



## NLP - IMDB Dataset - Sentiment Analysis

```
In [1]: 1 import pandas as pd
```

```
In [2]: 1 resenha = pd.read_csv('imdb-reviews-pt-br.csv')
2 resenha.head()
```

	<b>id</b>	<b>text_en</b>	<b>text_pt</b>	<b>sentiment</b>
0	1	Once again Mr. Costner has dragged out a movie...	Mais uma vez, o Sr. Costner arrumou um filme p...	neg
1	2	This is an example of why the majority of acti...	Este é um exemplo do motivo pelo qual a maiori...	neg
2	3	First of all I hate those moronic rappers, who...	Primeiro de tudo eu odeio esses raps imbecis, ...	neg
3	4	Not even the Beatles could write songs everyon...	Nem mesmo os Beatles puderam escrever músicas ...	neg
4	5	Brass pictures movies is not a fitting word fo...	Filmes de fotos de latão não é uma palavra apr...	neg

```
In [3]: 1 resenha
```

	<b>id</b>	<b>text_en</b>	<b>text_pt</b>	<b>sentiment</b>
0	1	Once again Mr. Costner has dragged out a movie...	Mais uma vez, o Sr. Costner arrumou um filme p...	neg
1	2	This is an example of why the majority of acti...	Este é um exemplo do motivo pelo qual a maiori...	neg
2	3	First of all I hate those moronic rappers, who...	Primeiro de tudo eu odeio esses raps imbecis, ...	neg
3	4	Not even the Beatles could write songs everyon...	Nem mesmo os Beatles puderam escrever músicas ...	neg
4	5	Brass pictures movies is not a fitting word fo...	Filmes de fotos de latão não é uma palavra apr...	neg
...	...	...	...	...
49454	49456	Seeing as the vote average was pretty low, and...	Como a média de votos era muito baixa, e o fat...	pos
49455	49457	The plot had some wretched, unbelievable twist...	O enredo teve algumas reviravoltas infelizes e...	pos
49456	49458	I am amazed at how this movie and most others h...	Estou espantado com a forma como este filme e ...	pos
49457	49459	A Christmas Together actually came before my t...	A Christmas Together realmente veio antes do m...	pos
49458	49460	Working-class romantic drama from director Mar...	O drama romântico da classe trabalhadora do di...	pos

49459 rows × 4 columns

```
In [4]: 1 from sklearn.model_selection import train_test_split
```

```
2
3 treino, teste, classe_treino, classe_teste = train_test_split(resenha.text_pt,
4
5                                         resenha.sentiment,
6                                         random_state = 42)
```

```
In [5]: 1 treino
```

```
1348 Embora o filme tenha sido apenas assim, o clos...
27466 Este é provavelmente um dos piores filmes que ...
29998 De vez em quando, um filme irá varrer ao seu r...
48186 Este é um conto completamente diabólico de quâ...
26473 Lenta, chata, extremamente repetitiva. Não adm...
...
11284 Naach teria ganhado um Razzie para o Pior Film...
44732 Apenas assisti a esse filme em DVD e achei a a...
38158 Melhor show desde Seinfeld. Ela é realmente mu...
860 Eu pareço estar discordando com muitas pessoas...
15795 Minhas duas filhas de 11 e 13 anos e eu tive s...
Name: text_pt, Length: 37894, dtype: object
```

```
In [6]: 1 teste
```

```
12532 Isso era incomum: um filme moderno que era ult...
35445 Alguns dos meus velhos amigos sugeriram que eu...
20279 Que prazer. Isto é realmente uma paródia. Some...
2969 Há cerca de dez minutos a meio da Strangeland...
45161 Otelo, a clássica história de Shakespeareen sob...
...
16421 Crescendo como filho do cinema, uma das trilog...
39861 Este filme é o melhor filme de todos os tempos...
309 "Electra Glide in Blue" é um movimento lento e...
20638 Eu amo esse filme ! Eu acho que já vi 5 vezes ...
38935 Eu vi alguns filmes sobre transtornos alimenta...
Name: text_pt, Length: 12365, dtype: object
```

```
In [7]: 1 classe_treino
```

```
1348 neg
27466 neg
29998 neg
48186 pos
26473 neg
...
```

```

11284    neg
44732    pos
38158    pos
860     neg
15795    pos
Name: sentiment, Length: 37094, dtype: object

