



linkedin.com/in/brunogianetti

NLP - IMDB Dataset - Sentiment Analysis

In [1]:
1 import pandas as pdIn [2]:
1 resenha = pd.read_csv('imdb-reviews-pt-br.csv')
2 resensa.head()

	id	text_en	text_pt	sentiment
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 resensa

	id	text_en	text_pt	sentiment
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(resensa.text_pt,
4
5
resensa.sentiment,
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: 37094, 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
309     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 dívidas ardentes do inferno "mensagem - 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 permitia 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 / bem orquestradas "jogadas" nos clubes foi nada menos do que espetacular e todos os grandes nomes de antigamente estão aqui ... Pepper La Beija, Paris Dupré, 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 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 qualquer 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 apr...	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_
```

```
In [19]: 1 matriz_esparsa
```

	Assisti	filme	ruim	um	ótimo
0 1	1	1	0	1	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 estiverem sendo chamados 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) (1.16.8)
```

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

In [25]: 1 len(todas_palabras)

62446464

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

[1.3.1](#) [1.3.2](#) [1.3.3](#) [1.3.4](#) [1.3.5](#) [1.3.6](#) [1.3.7](#) [1.3.8](#) [1.3.9](#) [1.3.10](#)

```
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"')
```

id		text_en	text_pt	sentiment	classificacao
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 delicios....	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

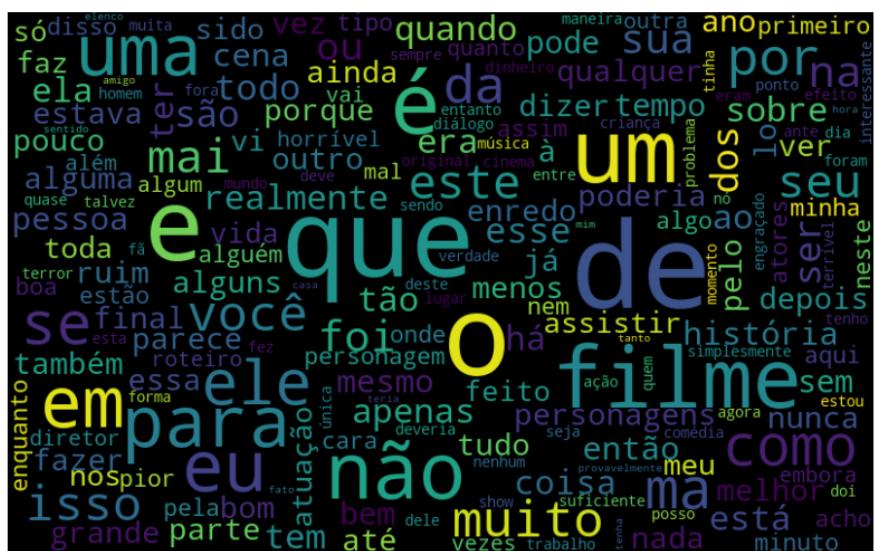
49455	49457	The plot had some wretched, unbelievable twist...	O enredo teve algumas reviravoltas infelizes e...	pos	1
49456	49458	I am amazed at how this movie and most others h...	Estou espantado com a forma como este filme e ...	pos	1
49457	49459	A Christmas Together actually came before my t...	A 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

24694 rows × 5 columns

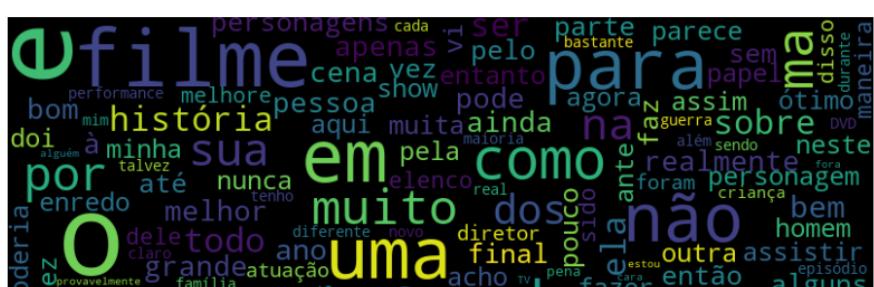
```
In [29]: 1 def nuvem_palavras_neg(texto, coluna_texto):
2     texto_negativo = texto.query('sentiment == "neg"')
3     todas_palavras = ' '.join([texto for texto in texto_negativo[coluna_texto]])
4
5     nuvem_palavras = WordCloud(width=800,
6                                 height=500,
7                                 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
```

