bimusic-3

April 17, 2024

[ ]:
```
#!pip install scikit-learn
```

[ ]:
```
#!cd drive/MyDrive/Colab\ Notebooks
```

[ ]:
```
#!sudo apt update
```

[ ]:
```
#!sudo apt-get install texlive-full
```

[ ]:
```
#!jupyter nbconvert --to pdf drive/MyDrive/Colab\ Notebooks/BiMusic.ipynb;
```

[ ]:
```
import os
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

# Função para ler o arquivo metadata.txt e obter o nome do artista e o gênero␣
 ↪musical
def ler_metadata(metadata_file):
    with open(metadata_file, 'r', encoding='utf-8') as f:
        artist_name = f.readline().strip()
        genre = f.readline().strip()
    return artist_name, genre

# Função para ler o conteúdo do arquivo .txt (letra da música)
def ler_letra(letra_file):
    with open(letra_file, 'r', encoding='utf-8') as f:
        letra = f.read()
    return letra

# Função para ler o conteúdo do arquivo .html (cifra da música)
def ler_cifra(cifra_file):
    with open(cifra_file, 'r', encoding='utf-8') as f:
        cifra = f.read()
    return cifra

# Lista para armazenar os dados de cada música
data = []
```

```
# Percorrer recursivamente as pastas de artistas
for root, dirs, files in os.walk('/content/drive/MyDrive/conteudo_artistas/'):

    if '.config' in dirs:
        dirs.remove('.config')
    if '.ipynb_checkpoints' in dirs:
        dirs.remove('.ipynb_checkpoints')
    if 'sample_data' in dirs:
        dirs.remove('sample_data')

    artist_name, genre, letra, cifra = ["", "", "", ""]

    if os.path.exists(os.path.join(root,'metadata.txt')):
        artist_name, genre = ler_metadata(os.path.join(root,'metadata.txt'))

    for file in files:
        m_file = os.path.join(root, file)

        if '.html' in file:
            cifra = ler_cifra(m_file)
        else:
            letra = ler_letra(m_file)

        data.append({'Artist': artist_name, 'Genre': genre, 'Lyrics': letra,␣
 ↪'Chords': cifra})

# Criar um DataFrame com os dados coletados
df = pd.DataFrame(data)

# Visualizar o DataFrame
print(df.head())
```

```
     Artist Genre                                              Lyrics  \
0  Zoe Wees   Pop  Control\n \nEarly in the morning \nI still get…
1  Zoe Wees   Pop  Control\n \nEarly in the morning \nI still get…
2  Zoe Wees   Pop  Girls Like Us\n \nIt's hard for girls like us …
3  Zoe Wees   Pop  Girls Like Us\n \nIt's hard for girls like us …
4  Zoe Wees   Pop  Hold Me Like You Used To\n \nCan you feel that…

                                              Chords
0
1  <html><h1>Control<h1><div><pre>[Intro] <b>Am</…
2  <html><h1>Control<h1><div><pre>[Intro] <b>Am</…
3  <html><h1>Girls Like Us<h1><div><pre>         …
4  <html><h1>Girls Like Us<h1><div><pre>         …
```

[ ]:
```
df.to_csv('bi_music.csv', index=False)
```

[3]:
```
popularidade_generos = df['Genre'].value_counts()

# Visualizar os resultados
print(popularidade_generos)
```

```
Genre
Gospel/Religioso       3620
Pop                    1566
                       1059
Hip Hop/Rap             926
Pop Rock                845
Rock and Roll           796
Sertanejo               737
MPB                     696
Rock Alternativo        568
Alternativo / Indie     561
Heavy Metal             376
R&B                     359
Pagode                  353
Trap                    277
Infantil                234
K-Pop                   226
Hard Rock               222
Forró                   220
Folk                    196
Arrocha                 182
Nativista               168
Reggae                  149
Soul                    138
Eletrônica              132
Funk                    128
J-Pop/J-Rock            116
Piseiro                 102
Romântico               100
Jazz                     93
Rock Progressivo         81
Grunge                   81
Emocore                  79
Surf Music               75
Punk Rock                75
Bossa Nova               73
Reggaeton                70
Country                  68
Samba                    68
Axé                      64
Marchas/Hinos            57
Regional                 56
```

```
Corridos                50
World Music             44
Afrobeats               44
Hardcore                41
Jovem Guarda            40
Industrial              39
Soft Rock               37
New Wave                37
Brega                   31
Cuarteto                24
Clássico                24
Name: count, dtype: int64
```

