

✓ Exercícios - Aula 1

✓ 1) Dado o dataset de produtos [1]:

[1] - <https://dados-ml-pln.s3-sa-east-1.amazonaws.com/produtos.csv>

```
import pandas as pd
```

```
df = pd.read_csv(  
    "https://dados-ml-pln.s3-sa-east-1.amazonaws.com/produtos.csv",  
    delimiter=";",  
    encoding='utf-8' )
```

```
# Execute uma vez se ainda não tiver essas libs  
%pip install wordcloud unicode
```

```
➡ Requirement already satisfied: wordcloud in /usr/local/lib/python3.11/dist-packages (1.9.4)  
Requirement already satisfied: unicode in /usr/local/lib/python3.11/dist-packages (1.4.0)  
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.11/dist-packages (from wordcloud)  
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from wordcloud)  
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from wordcloud)  
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib)<br>Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib)<br>Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib)<br>Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib)<br>Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib)<br>Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib)<br>Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib)<br>Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from matplotlib)
```

✓ 1 Importação de Bibliotecas e Configuração do Ambiente

Importamos as bibliotecas essenciais para análise de dados, visualizações e processamento de texto. Também configuramos o estilo padrão para gráficos.

```
# Execute esta célula para instalar todas as dependências necessárias  
%pip install seaborn wordcloud unicode
```

```
➡ Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)  
Requirement already satisfied: wordcloud in /usr/local/lib/python3.11/dist-packages (1.9.4)  
Requirement already satisfied: unicode in /usr/local/lib/python3.11/dist-packages (1.4.0)  
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.11/dist-packages (from seaborn)<br>Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.11/dist-packages (from seaborn)<br>Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.11/dist-packages (from seaborn)<br>Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from matplotlib)<br>Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib)<br>Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib)<br>Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib)
```

```
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from  
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from  
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from  
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages  
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pa  
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from  
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from pythor
```

```
# Bibliotecas de análise e visualização
```

```
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns
```

```
# Bibliotecas de NLP e manipulação de texto
```

```
from collections import Counter  
from wordcloud import WordCloud  
import unidecode  
import re
```

```
# Estilo de gráficos
```

```
sns.set(style="whitegrid")  
plt.rcParams["figure.figsize"] = (12, 6)
```

2 Carregamento do Dataset de Produtos

Nesta etapa, carregamos o arquivo CSV diretamente da URL disponibilizada e armazenamos os dados em um DataFrame `df`. Em seguida, visualizamos as primeiras linhas para ter uma noção da estrutura da base.

✖️ Correção do Carregamento: Delimitador Semicolon

O arquivo CSV parece estar usando ponto e vírgula (;) como separador. Vamos indicar isso explicitamente ao `read_csv` para corrigir o erro de parsing.

```
# URL do dataset  
url = "https://dados-ml-pln.s3-sa-east-1.amazonaws.com/produtos.csv"  
  
# Forçando o separador como ponto e vírgula  
df = pd.read_csv(url, sep=';')  
  
# Visualização inicial  
df.head()
```



	nome	descricao	categoria
0	O Hobbit - 7ª Ed. 2013	Produto NovoBilbo Bolseiro é um hobbit que lev...	livro
1	Livro - It A Coisa - Stephen King	Produto NovoDurante as férias escolares de 195...	livro
2	Box As Crônicas De Gelo E Fogo Pocket 5 Li...	Produto NovoTodo o reino de Westeros ao alcanc...	livro
3	Box Harry Potter	Produto Novo e Físico A série Harry Potter ch...	livro
4	Livro Origem - Dan Brown	Produto NovoDe Onde Viemos? Para Onde Vamos? R...	livro

3 Estrutura e Informações do Dataset

Vamos examinar a estrutura do DataFrame carregado: colunas, tipos de dados, quantidade de registros e se há duplicatas aparentes. Isso ajuda a orientar as transformações futuras.

```
# Dimensão do dataset
print(f"📊 Formato: {df.shape[0]} linhas x {df.shape[1]} colunas")

# Tipos de dados e contagem de não nulos
print("\n🔍 Info:")
df.info()

# Visualizar nomes de colunas
print("\n📄 Colunas disponíveis:")
print(df.columns.tolist())
```