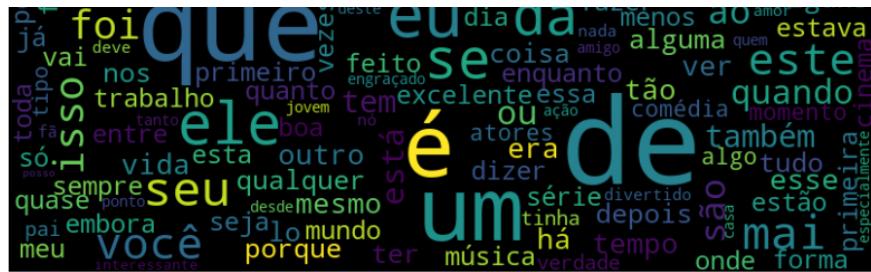
```
In [30]: 1 def nuvem_palavras_pos(texto, coluna_texto):
2     texto_positivo = texto.query('sentiment == "pos"')
3     todas_palavras = ' '.join([texto for texto in texto_positivo[coluna_texto]])
4
5     nuvem_palavras = WordCloud(width=800,
6                                 height=500,
7                                 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  
[nltk_data] |   up-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 bcp47 to  
[nltk_data] |   C:\Users\bruno\AppData\Roaming\nltk_data...  
[nltk_data] |   Package bcp47 is already up-to-date!  
[nltk_data] | Downloading package biocreative_ppi to  
[nltk_data] |   C:\Users\bruno\AppData\Roaming\nltk_data...  
  
In [34]: 1 frase = ["um filme ruim", "um filme bom"]  
2 frequencia = nltk.FreqDist(frase)  
3 frequencia  
  
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_espacos = tokenize.WhitespaceTokenizer()  
6 token_frase = token_espacos.tokenize(frase)  
7 print(token_frase)  
  
['Bem', 'vindo', 'ao', 'mundo', 'do', 'PLN!']  
  
In [36]: 1 token_frase = token_espacos.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	Frequência
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

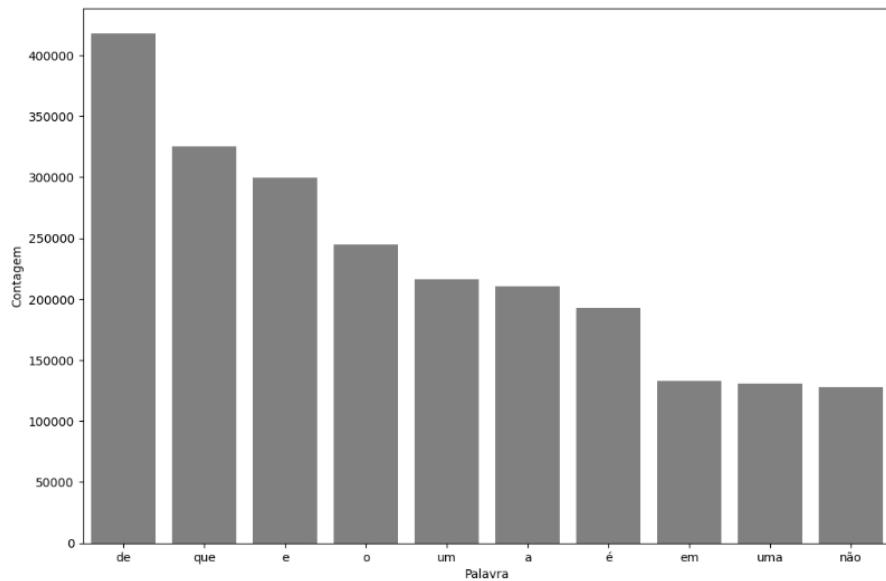
In [39]:

```
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]:

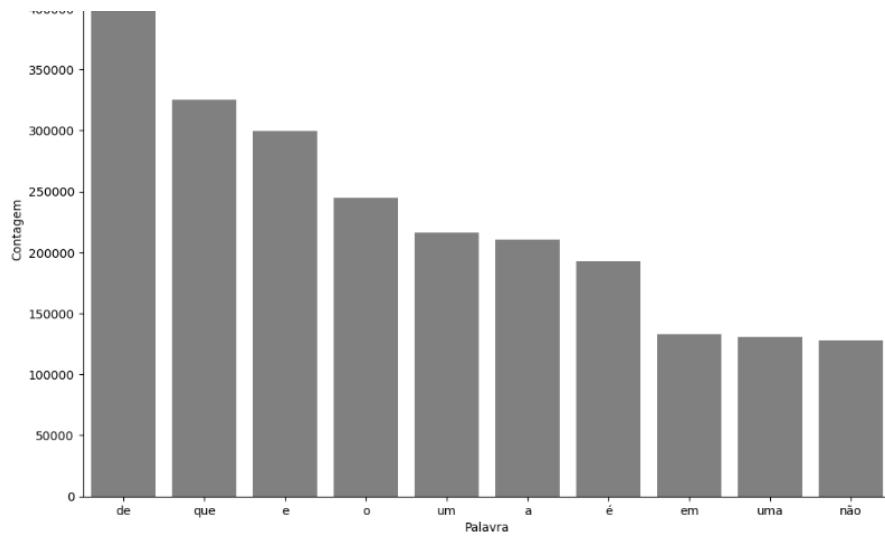
```
1 import seaborn as sns
2
3 plt.figure(figsize=(12,8))
4 ax = sns.barplot(data = df_frequencia.nlargest(columns="Frequencia", n=10),
5                   x = "Palavra",
6                   y = "Frequencia",
7                   color = 'gray')
8 ax.set(ylabel = "Contagem")
9 plt.show()
```



In [41]:

```
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 [42]: 1 palavras_irrelevantes = nltk.corpus.stopwords.words('portuguese')
```

```
In [43]: 1 palavras_irrelevantes
```

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

```
In [44]: 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 [45]: 1 resenha.head()
```