[4]:
```
df.groupby(['Artist', 'Genre']).size()
```

[4]:
```
Artist                                          Genre
Diddy (P. Diddy / Puff Daddy / Brother Love)    Hip Hop/Rap            22
Eagles                                          Rock and Roll          41
Grupo Pixote                                    Pagode                 36
Grupo Revelação                                 Pagode                 39
LUDMILLA                                        Funk                   31
                                                                       ..
Zélia Duncan                                    MPB                    35
wave to earth                                   Alternativo / Indie    23
xcho                                                                    2
xikers                                          K-Pop                  21
zandros                                                                 4
Length: 527, dtype: int64
```

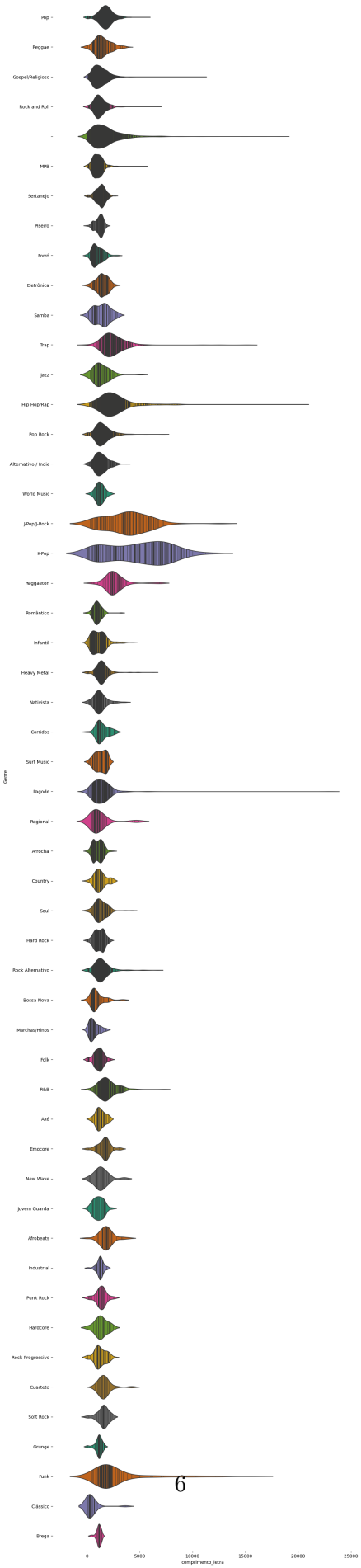
[5]:
```
df['comprimento_letra'] = df['Lyrics'].str.len()
df
```

[5]:
```
          Artist      Genre                                             Lyrics  \
0      Zoe Wees        Pop  Control\n \nEarly in the morning \nI still get…
1      Zoe Wees        Pop  Control\n \nEarly in the morning \nI still get…
2      Zoe Wees        Pop  Girls Like Us\n \nIt's hard for girls like us …
3      Zoe Wees        Pop  Girls Like Us\n \nIt's hard for girls like us …
4      Zoe Wees        Pop  Hold Me Like You Used To\n \nCan you feel that…
…         …         …                                                  …
16398  Leonardo  Sertanejo  Mano\n \nMano, você é meu sangue, mais que um …
16399  Leonardo  Sertanejo  Lembranças\n \nJá faz tanto tempo que eu deixe…
16400  Leonardo  Sertanejo  Lembranças\n \nJá faz tanto tempo que eu deixe…
16401  Leonardo  Sertanejo  Choro\n \nEu não choro porque te amo \nE nem p…
16402  Leonardo  Sertanejo  Choro\n \nEu não choro porque te amo \nE nem p…

                                                  Chords  comprimento_letra
0                                                                       1423
```

```
1       <html><h1>Control<h1><div><pre>[Intro] <b>Am</…              1423
2       <html><h1>Control<h1><div><pre>[Intro] <b>Am</…              1846
3       <html><h1>Girls Like Us<h1><div><pre>         …              1846
4       <html><h1>Girls Like Us<h1><div><pre>         …              1829
…                                                   …               …
16398   <html><h1>O Quanto Nosso Amor Valeu<h1><div><p…              1346
16399   <html><h1>O Quanto Nosso Amor Valeu<h1><div><p…               859
16400   <html><h1>Lembranças<h1><div><pre>[Intro] <b>B…               859
16401   <html><h1>Lembranças<h1><div><pre>[Intro] <b>B…               972
16402   <html><h1>Choro<h1><div><pre><b>D</b>         …               972

[16403 rows x 5 columns]
```