```

In [8]: 1 classe teste

```

12532    pos
35445    neg
20279    pos
2969    neg
45161    pos
...
16421    pos
39861    pos
389     neg
20638    pos
38935    pos
Name: sentiment, Length: 12365, dtype: object

```

In [9]: 1 from sklearn.linear\_model import LogisticRegression

```

2
3 # regressao_logistica = LogisticRegression()
4 # regressao_logistica.fit(treino, classe_treino)
5 #acuracia = regressao_logistica.score(teste, classe_teste)
6 #print(acuracia)
7
8 # a célula não rodará porque é necessário fazer a etapa de NLP nesse momento
9 # o erro será do tipo ValueError por conflito de tipagem.

```

In [10]: 1 print('Negativa \n')

```

2 print(resenha.text_pt[189])
3
4 #leia para analisar as características do texto.

```

Negativa

Este é sem dúvida o pior filme que eu já vi. E acredite em mim, eu vi muitos filmes. A reviravolta inacreditável que o filme faz - passando de um extremamente mau filme "Formas de vida alienígenas habitam a terra", com um filme que tenta espalhar um arquicristão "O dia do julgamento está próximo, buscar Jesus ou queimar por toda a eternidade em as dividas ardentes do inferno "mensagen - deixou-me atordoado de pois de ter sido atormentado por 85 minutos. Até mesmo os cristãos religiosos devem se envergonhar ou ficar furiosos ao ver suas crenças postadas dessa maneira. Eu não sabia o que fazer comigo quando assisti a atuação horrível que poderia ter sido realizada por crianças de 7 anos de idade. Simplesmente repugnante. Eu não sou cristão nem muito religioso. Mas se eu estivesse, não teria mais medo do Inferno. Rich Christiano mostrou ser algo muito pior.

In [11]: 1 print('Positivo \n')

```

2 print(resenha.text_pt[49002])

```

Positivo

Crescendo em Nova York no final dos anos 80 e início dos anos 90, posso dizer pessoalmente que este é um dos documentários mais importantes feitos para cobrir esse lugar neste período de tempo. Não Madonna não veio com a ideia de Voguing, mas é de onde ela tirou! Em vez de combater a violência uns dos outros ou em brigas de gato, o voguing permitiu que as pessoas "lutassem" dentro dos confins de tudo, menos que tocassem umas às outras, o que justificaria uma desqualificação automática. Vendo este tipo de extraordinariamente talentosas / bairros orquestradas "jogadas" nos clubes foi nada menos do que espetacular e todos os grandes nomes de antigamente estão aqui ... Pepper La Beija, Paris Duprée, Xtravaganza, etc. ... tudo comemorado nos gostos de peças de época como a música de Malcolm McLaren's "Deep in Vogue" ... não importava quem você era, ou de onde você era porque quando você passava por aquelas portas nesse "reino mágico" de De certa forma, você se tornou parte de algo maior que você / você era importante / e, o mais importante, a criação de seus próprios movimentos e imaginação ... e que alquer pessoa de qualquer lugar poderia se tornar Rei ou Rainha como poderia ter sido. As palavras e inteligência eram tão afiadas quanto os movimentos no chão. Toda a tensão, excitação e magia dessa energia urbana de NYC é capturada neste filme. BRILHANTE!!! POR FAVOR, LIBER E EM DVD para o mundo ver !!! Obrigado!