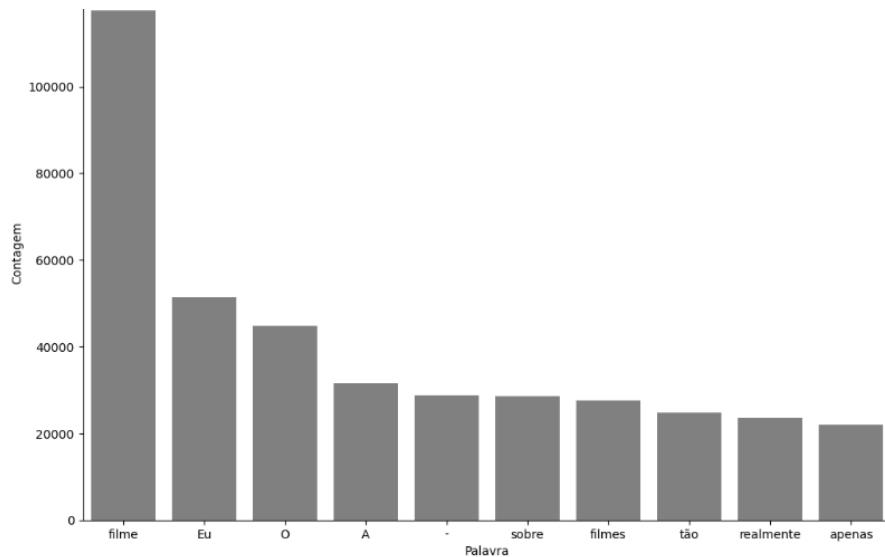
		text_en	text_pt	sentiment	classificacao	tratamento_1
0	1	Once again Mr. Costner has dragged out a movie...	Mais uma vez, o Sr. Costner arrumou um filme p...	neg	0	Mais vez, Sr. Costner arrumou filme tempo nece...
1	2	This is an example of why the majority of acti...	Este é um exemplo do motivo pelo qual a maiori...	neg	0	Este exemplo motivo maioria filmes ação mesmos...
2	3	First of all I hate those moronic rappers, who...	Primeiro de tudo eu odeio esses raps imbecis, ...	neg	0	Primeiro tudo odeio raps imbecis, poderiam agi...
3	4	Not even the Beatles could write songs everyon...	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 apt...	neg	0	Filmes fotos latão palavra apropriada eles, ve...

```
In [46]: 1 classificar_texto(resenha, "tratamento_1", "classificacao")
```

```
0.6811160533764659
```

```
In [47]: 1 pareto(resenha, "tratamento_1", 10)
```





Retirando pontuações

```
In [48]: 1 from nltk import tokenize
2 from string import punctuation
3
4 token_pontuacao = tokenize.WordPunctTokenizer()
5
6 pontuacao = list()
7 for ponto in punctuation:
8     pontuacao.append(ponto)
9
10 pontuacao_stopwords = pontuacao + palavras_irrelevantes
11
12 frase_processada = list()
13 for opiniao in resenha['tratamento_1']:
14     nova_frase = list()
15     palavras_texto = token_pontuacao.tokenize(opiniao)
16     for palavra in palavras_texto:
17         if palavra not in pontuacao_stopwords:
18             nova_frase.append(palavra)
19     frase_processada.append(' '.join(nova_frase))
20
21 resenha['tratamento_2'] = frase_processada
```

```
In [49]: 1 resenha.head()
```

	id	text_en	text_pt	sentiment	classificacao	tratamento_1	tratamento_2
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...	Mais vez Sr Costner arrumou filme tempo necess...
1	2	This is an example of why the majority of acti...	Este é um exemplo do motivo pelo qual a maiori...	neg	0	Este exemplo motivo maioria filmes ação mesmos...	Este exemplo motivo maioria filmes ação mesmos...
2	3	First of all I hate those moronic rappers, who...	Primeiro de tudo eu odeio esses raps imbecis, ...	neg	0	Primeiro tudo odeio raps imbecis, poderiam agi...	Primeiro tudo odeio raps imbecis poderiam agir...
3	4	Not even the Beatles could write songs everyon...	Nem mesmo os Beatles puderam escrever músicas ...	neg	0	Nem Beatles puderam escrever músicas todos gos...	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...	Filmes fotos latão palavra apropriada verdade ...

```
In [50]: 1 resenha['tratamento_1'][0]
```