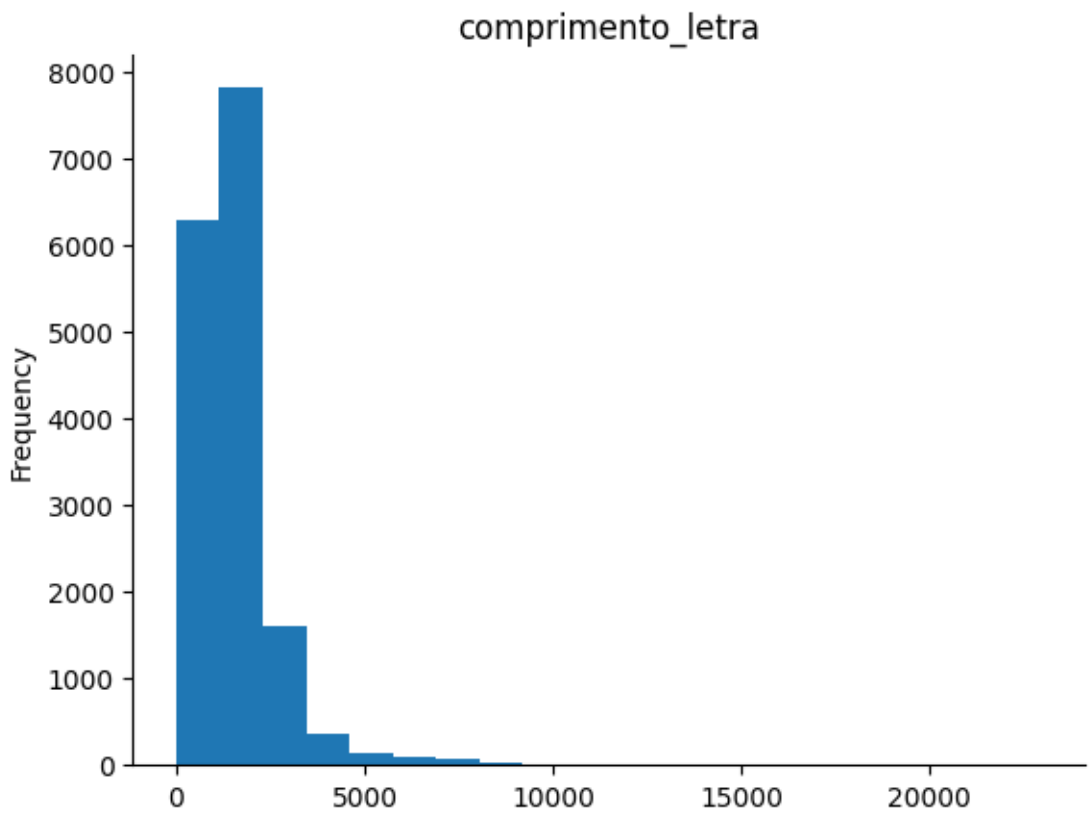
[6]:
```
figsize = (12, 1.2 * len(df['Genre'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(df, x='comprimento_letra', y='Genre', inner='stick',␣
 ↪palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
<ipython-input-6-7e7c5111a10c>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.violinplot(df, x='comprimento_letra', y='Genre', inner='stick',
palette='Dark2')
```


