, teste, classe_treino, classe_teste = train_test_split(tfidf_tratado, dados["c

regressao_logistica2 = LogisticRegression(solver = "lbfgs")

regressao_logistica2.fit(treino, classe_treino)

acuracia_tfidf_tratado = regressao_logistica2.score(teste, classe_teste)

print(acuracia_tfidf_tratado)
```

```
0.6885021429953068
```

```
tfidf3 = TfidfVectorizer(lowercase=False,ngram_range=(1,2))

tfidf_ngrams = tfidf3.fit_transform(dados["tratamento_3"])
treino, teste, classe_treino, classe_teste = train_test_split(tfidf_ngrams, dados["cl
regressao_logistica3 = LogisticRegression(solver = "lbfgs", max_iter=1000)
regressao_logistica3.fit(treino, classe_treino)
acuracia_tfidf_ngrams = regressao_logistica3.score(teste, classe_teste)

print(acuracia_tfidf_ngrams)
```

```
0.794929092815621
```

```
print(tfidf1.get_feature_names())
```

```
['Eu', 'amo', 'as', 'bem', 'co', 'com', 'da', 'de', 'dia', 'do', 'ela', 'ele', 'em',
```

```
trata_frase("Faltá Frase")
```

```
['falt fras']
```

```
def testa_frase(frase):
  print("Regressão 1")
  print(regressao_logistica1.predict(tfidf1.transform(trata_frase(frase))))
  print(regressao_logistica1.predict_proba(tfidf1.transform(trata_frase(frase))))
  print("Regressão 2")
  print(regressao_logistica2.predict(tfidf2.transform(trata_frase(frase))))
  print(regressao_logistica2.predict_proba(tfidf2.transform(trata_frase(frase))))
  print("Regressão 3")
  print(regressao_logistica3.predict(tfidf3.transform(trata_frase(frase))))
```

```
    print(regressao_logistica3.predict_proba(tfidf3.transform(trata_frase(frase))))
```

```
testa_frase("Um dia ruim")
testa_frase("Um dia bom")
```

```
Regressão 1
[1]
[[0.36962036 0.63037964]]
Regressão 2
[1]
[[0.47902906 0.52097094]]
Regressão 3
[0]
[[0.87927611 0.12072389]]
Regressão 1
[1]
[[0.36962036 0.63037964]]
Regressão 2
[1]
[[0.22172233 0.77827767]]
Regressão 3
[1]
[[0.08002752 0.91997248]]
```

Resultados Encontrados

O treinamento sem o tratamento dos dados gerou uma taxa de acerto de 0.6881, já com o tratamento dos dados a taxa vai para 0.6885, o que não representa um ganho significativo. Utilizando-se um bigram a taxa de acerto foi de 0.7949, que representa um aumento de 15,5% com relação aos dados sem tratamento. Foi possível observar também que a confiança do resultado é significativamente maior utilizando bigrams.