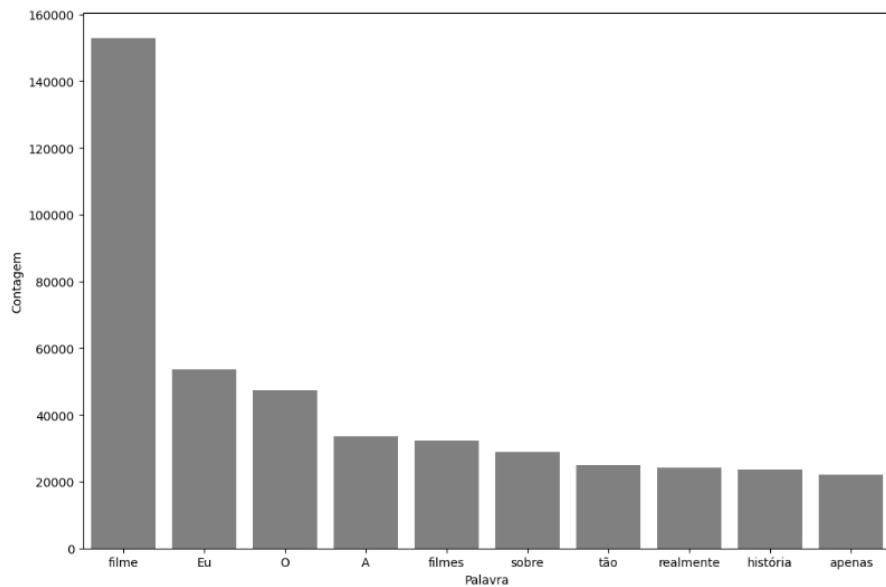
"Mais vez, Sr. Costner arrumou filme tempo necessário. Além terríveis sequências resgate mar, quais poucas, simplesmente importei nenhum personagens. A maioria fantasmas armário, personagem Costers realizado logo inicio, esquecido tarde, importava. O personagem deveríamos importar arrogante superconfiante, Ashton Kutcher. O problema sai garoto pensa melhor qualquer outra pessoa redor mostra sinais armário desordenado. Seu único obstáculo parece vencendo Costner. Finalmente, bem além meio caminho, costner conta sobre fantasmas Kutchers. Somos informados Kutcher levado melhor pressentimentos presságios anteriores. Nenhuma mágica aqui, tudo podia fazer desligar hora.'

```
In [51]: 1 resenha['tratamento_2'][0]
```

"Mais vez Sr Costner arrumou filme tempo necessário Além terríveis seqüências resgate mar quais poucas simplesmente importei nenhum personagens A maioria fantasmas armário personagem Costers realizado logo inicio esquecido tarde importava o personagem deveríamos importar arrogante superconfiante Ashton Kutcher O problema sai garoto pensa melhor qualquer outra pessoa redor mostra sinais armário desordenado seu único obstáculo parece vencendo Costner Finalmente bem além meio caminho Costner conta sobre fantasmas Kutchers Somos informados Kutcher le

vado melhor pressentimentos presságios anteriores Nenhuma mágica aqui tudo podia fazer desligar hora

```
In [52]: 1 pareto(resenha, "tratamento 2", 10)
```



```
In [53]: 1 import unidecode  
2  
3 sem_acentos = [unidecode.unidecode(texto) for texto in resenha['tratamento_2']]
```

In [54]: 1 sem acentos[0]

'Mais vez Sr Costner arrumou filme tempo necessario Alem terriveis sequencias resgate mar quais poucas simplesmente importei nenhum personagens A maioria fantasmas armario personagem Costers realizado logo inicio esquecido tarde importava O personagem deveriamos importar arrojante superconfiante Ashton Kutcher O problema sai garoto pensa melhor qualquer outra pessoa redor mostra sinais armario desordenado Seu unico obstaculo parece vencendo Costner Finalmente bem alem meio caminho Costner conta sobre fantasmas Kutchers Somos informados Kutcher levado melhor pressentimentos pressagios anteriores Nenhuma magica aqui tudo podia fazer desligar hora'

```
In [55]: 1 stopwords sem acento = [unidecode.unidecode(texto) for texto in pontuacao_stopwords]
```

In [56]: 1 stopwords semacentro

```
[',',  
'',',  
'#',  
'$',  
'%',  
&',  
'''',  
'(','  
)','  
'*',  
'+',  
',',  
'-',  
'.',  
/,',  
/,',  
:,',  
<',  
=,',  
>',  
?',  
@',  
[',',  
\n]
```

```
In [57]: 1 resenha['tratamento_3'] = sem_acentos
2
3 frase_processada = list()
4 for opiniao in resenha['tratamento_3']:
5     nova_frase = list()
6     palavras_texto = token_pontuacao.tokenize(opiniao)
7     for palavra in palavras_texto:
8         if palavra not in pontuacao_stopwords:
9             nova_frase.append(palavra)
10    frase_processada.append(' '.join(nova_frase))
11
12 resenha['tratamento_3'] = frase_processada
```