📊 Formato: 4080 linhas x 3 colunas

```
🔍 Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4080 entries, 0 to 4079
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  -
0    nome        4080 non-null   object
1    descricao   2916 non-null   object
2    categoria   4080 non-null   object
dtypes: object(3)
memory usage: 95.8+ KB
```

```
📄 Colunas disponíveis:
['nome', 'descricao', 'categoria']
```

4 Análise de Valores Nulos

Verificamos agora a quantidade e o percentual de valores ausentes em cada coluna. Isso orienta decisões sobre limpeza dos dados e eliminação de registros.

```
# Contagem absoluta e percentual de nulos
null_counts = df.isnull().sum()
null_percent = df.isnull().mean() * 100

# Exibição
print("🔍 Valores Nulos por Coluna:")
display(pd.DataFrame({
    'Nulos': null_counts,
    '%': null_percent.round(2)
}).sort_values(by='%', ascending=False))
```

🔄 🔍 Valores Nulos por Coluna:

	Nulos	%
descricao	1164	28.53
nome	0	0.00
categoria	0	0.00

1.1. Analise o % de valores nulos no dataset

1164 nulos na coluna descrição (28,53%)

5 Remoção de Registros com Valores Nulos

Removemos todas as linhas que contêm valores ausentes em qualquer coluna. Essa decisão é válida neste caso por se tratar de uma base textual e com foco em NLP.

```
# Remoção de registros com valores nulos
df_clean = df.dropna().reset_index(drop=True)

# Verificando o novo tamanho do dataset
print(f"✅ Linhas restantes após remoção: {df_clean.shape[0]} de {df.shape[0]} originais")
```

🔄 ✅ Linhas restantes após remoção: 2916 de 4080 originais

6 Distribuição das Categorias

Nesta etapa, analisamos a distribuição da coluna `categoria`, que representa a classificação dos produtos. Vamos visualizar as categorias mais frequentes em forma tabular e gráfica.

```
# Contagem de categorias
categoria_counts = df_clean['categoria'].value_counts()
```

```
# Tabela
print("🇧🇷 Frequência por categoria:")
display(categoria_counts)
```