In [12]: 1 # verificando a proporcionalidade dos dados.

```

2 print(resenha.sentiment.value_counts())

```

```

sentiment
neg    24765
pos    24694
Name: count, dtype: int64

```

In [13]: 1 #transformando dados

```

2 classificacao = resenha['sentiment'].replace(['neg', 'pos'], [0,1])
3 resenha['classificacao'] = classificacao
4 resenha.head()

```

	id	text_en	text_pt	sentiment	classificacao
0	1	Once again Mr. Costner has dragged out a movie...	Mais uma vez, o Sr. Costner arrumou um filme p...	neg	0
1	2	This is an example of why the majority of acti...	Este é um exemplo do motivo pelo qual a maioria...	neg	0
2	3	First of all I hate those moronic rappers, who...	Primeiro de tudo eu odeio esses raps imbecis, ...	neg	0
3	4	Not even the Beatles could write songs everyon...	Nem mesmo os Beatles puderam escrever músicas ...	neg	0
4	5	Brass pictures movies is not a fitting word fo...	Filmes de fotos de latão não é uma palavra aprop...	neg	0

In [14]: 1 resenha.tail()

	id	text_en	text_pt	sentiment	classificacao
49454	49456	Seeing as the vote average was pretty low, and...	Como a média de votos era muito baixa, e o fat...	pos	1
49455	49457	The plot had some wretched, unbelievable twist...	O enredo teve algumas reviravoltas infelizes e...	pos	1
49456	49458	I am amazed at how this movie and most others h...	Estou espantado com a forma como este filme e ...	pos	1
49457	49459	A Christmas Together actually came before my t...	A Christmas Together realmente veio antes do m...	pos	1
49458	49460	Working-class romantic drama from director Mar...	O drama romântico da classe trabalhadora do di...	pos	1

```

In [15]: 1 from sklearn.feature_extraction.text import CountVectorizer
2
3 texto = ['Assisti um filme ótimo', 'Assisti um filme ruim']
4
5 vetorizar = CountVectorizer(lowercase=False)
6 bag_of_words = vetorizar.fit_transform(texto)

In [16]: 1 vetorizar.get_feature_names_out()

array(['Assisti', 'filme', 'ruim', 'um', 'ótimo'], dtype=object)

In [17]: 1 bag_of_words

<2x5 sparse matrix of type '<class 'numpy.int64'>'  

       with 8 stored elements in Compressed Sparse Row format>

In [18]: 1 matriz_esparsa = pd.DataFrame.sparse.from_spmatrix(bag_of_words, columns = vetorizar.get_feature_names_out())

In [19]: 1 matriz_esparsa

      Assisti filme ruim um ótimo
0 1 1 0 1 1
1 1 1 1 1 0

In [20]: 1 vetorizar = CountVectorizer(lowercase=False, max_features=50)
2 bag_of_words = vetorizar.fit_transform(resenha.text_pt)
3 print(bag_of_words.shape)

(49459, 50)

In [21]: 1 treino, teste, classe_treino, classe_teste = train_test_split(bag_of_words,
2                                         resenha.classificacao,
3                                         random_state = 42)
4
5 regressao_logistica = LogisticRegression()
6 regressao_logistica.fit(treino, classe_treino)
7 acuracia = regressao_logistica.score(teste, classe_teste)
8 print(acuracia)

0.6583097452486858

In [22]: 1 # Se blocos inteiros de código estão tendo que ser invocados novamente, é melhor criar uma função.
2
3 def classificar_texto(texto, coluna_texto, coluna_classificacao):
4     vetorizar = CountVectorizer(lowercase=False, max_features=50)
5     bag_of_words = vetorizar.fit_transform(texto[coluna_texto])
6     treino, teste, classe_treino, classe_teste = train_test_split(bag_of_words,
7                                                                     texto[coluna_classificacao],
8                                                                     random_state = 42)
9     regressao_logistica = LogisticRegression()
10    regressao_logistica.fit(treino, classe_treino)
11    return regressao_logistica.score(teste, classe_teste)
12
13 print(classificar_texto(resenha, "text_pt", 'classificacao'))

0.6583097452486858

In [23]: 1 !pip install wordcloud

Requirement already satisfied: wordcloud in c:\users\bruno\anaconda3\lib\site-packages (1.9.2)
Requirement already satisfied: numpy>=1.6.1 in c:\users\bruno\anaconda3\lib\site-packages (from wordcloud) (1.24.3)
Requirement already satisfied: pillow in c:\users\bruno\anaconda3\lib\site-packages (from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in c:\users\bruno\anaconda3\lib\site-packages (from wordcloud) (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\bruno\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\bruno\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\bruno\anaconda3\lib\site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\bruno\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\bruno\anaconda3\lib\site-packages (from matplotlib->wordcloud) (23.1)
Requirement already satisfied: pyparsing>=3.1,>=2.3.1 in c:\users\bruno\anaconda3\lib\site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\bruno\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\bruno\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)

In [24]: 1 import matplotlib.pyplot as plt
2 %matplotlib inline
3 from wordcloud import WordCloud
4
5 todas_palavras = ''.join([texto for texto in resenha.text_pt])

In [25]: 1 len(todas_palavras)

63448424

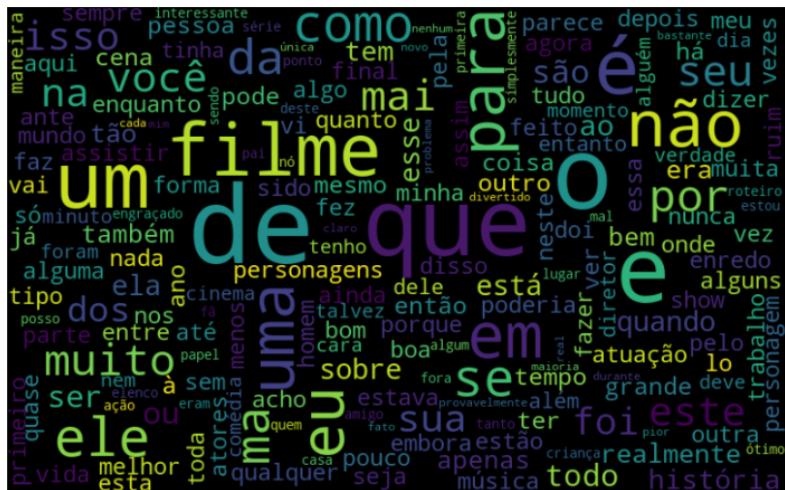
```

```
In [26]: 1 nuvem_palavras = WordCloud(width=800, height=500, max_font_size=110, collocations = False).generate(tod  
2 nuvem_palavras
```

```
<wordcloud.wordcloud.WordCloud at 0x16e400b4510>

In [27]: 1 plt.figure(figsize=(10,7))
2 plt.imshow(nuvem_palavras, interpolation='bilinear')
3 plt.axis('off')
4 plt.show

<function matplotlib.pyplot.show(close=None, block=None)>
```