Pop
Reggae
Gospel/Religioso
Rock and Roll
-
MPB
Sertanejo
Piseiro
Forró
Eletrônica
Samba
Trap
Jazz
Hip Hop/Rap
Pop Rock
Alternativo / Indie
World Music
J-Pop/J-Rock
K-Pop
Reggaeton
Romântico
Infantil
Heavy Metal
Nativista
Corridos
Surf Music
Pagode
Regional
Arrocha
Country
Soul
Hard Rock
Rock Alternativo
Bossa Nova
Marchas/Hinos
Folk
R&B
Axé
Emocore
New Wave
Jovem Guarda
Afrobeats
Industrial
Punk Rock
Hardcore
Rock Progressivo
Cuarteto
Soft Rock
Grunge
Funk
Clássico
Brega
Genre
0
5000
10000
15000
20000
25000
comprimento_letra

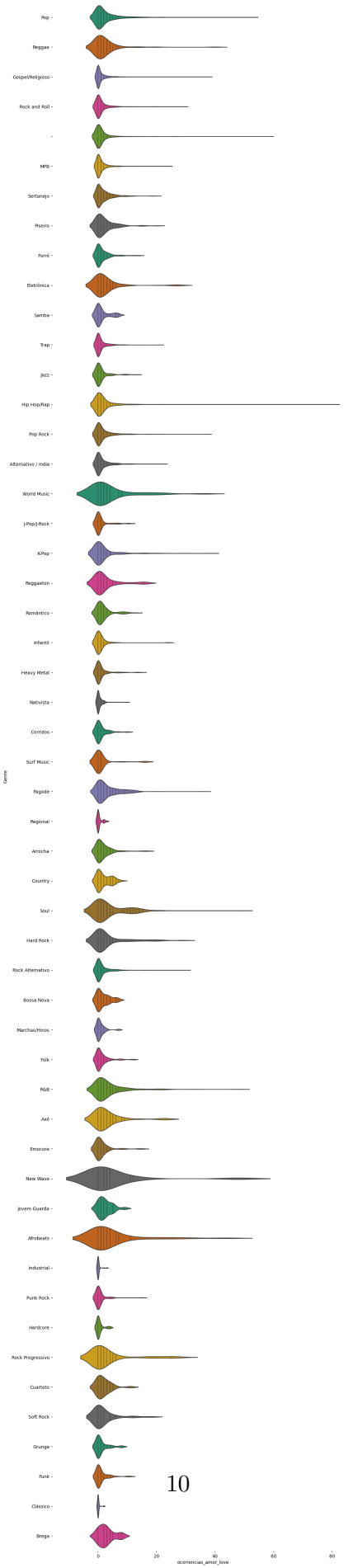
[7]:
```
figsize = (12, 1.2 * len(df['Artist'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(df, x='comprimento_letra', y='Artist', inner='stick',␣
 ↪palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
<ipython-input-7-92ecfcd91a3a>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.violinplot(df, x='comprimento_letra', y='Artist', inner='stick',
palette='Dark2')
/usr/local/lib/python3.10/dist-packages/IPython/core/events.py:89: UserWarning:
Glyph 24352 (\N{CJK UNIFIED IDEOGRAPH-5F20}) missing from current font.
  func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/IPython/core/events.py:89: UserWarning:
Glyph 27938 (\N{CJK UNIFIED IDEOGRAPH-6D22}) missing from current font.
  func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/IPython/core/events.py:89: UserWarning:
Glyph 35946 (\N{CJK UNIFIED IDEOGRAPH-8C6A}) missing from current font.
  func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151:
UserWarning: Glyph 24352 (\N{CJK UNIFIED IDEOGRAPH-5F20}) missing from current
font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151:
UserWarning: Glyph 27938 (\N{CJK UNIFIED IDEOGRAPH-6D22}) missing from current
font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151:
UserWarning: Glyph 35946 (\N{CJK UNIFIED IDEOGRAPH-8C6A}) missing from current
font.
  fig.canvas.print_figure(bytes_io, **kw)
```



[9]:
```
from matplotlib import pyplot as plt
import seaborn as sns

# Contabilizar a quantidade de vezes que as palavras "amor" ou "love" aparecem␣
 ↪em cada música
df['ocorrencias_amor_love'] = df['Lyrics'].str.count('amor|love')

# Definir o tamanho da figura
figsize = (12, 1.2 * len(df['Genre'].unique()))

# Criar o gráfico de violino
plt.figure(figsize=figsize)
sns.violinplot(data=df, x='ocorrencias_amor_love', y='Genre', inner='stick',␣
 ↪palette='Dark2')

# Remover bordas do gráfico
sns.despine(top=True, right=True, bottom=True, left=True)

# Mostrar o gráfico
plt.show()
```

```
<ipython-input-9-b1268b6a107f>:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.violinplot(data=df, x='ocorrencias_amor_love', y='Genre', inner='stick',
palette='Dark2')
```