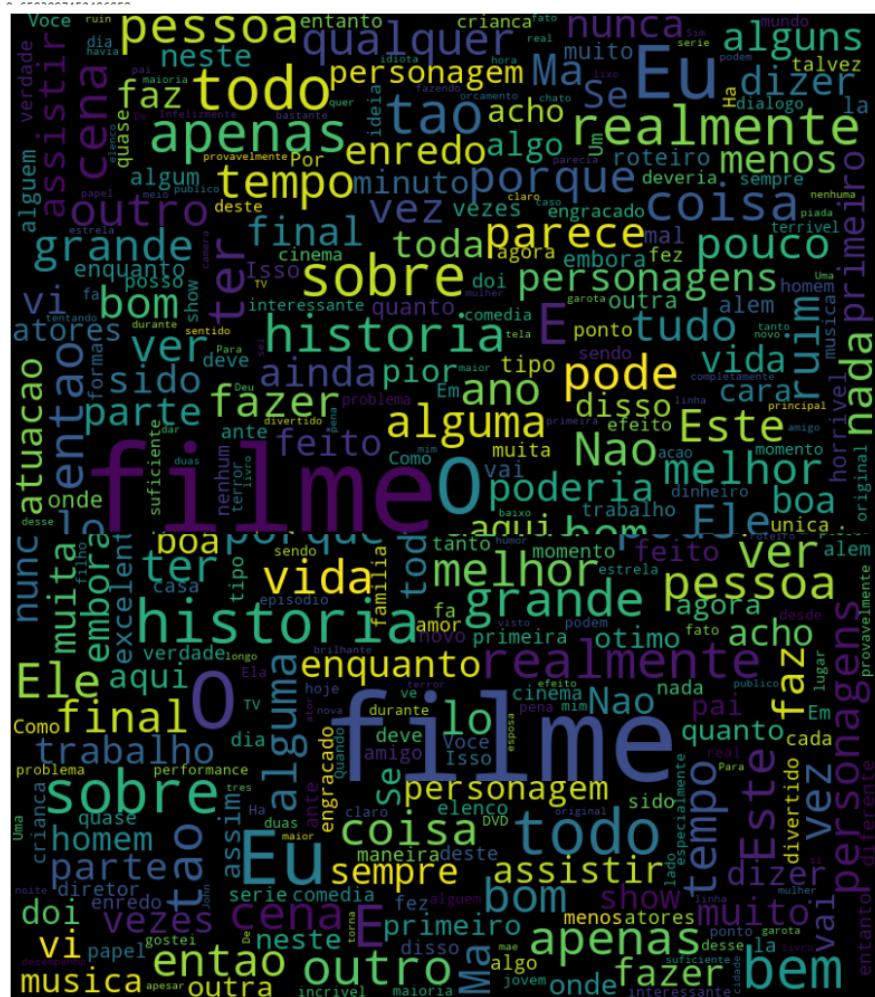
```
1 resenha.head()
```

id	text_en	text_pt	sentiment	classificacao	tratamento_1	tratamento_2	tratamento_3
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...	Mais vez Sr Costner arrumou filme tempo necess...	Mais vez Sr Costner arrumou filme tempo necess...
1 2	This is an example of why the majority of acti...	Este é um exemplo do motivo pelo qual a maior... dragged out a movie...	neg	0	Este exemplo motivo maioria filmes ação mesmos...	Este exemplo motivo maioria filmes ação mesmos...	Este exemplo motivo maioria filmes acao mesmos...
1 2	This is an example of why the majority of acti...	Este é um exemplo do motivo pelo qual a maior... dragged out a movie...	neg	0	Este exemplo motivo maioria filmes ação mesmos...	Este exemplo motivo maioria filmes ação mesmos...	Este exemplo motivo maioria filmes acao mesmos...
2 3	First of all I hate those moronic rappers, who...	Primeiro de tudo eu odeio esses raps imbecis, ...	neg	0	Primeiro tudo odeio raps imbecis, poderiam agir...	Primeiro tudo odeio raps imbecis poderiam agir...	Primeiro tudo odeio raps imbecis poderiam agir...
3 4	Not even the Beatles could write songs everyon...	Nem mesmo os Beatles puderam escrever músicas ...	neg	0	Nem Beatles puderam escrever músicas todos gos...	Nem Beatles puderam escrever músicas todos gos...	Nem Beatles puderam escrever musicas 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...	Filmes fotos latão palavra apropriada verdade ...	Filmes fotos latao palavra apropriada verdade ...

```
In [59]: 1 acuracia_tratamento3 = classificar_texto(resenha, "tratamento_3", "classificacao")
2 print(acuracia_tratamento3)
```

0.6887990295188031

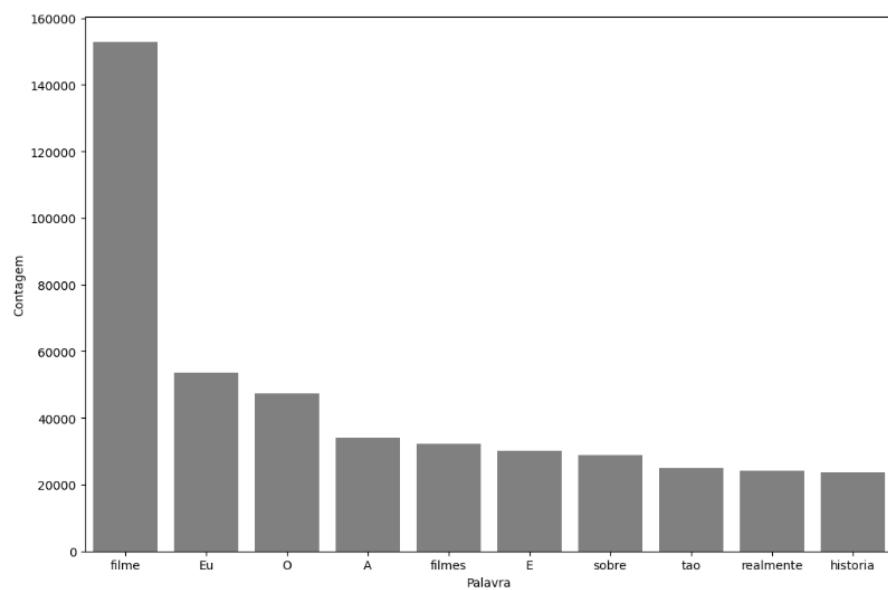
In [60]: 1 acuracia



In [63]:



```
In [63]: 1 pareto(resenha, "tratamento_3", 10)
```



```
In [64]: 1 frase_processada = list()
2 for opiniao in resenha['tratamento_3']:
3     nova_frase = list()
4     opiniao = opiniao.lower()
5     palavras_texto = token_pontuacao.tokenize(opiniao)
6     for palavra in palavras_texto:
7         if palavra not in palavras_irrelevantes:
8             nova_frase.append(palavra)
9     frase_processada.append(' '.join(nova_frase))
10
11 resenha['tratamento_4'] = frase_processada
```