Separando o dataset em sentimento "positivo" e "negativo" para analisar com o WordCloud.

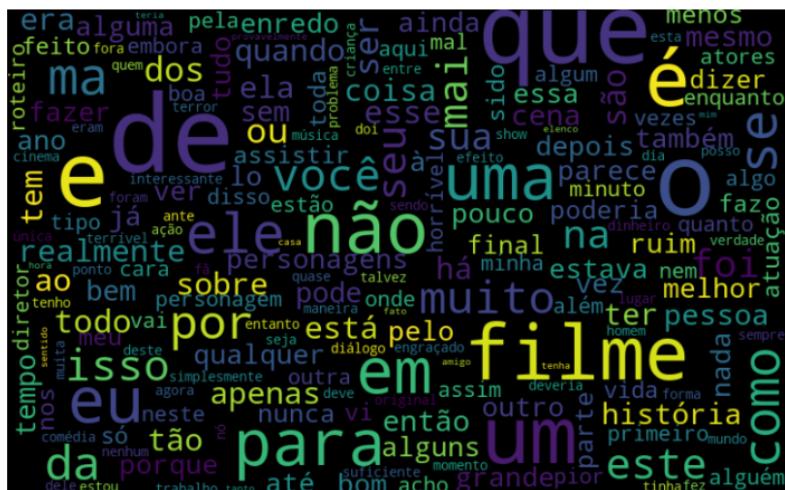
```
In [28]: 1 resenha.query('sentiment == "pos"')
```