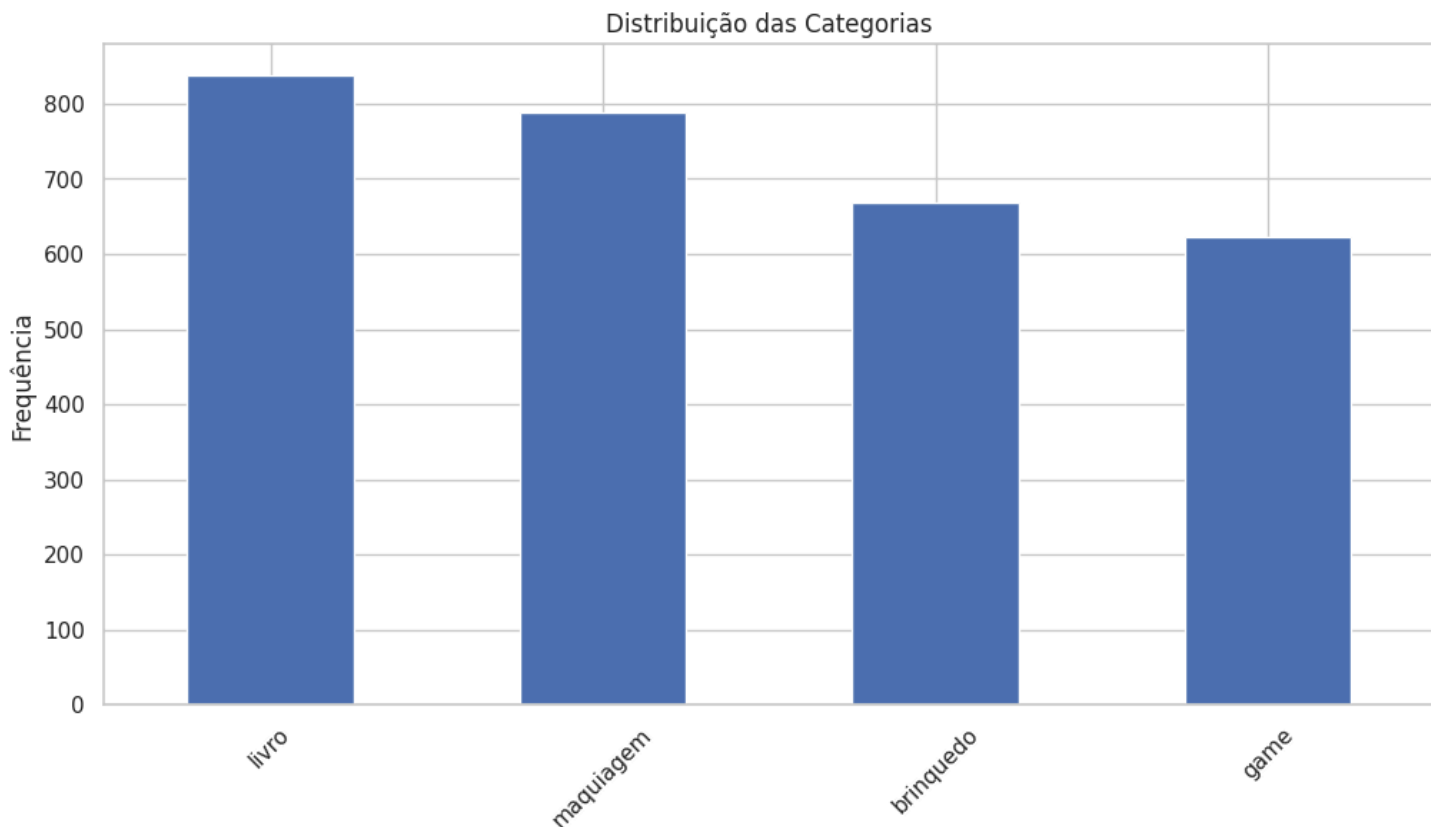
```
# Gráfico
categoria_counts.plot(kind='bar', title='Distribuição das Categorias')
plt.ylabel("Frequência")
plt.xticks(rotation=45)
plt.show()
```



🇧🇷 Frequência por categoria:

count	
categoria	
livro	838
maquiagem	788
brinquedo	668
game	622


dtype: int64



Combinamos as colunas `nome` e `descricao` para formar uma nova coluna `texto`, que será usada para tokenização, contagem de palavras e geração da nuvem de palavras.

```
# Criação da coluna "texto"
df_clean['texto'] = df_clean['nome'].astype(str) + " " + df_clean['descricao'].astype(str)

# Exibição de amostras
df_clean[['nome', 'descricao', 'texto']].head()
```



	nome	descricao	texto
0	O Hobbit - 7ª Ed. 2013	Produto NovoBilbo Bolseiro é um hobbit que lev...	O Hobbit - 7ª Ed. 2013 Produto NovoBilbo Bol...
1	Livro - It A Coisa - Stephen King	Produto NovoDurante as férias escolares de 195...	Livro - It A Coisa - Stephen King Produto No...
2	Box As Crônicas De Gelo E Fogo Pocket 5 Li...	Produto NovoTodo o reino de Westeros ao alcanc...	Box As Crônicas De Gelo E Fogo Pocket 5 Li...
3	Box Harry Potter	Produto Novo e Físico A série Harry Potter ch...	Box Harry Potter Produto Novo e Físico A sé...
4	Livro Origem - Dan Brown	Produto NovoDe Onde Viemos? Para Onde Vamos? R...	Livro Origem - Dan Brown Produto NovoDe Onde...