In [65]: 1 resenha.head()

id	text_en	text_pt	sentiment	classificacao	tratamento_1	tratamento_2	tratamento_3	tratamento_4
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...	Mais vez Sr Costner arrumou filme tempo necess...	Mais vez Sr Costner arrumou filme tempo necess...	vez sr costner arrumou filme tempo necessario ...
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...	Este exemplo motivo maioria filmes ação mesmos...	Este exemplo motivo maioria filmes ação mesmos...	exemplo motivo maioria filmes acao mesmos gene...
2 3	First of all I hate those moronic rappers, who...	Primeiro de tudo eu odeio esses raps imbecis, ...	neg	0	Primeiro tudo odeio raps imbecis, poderiam agir...	Primeiro tudo odeio raps imbecis poderiam agir...	Primeiro tudo odeio raps imbecis poderiam agir...	primeiro tudo odeio raps imbecis poderiam agir...
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...	Nem Beatles puderam escrever músicas todos gos...	Nem Beatles puderam escrever músicas todos gos...	beatles pudaram escrever musicas todos gostass...
4 5	Brass pictures movies is not a fitting word fo...	Filmes de fotos de latão não é uma palavra an...	neg	0	Filmes fotos latão palavra apropriada eles, ve...	Filmes fotos latão palavra apropriada verdade ...	Filmes fotos latao palavra apropriada verdade ...	filmes fotos latao palavra apropriada verdade ...

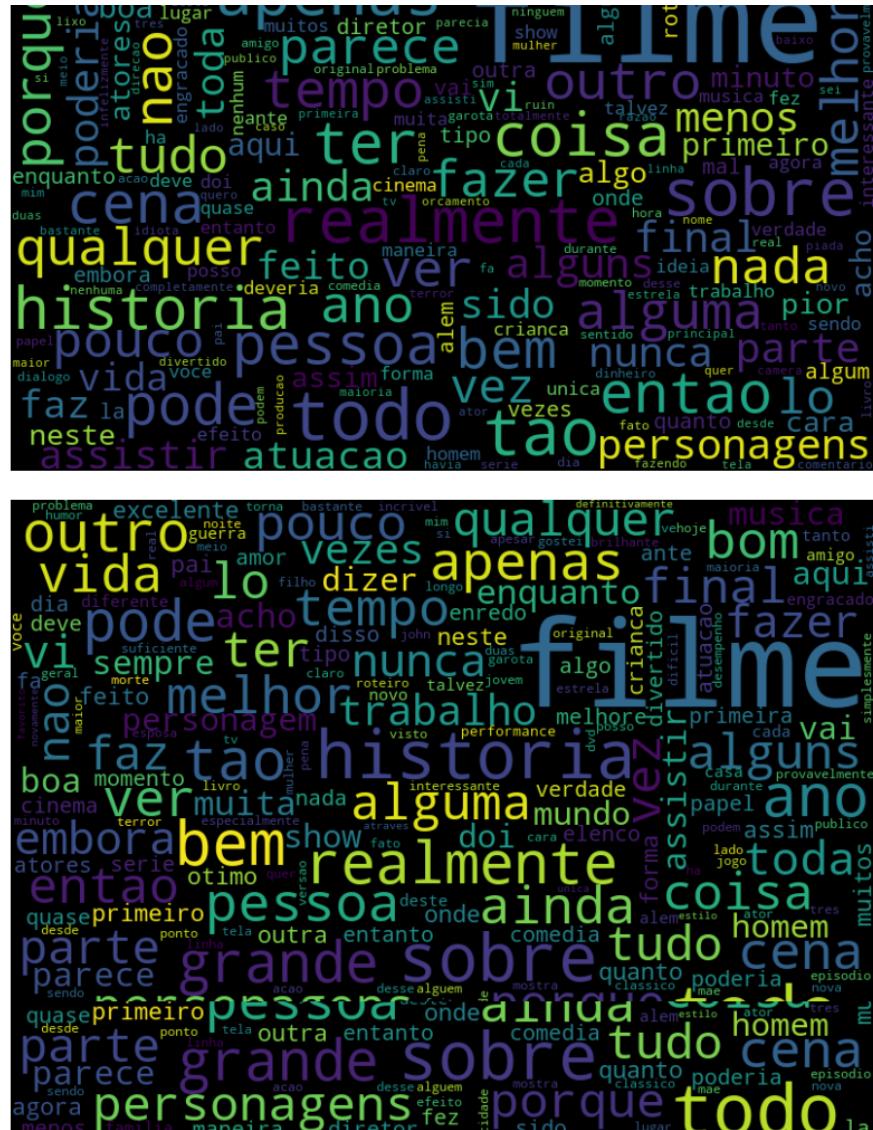
```
In [66]: 1 asuncia_tratamiento4 = clasifican_texto(posecha, "tratamiento 4", "clasificacio")
```

```
In [67]: 1 print(acuracia_tratamento3)
          2 print(acuracia_tratamento4)
```