<b>id</b>	<b>text_en</b>	<b>text_pt</b>	<b>sentiment</b>	<b>classificacao</b>
12389	12391 I went and saw this movie last night after bei...	Eu fui e vi este filme ontem à noite depois de...	pos	1
12390	12392 Actor turned director Bill Paxton follows up h...	O diretor do ator, Bill Paxton, segue sua prom...	pos	1
12391	12393 As a recreational golfer with some knowledge o...	Como um jogador de recreio com algum conhecime...	pos	1
12392	12394 I saw this film in a sneak preview, and it is ...	Eu vi esse filme em uma prévia, e é delicioso....	pos	1
12393	12395 Bill Paxton has taken the true story of the 19...	Bill Paxton levou a verdadeira história do gol...	pos	1
...	...	...	...	...
49454	49456 Seeing as the vote average was pretty low, and...	Como a média de votos era muito baixa, e o fat...	pos	1
49455	49457 The plot had some wretched, unbelievable twist...	O enredo teve algumas reviravoltas infelizes e...	pos	1
49456	49458 I am amazed at how this movie and most others h...	Estou espantado com a forma como este filme e ...	pos	1
49457	49459 A Christmas Together actually came before my t...	A Christmas Together realmente veio antes do m...	pos	1
49458	49460 Working-class romantic drama from director Mar...	O drama romântico da classe trabalhadora do di...	pos	1