Pop
Reggae
Gospel/Religioso
Rock and Roll
-
MPB
Sertanejo
Piseiro
Forró
Eletrônica
Samba
Trap
Jazz
Hip Hop/Rap
Pop Rock
Alternativo / Indie
World Music
J-Pop/J-Rock
K-Pop
Reggaeton
Romântico
Infantil
Heavy Metal
Nativista
Corridos
Surf Music
Pagode
Regional
Arrocha
Country
Soul
Hard Rock
Rock Alternativo
Bossa Nova
Marchas/Hinos
Folk
R&B
Axé
Emocore
New Wave
Jovem Guarda
Afrobeats
Industrial
Punk Rock
Hardcore
Rock Progressivo
Cuarteto
Soft Rock
Grunge
Funk
Clássico
Brega
Genre
0
20
40
60
80
ocorrencias_amor_love

[10]:
```
from sklearn.feature_extraction.text import CountVectorizer

# Criar uma matriz de frequência de palavras
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df['Lyrics'])

# Obter as palavras mais comuns
palavras_comuns = pd.DataFrame(X.toarray(), columns=vectorizer.
 ↪get_feature_names_out()).sum().sort_values(ascending=False)

# Visualizar as palavras mais comuns
print(palavras_comuns.head(10))
```

```
you     88759
que     77810
span    75272
me      75251
the     67647
eu      62050
de      58216
não     47436
to      40618
it      40166
dtype: int64
```

[11]:
```
import re

# Função para contar palavras entre as tags <b> e </b>
def contar_palavras_b(texto):
    # Encontrar todas as palavras entre as tags <b> e </b> usando expressão␣
 ↪regular
    palavras = re.findall(r'<b>(.*?)</b>', texto)
    # Contar a ocorrência de cada palavra e retornar o resultado como um␣
 ↪dicionário
    contador = {}
    for palavra in palavras:
        contador[palavra] = contador.get(palavra, 0) + 1
    return contador

# Aplicar a função contar_palavras_b à coluna "chord" e expandir o resultado em␣
 ↪um DataFrame
contagem_palavras = df['Chords'].apply(contar_palavras_b).apply(pd.Series).
 ↪fillna(0).astype(int)
```

```
# Exibir o DataFrame resultante
print(contagem_palavras)
```

```
       Am   F   C   G  F7M  Dm  Em   D   A  Cmaj7  …  C7/F  Bb7M(9)/D  \
0       0   0   0   0    0   0   0   0   0      0  …     0          0
1      27  27  26   6    0   0   0   0   0      0  …     0          0
2      27  27  26   6    0   0   0   0   0      0  …     0          0
3      19  19  19  19    0   0   0   0   0      0  …     0          0
4      19  19  19  19    0   0   0   0   0      0  …     0          0
…      ..  ..  ..  ..  …  ..  ..  ..  ..    …  …   …       …
16398   5   0  16  16    0   0   7  21   0      0  …     0          0
16399   5   0  16  16    0   0   7  21   0      0  …     0          0
16400   0   0  11  11    0   0  11  11  11      0  …     0          0
16401   0   0  11  11    0   0  11  11  11      0  …     0          0
16402   0   0   0  13    0   0   0  10  26      0  …     0          0

       F7/9/13  A7(5+)/C#  B/9  C7/B  F11M  \
0            0          0    0     0     0
1            0          0    0     0     0
2            0          0    0     0     0
3            0          0    0     0     0
4            0          0    0     0     0
…          …        …  …    …   …
16398        0          0    0     0     0
16399        0          0    0     0     0
16400        0          0    0     0     0
16401        0          0    0     0     0
16402        0          0    0     0     0

       Eb(19)-15-17-15-17-15----------  Am/Am  E(7)
0                                    0      0     0
1                                    0      0     0
2                                    0      0     0
3                                    0      0     0
4                                    0      0     0
…                                  …    …   …
16398                                0      0     0
16399                                0      0     0
16400                                0      0     0
16401                                0      0     0
16402                                0      0     0

[16403 rows x 2039 columns]
```