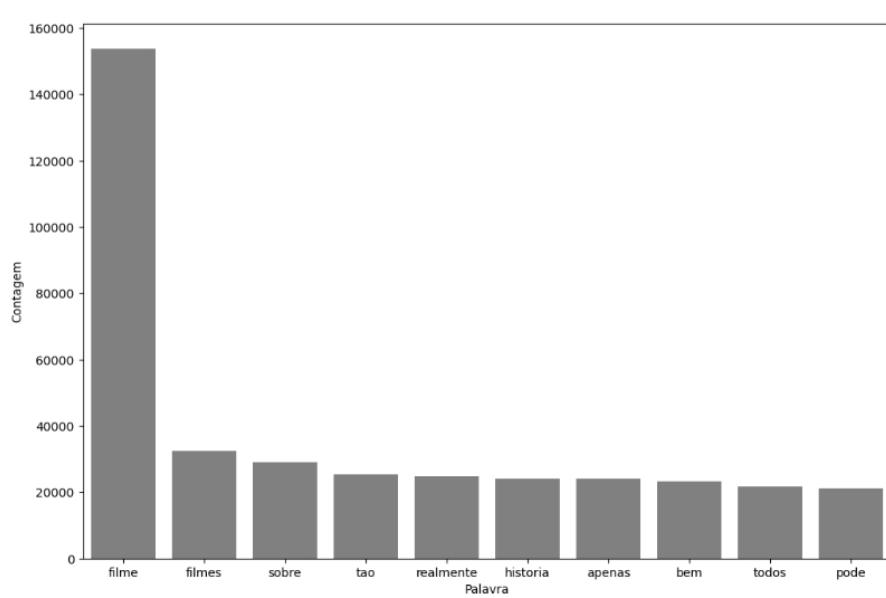
0.6887990295188031
0.6909017387788111

```
In [68]: 1 nuvem_palavras_neg(resenha, "tratamento_4")
```





```
In [70]: 1 pareto(resenha, "tratamento_4", 10)
```



```
In [71]: 1 stemmer = nltk.RSLPStemmer()
          2
          3 frase_processada = list()
          4 for opiniao in resenha['tratamento_4']:
          5     frase_processada.append(stemmer.stem(opiniao))
          6
          7 print(frase_processada)
```

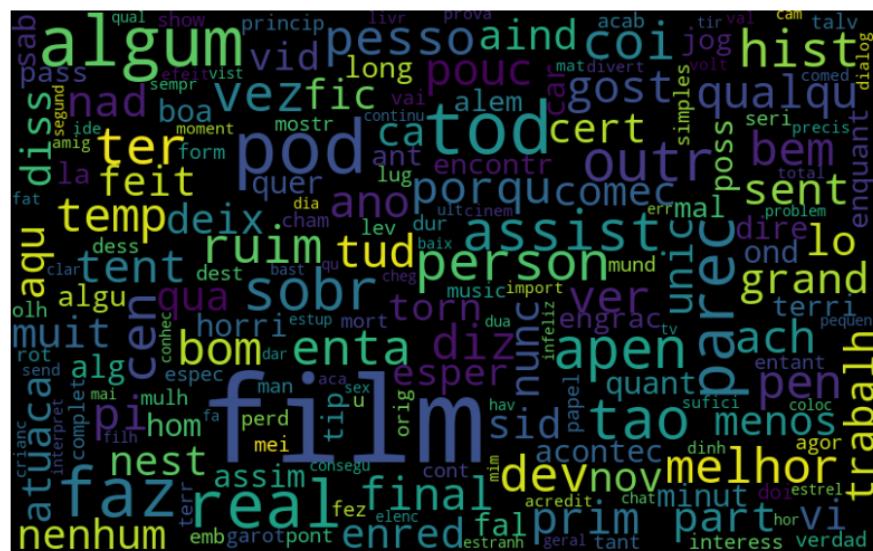
```
nova_frase = []
palavras_texto = token_pontuacao.tokenize(opiniao)
for palavra in palavras_texto:
    if palavra not in stopwords_sem_acento:
        nova_frase.append(stemmer.stem(palavra))
frase_processada.append(' '.join(nova_frase))

senha['tratamento_5'] = frase_processada
```

```
In [72]: 1 acuracia_tratamento5 = classificar_texto(resenha, "tratamento_5", "classificacao")
          2 print(acuracia_tratamento3)
          3 print(acuracia_tratamento4)
          4 print(acuracia_tratamento5)
```

0.6887990295188031
0.6909017387788111
0.6954306510311363

```
In [73]: 1 nuvem_palavras_neg(resenha, "tratamento_5")
```