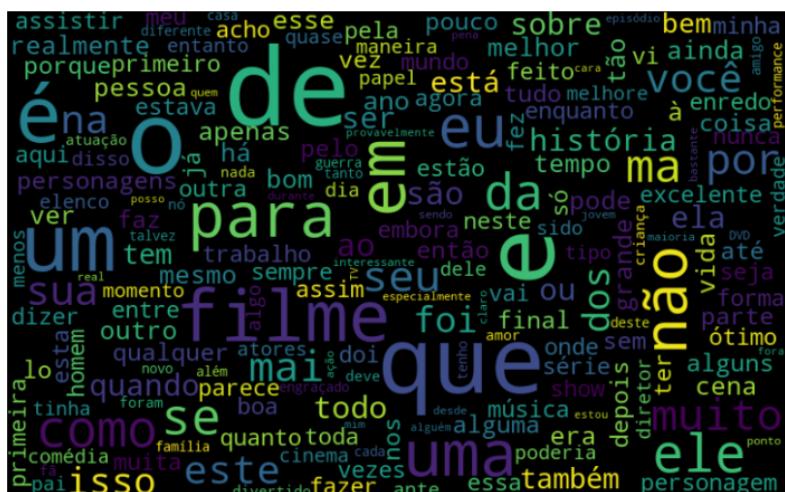
```
In [29]: 1 def nuvem_palavras_neg(texto, coluna_texto):
2     texto_negativo = texto.query('sentiment == "neg"')
3     todas_palavras = ' '.join([texto for texto in texto_negativo[coluna_texto]])
4
5     nuvem_palavras = WordCloud(width=800,
6                                 height=500,
7                                 max_font_size=110,
8                                 collocations = False).generate(todas_palavras)
9
10    plt.figure(figsize=(10,7))
11    plt.imshow(nuvem_palavras, interpolation='bilinear')
12    plt.axis('off')
13    plt.show
```

```
In [30]: 1 def nuvem_palavras_pos(texto, coluna_texto):
2     texto_positivo = texto.query('sentiment == "pos"')
3     todas_palavras = ' '.join([texto for texto in texto_positivo[coluna_texto]])
4
5     nuvem_palavras = WordCloud(width=800,
6                                 height=500,
7                                 max_font_size=110,
8                                 collocations = False).generate(todas_palavras)
9
10    plt.figure(figsize=(10,7))
11    plt.imshow(nuvem_palavras, interpolation='bilinear')
12    plt.axis('off')
13    plt.show
```

```
In [31]: 1 nuvem palavras neg(resenha, 'text_pt')
```



```
In [32]: 1 nuvem palavras pos(resenha, 'text_pt')
```



```
In [33]: 1 import nltk  
          2  
          3 nltk.download("all")
```

```
[nltk_data] Downloading collection 'all'
[nltk_data]
[nltk_data] |   Downloading package abc to
[nltk_data] |   C:/Users/bruno/AppData/Roaming/nltk_data...
[nltk_data] |   Package abc is already up-to-date!
[nltk_data] |   Downloading package alpino to
[nltk_data] |   C:/Users/bruno/AppData/Roaming/nltk_data...
[nltk_data] |   Package alpino is already up-to-date!
[nltk_data] |   Downloading package averaged_perceptron_tagger to
[nltk_data] |   C:/Users/bruno/AppData/Roaming/nltk_data...
[nltk_data] |   Package averaged_perceptron_tagger is already up-
[nltk_data] |       to-date!
[nltk_data] |   Downloading package averaged_perceptron_tagger_ru to
[nltk_data] |   C:/Users/bruno/AppData/Roaming/nltk_data...
[nltk_data] |   Package averaged_perceptron_tagger_ru is already up-
[nltk_data] |       to-date!
[nltk_data] |   Downloading package basque_grammars to
[nltk_data] |   C:/Users/bruno/AppData/Roaming/nltk_data...
[nltk_data] |   Package basque_grammars is already up-to-date!
[nltk_data] |   Downloading package bcpc7 to
[nltk_data] |   C:/Users/bruno/AppData/Roaming/nltk_data...
[nltk_data] |   Package bcpc7 is already up-to-date!
[nltk_data] |   Downloading package biocreative_ppi to
```

```
In [34]: frase = ["um filme ruim", "um filme bom"]
frecuencia = nltk.FreqDist(frase)
frecuencia
```

```
FreqDist({'um filme ruim': 1, 'um filme bom': 1})
```

```
In [35]: 1 from nltk import tokenize  
2  
3 frase = "Bem vindo ao mundo do PLN!"  
4  
5 token.espaço = tokenize.WhitespaceTokenizer()
```

```
6 token_frase = token_espaco.tokenize(frase)
7 print(token_frase)

['Bem', 'vindo', 'ao', 'mundo', 'do', 'PLN!']
```

```
In [36]: 1 token_frase = token_espaco.tokenize(todas_palavras)
2 frequencia = nltk.FreqDist(token_frase)
```

```
In [37]: 1 frequencia
```

```
FreqDist({'de': 417651, 'que': 325070, 'e': 299743, 'o': 244881, 'um': 216410, 'a': 210179, 'é': 192381, 'em': 132778, 'uma': 130888, 'não': 127915, ...})
```

```
In [38]: 1 df_frequencia = pd.DataFrame({"Palavra": list(frequencia.keys()),
2                               "Frequencia": list(frequencia.values())})
3 df_frequencia
```