[ ]:
```
import matplotlib.pyplot as plt

# Obter o top 20 das contagens totais de palavras
```

```
top_20_contagem_total = contagem_palavras.sort_values(by=[] ascending=False).
 ↪head(20)

# Plotar um gráfico de barras do top 20 das contagens totais de palavras
plt.figure(figsize=(10, 6))
top_20_contagem_total.plot(kind='bar', color='skyblue')
plt.title('Top 20 da Contagem Total de acordes')
plt.xlabel('Acorde')
plt.ylabel('Contagem')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

[ ]:
```
import nltk
nltk.download('vader_lexicon')
```

[ ]:
```
import os
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from nltk.sentiment.vader import SentimentIntensityAnalyzer


# Função para ler todos os arquivos de texto em um diretório
def ler_arquivos_texto(diretorio):
    textos = []
    for arquivo in os.listdir(diretorio):
        print(arquivo)
        if arquivo.endswith(".txt"):
            with open(os.path.join(diretorio, arquivo), "r", encoding="utf-8")␣
 ↪as f:
                sia = SentimentIntensityAnalyzer()
                t = f.read()
                sentiment_score = sia.polarity_scores(t)
                print(sentiment_score)
                textos.append(t)

    return textos

# Diretório onde os arquivos de texto estão localizados
diretorio = "/content/drive/MyDrive/conteudo_artistas/aerosmith/"

# Ler os arquivos de texto
textos = ler_arquivos_texto(diretorio)

# Vetorização dos textos usando TF-IDF
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(textos)
```

```
# Execução do algoritmo KMeans
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)

# Exibir os clusters e suas palavras-chave
print("Palavras-chave dos clusters:")
order_centroids = kmeans.cluster_centers_.argsort()[:, ::-1]
terms = vectorizer.get_feature_names_out()
for i in range(3):  # Altere o número de clusters conforme necessário
    print("Cluster %d:" % i)
    for ind in order_centroids[i, :10]:  # Altere o número de palavras-chave␣
 ↪exibidas conforme necessário
        print(' %s' % terms[ind])
```

[ ]:
```
import os
import pandas as pd
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Função para ler todos os arquivos de texto em um diretório
def ler_arquivos_texto(diretorio):
    textos = []
    for arquivo in os.listdir(diretorio):
        if arquivo.endswith(".txt"):
            with open(os.path.join(diretorio, arquivo), "r", encoding="utf-8")␣
 ↪as f:
                sia = SentimentIntensityAnalyzer()
                t = f.read()
                sentiment_score = sia.polarity_scores(t)
                textos.append((t, sentiment_score))

    return textos

# Diretório onde os arquivos de texto estão localizados
diretorio = "/content/drive/MyDrive/conteudo_artistas/queen/"

# Ler os arquivos de texto
textos = ler_arquivos_texto(diretorio)

# Vetorização dos textos usando TF-IDF
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform([texto[0] for texto in textos])

# Execução do algoritmo KMeans
```

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)

# Obter os rótulos dos clusters
labels = kmeans.labels_

# Criar DataFrame com os dados dos textos e seus rótulos de cluster
df = pd.DataFrame({
    'Texto': [texto[0] for texto in textos],
    'Sentiment_Score': [texto[1]['compound'] for texto in textos],
    'Cluster': labels
})

# Exibir os clusters e suas palavras-chave
print("Palavras-chave dos clusters:")
order_centroids = kmeans.cluster_centers_.argsort()[:, ::-1]
terms = vectorizer.get_feature_names_out()
for i in range(3):  # Altere o número de clusters conforme necessário
    print("Cluster %d:" % i)
    for ind in order_centroids[i, :10]:  # Altere o número de palavras-chave␣
 ↪exibidas conforme necessário
        print(' %s' % terms[ind])

# Plotar gráfico de dispersão dos textos com os sentimentos coloridos por␣
 ↪cluster
sns.scatterplot(data=df, x='Texto', y='Sentiment_Score', hue='Cluster',␣
 ↪palette='Dark2')
plt.title('Gráfico de Dispersão dos Textos com Sentimento Colorido por Cluster')
plt.xlabel('Texto')
plt.ylabel('Sentiment Score')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```