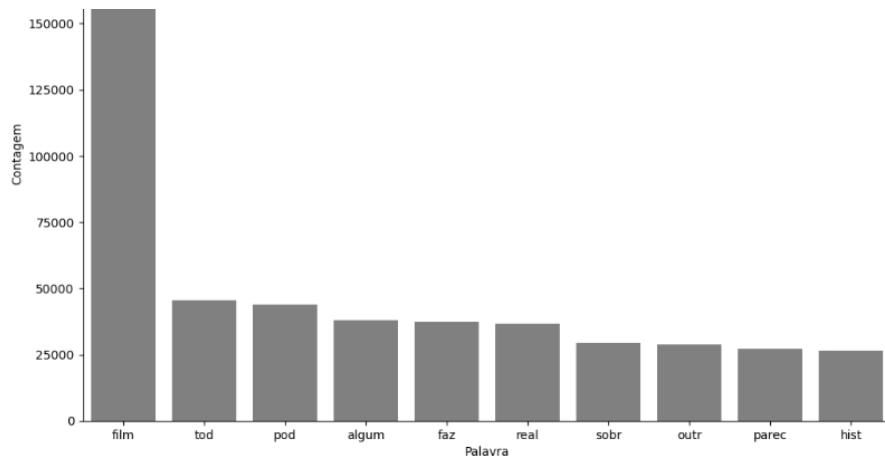


In [74]: 1 nuvem_palavras_pos(resenha, "tratamento 5")



```
In [75]: 1 pareto(resepha, "tratamento_5", 10)
```





```
In [77]: 1 from sklearn.feature_extraction.text import TfidfVectorizer
2
3 tfidf = TfidfVectorizer(lowercase = False, max_features = 50)
4 tfidf_bruto = tfidf.fit_transform(resenha['text_pt'])
5 treino, teste, classe_treino, classe_teste = train_test_split(tfidf_bruto,
6                                         resenha['classificacao'],
7                                         random_state = 42)
8 regressao_logistica.fit(treino, classe_treino)
9 acuracia_tfidf_bruto = regressao_logistica.score(teste, classe_teste)
10 print(acuracia_tfidf_bruto)
```

0.6600889607763849

```
In [78]: 1 tfidf_tratados = tfidf.fit_transform(resenha['tratamento_5'])
2 treino, teste, classe_treino, classe_teste = train_test_split(tfidf_tratados,
3                                         resenha['classificacao'],
4                                         random_state = 42)
5 regressao_logistica.fit(treino, classe_treino)
6 acuracia_tfidf_tratados = regressao_logistica.score(teste, classe_teste)
7 print(acuracia_tfidf_tratados)
```

0.6963202587949858

```
In [80]: 1 from nltk import ngrams
2
3 tfidf = TfidfVectorizer(lowercase = False, ngram_range = (1, 2))
4 vetor_tfidf = tfidf.fit_transform(resenha['tratamento_5'])
5 treino, teste, classe_treino, classe_teste = train_test_split(vetor_tfidf,
6                                         resenha['classificacao'],
7                                         random_state = 42)
8 regressao_logistica.fit(treino, classe_treino)
9 acuracia_tfidf_ngrams = regressao_logistica.score(teste, classe_teste)
10 print(acuracia_tfidf_ngrams)
```

0.8856449656287909

```
In [81]: 1 tfidf = TfidfVectorizer(lowercase = False)
2 vetor_tfidf = tfidf.fit_transform(resenha['tratamento_5'])
3 treino, teste, classe_treino, classe_teste = train_test_split(vetor_tfidf,
4                                         resenha['classificacao'],
5                                         random_state = 42)
6 regressao_logistica.fit(treino, classe_treino)
7 acuracia_tfidf = regressao_logistica.score(teste, classe_teste)
8 print(acuracia_tfidf)
```

0.8847553578649414

```
In [86]: 1 pesos = pd.DataFrame(regressao_logistica.coef_[0].T,
2                             index = tfidf.get_feature_names_out()
3                             )
4 pesos.nlargest(50,0)
```

otim	0
otim	8.308764

```
excel    7.992873
perfeit   6.548340
favorit   5.781070
maravilh  5.291923
incre     5.052462
hilari    4.653921
divert    4.552039
ador      4.409966
brilh     4.268854
hoj       4.208786
recom     4.193041
agrada    4.142481
definitiv 4.140507
soberb    3.571461
surpreend 3.568743
maravilhos 3.496956
ame       3.493814
bom      3.485206
class     3.461470
aind     3.449482
impression 3.391051
comov     3.371855
lind      3.179258
dvd       3.139152
mant     3.099516
fort      3.073772
joi       3.064044
fascin    3.039393
apreci    2.979625
subestim  2.962781
sutil     2.954731
gost      2.949283
melhor    2.942800
espec     2.852008
difer     2.845665
refresc   2.836689
sempr     2.794817
human     2.714639
am        2.708111
fantas    2.675713
emocion   2.626685
obr       2.615441
intens    2.613389
pequen   2.596191
inesper   2.591139
poder    2.587926
surpres   2.577427
performanc 2.573411
bem      2.550362
```

```
In [88]: 1 pesos.nsmallest(10,0)
```

```
          0
pi      -9.185322
ruim   -9.179967
horri  -8.761630
terri  -7.195886
chat   -6.921456
nad    -6.032698
infeliz -5.297961
decepca -4.984679
ridicul -4.971187
parec   -4.939297
```

```
In [ ]: 1
```