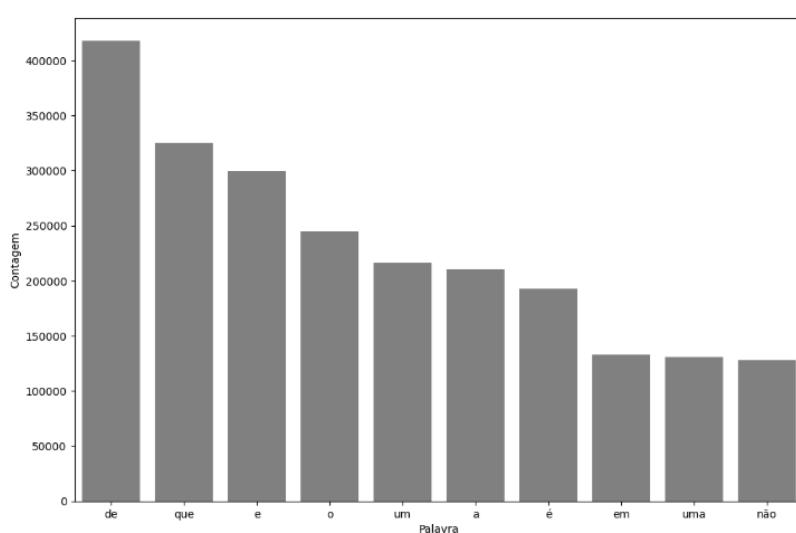
	Palavra	Frequencia
0	Mais	1538
1	uma	130888
2	vez,	1927
3	o	244881
4	Sr.	1741
...	...	...
348280	Muppified,	1
348281	inventora	1
348282	"Union	1
348283	beirar	1
348284	rosados.	1

348285 rows × 2 columns

```
In [39]: 1 df_frequencia.nlargest(columns = "Frequencia", n = 10)
```

	Palavra	Frequencia
20	de	417651
14	que	325070
42	e	299743
3	o	244881
7	um	216410
102	a	210179
45	é	192381
200	em	132778
1	uma	130888
29	não	127915

```
In [40]: 1 import seaborn as sns
2
3 plt.figure(figsize=(12,8))
4 ax = sns.barplot(data = df_frequencia.nlargest(columns="Frequencia", n=10),
5                   x = "Palavra",
6                   y = "Frequencia",
7                   color = 'gray')
8 ax.set(ylabel = "Contagem")
9 plt.show()
```

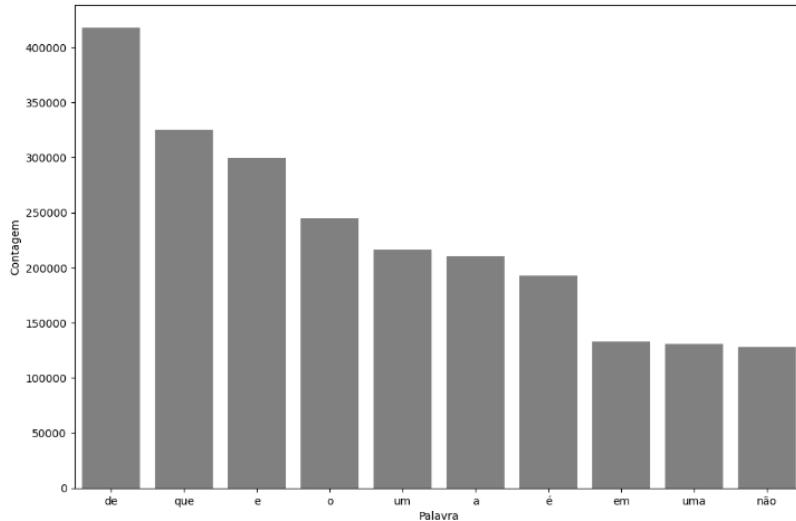


```
In [42]: 1 # Montando a função
2 def pareto(texto, coluna_texto, quantidade):
```

```

3     todas_palavras = ' '.join([texto for texto in texto[coluna_texto]])
4
5     token_frase = token_espaco.tokenize(todas_palavras)
6     frequencia = nltk.FreqDist(token_frase)
7     df_frequencia = pd.DataFrame({"Palavra": list(frequencia.keys()),
8                                    "Frequencia": list(frequencia.values())})
9
10    df_frequencia = df_frequencia.nlargest(columns = "Frequencia", n = quantidade)
11    plt.figure(figsize=(12,8))
12    ax = sns.barplot(data = df_frequencia,
13                      x = "Palavra",
14                      y = "Frequencia",
15                      color = 'gray')
16    ax.set(ylabel = "Contagem")
17    plt.show()
18
19 pareto(resenha, "text_pt", 10)

```



```
In [43]: 1 palavras_irrelevantes = nltk.corpus.stopwords.words('portuguese')
```

```
In [44]: 1 palavras_irrelevantes
```

```
['a',
 'á',
 'ao',
 'aos',
 'aquela',
 'aqueelas',
 'aquele',
 'aqueles',
 'aquilo',
 'as',
 'ás',
 'até',
 'com',
 'como',
 'da',
 'das',
 'de',
 'dela',
 'delas',
 'dele',
 'deles',
 'depois',
 'do',
 'não']
```

```
In [45]: 1 frase_processada = list()
2 for opiniao in resenha.text_pt:
3     nova_frase = list()
4     palavras_texto = token_espaco.tokenize(opiniao)
5     for palavra in palavras_texto:
6         if palavra not in palavras_irrelevantes:
7             nova_frase.append(palavra)
8     frase_processada.append(' '.join(nova_frase))
9
10 resenha['tratamento_1'] = frase_processada
```

```
In [46]: 1 resenha.head()
```

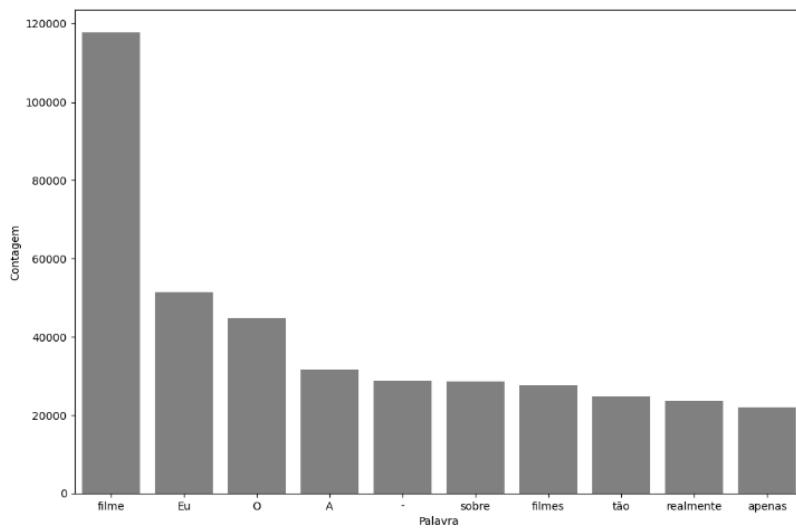
	<b>id</b>	<b>text_en</b>	<b>text_pt</b>	<b>sentiment</b>	<b>classificacao</b>	<b>tratamento_1</b>
0	1	Once again Mr. Costner has dragged out a movie...	Mais uma vez, o Sr. Costner arrumou um filme p...	neg	0	Mais vez, Sr. Costner arrumou filme tempo nece...
1	2	This is an example of why the majority of acti...	Este é um exemplo do motivo pelo qual a maioria...	neg	0	Este exemplo motivo maioria filmes ação mesmos...
2	3	First of all I hate those moronic	Primeiro de tudo eu odeio esses	neu	0	Primeiro tudo odeio raps imbecis,

rappers, who...	raps impecis, ...	-	-	-
3 4 Not even the Beatles could write songs everyone...	Nem mesmo os Beatles puderam escrever músicas ...	neg	0	Nem Beatles puderam escrever músicas todos gos...
4 5 Brass pictures movies is not a fitting word fo...	Filmes de fotos de latão não é uma palavra apr...	neg	0	Filmes fotos latão palavra apropriada eles, ve...

```
In [47]: 1 classificar_texto(resenha, "tratamento_1", "classificacao")
```

0.6811160533764659

```
In [48]: 1 pareto(resenha, "tratamento_1", 10)
```



```
In [ ]: 1
```