▼ **8** Tokenização e Contagem de Palavras


Nesta etapa, unificamos todos os textos da coluna `texto`, removemos acentuação, colocamos tudo em minúsculas e extraímos palavras com 3 ou mais letras para contar a frequência de ocorrência.

```
# Juntando todos os textos
todos_textos = " ".join(df_clean['texto'].values)

# Normalização: lowercase + remover acentos
todos_textos = unidecode.unidecode(todos_textos.lower())

# Tokenização: somente palavras com 3 ou mais letras
tokens = re.findall(r'\b[a-z]{3,}\b', todos_textos)

# Contagem
contador = Counter(tokens)
print(f"📊 Total de palavras únicas (com 3+ letras): {len(contador)}")
```



📊 Total de palavras únicas (com 3+ letras): 28653

▼ **9** Top 10 Palavras Mais Frequentes

Exibimos aqui as 10 palavras que mais ocorrem na coluna `texto`, já tratadas (minúsculas, sem acento e com no mínimo 3 letras).

```
# Top 10 palavras mais frequentes
top_10 = contador.most_common(10)

# Impressão formatada
print("10 Palavras mais frequentes:")
for palavra, freq in top_10:
    print(f"{palavra}: {freq}")
```

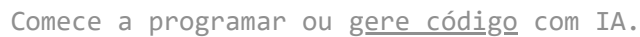
```
↔ 10 Palavras mais frequentes:
para: 8222
que: 6923
com: 6621
nao: 4118
produto: 3380
uma: 3300
mais: 2943
por: 2421
dos: 2075
sua: 1932
```

10 Nuvem de Palavras (Word Cloud)

Geramos uma nuvem de palavras com os tokens extraídos do campo `texto`, onde o tamanho de cada palavra representa sua frequência relativa. Essa visualização ajuda a identificar rapidamente os termos dominantes na base.

```
# Gerando a nuvem de palavras
wordcloud = WordCloud(
    background_color='white',
    max_words=100,
    width=800,
    height=400
).generate(" ".join(tokens))

# Exibindo a nuvem
plt.figure(figsize=(15, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title("10 Nuvem de Palavras")
plt.show()
```



- Aplicação de Stopwords (NLTK)

Nesta etapa, instalamos o pacote `nltk`, baixamos a lista de stopwords em português e removemos essas palavras dos nossos `tokens`. Em seguida, recalculamos a contagem e as palavras mais frequentes.


```
# Instalar o nltk (se necessário) e baixar as stopwords
%pip install nltk
```

```
import nltk
nltk.download('stopwords')

from nltk.corpus import stopwords

# Lista de stopwords em português
stopwords_pt = set(stopwords.words('portuguese'))

# Filtrando os tokens
tokens_filtrados = [t for t in tokens if t not in stopwords_pt]

# Recontagem com tokens filtrados
contador_filtrado = Counter(tokens_filtrados)

# Exibir Top 10 palavras filtradas
top_10_filtrado = contador_filtrado.most_common(10)
print("10 Top 10 Palavras (sem stopwords):")
for palavra, freq in top_10_filtrado:
    print(f"{palavra}: {freq}")
```



```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex<=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2021.8.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.66.1)
10 Top 10 Palavras (sem stopwords):
nao: 4118
produto: 3380
mercado: 1918
voce: 1885
frete: 1645
entrega: 1625
produtos: 1531
pagamento: 1494
envio: 1462
sao: 1409
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

▼ Nuvem de Palavras (Sem Stopwords)

Agora que removemos palavras irrelevantes (stopwords), geramos uma nova nuvem de palavras mais representativa do conteúdo real da base.

```
# Nuvem de palavras sem stopwords
wordcloud_filtrada = WordCloud(
    background_color='white',
    max_words=100,
    width=800,
    height=400
).generate(" ".join(tokens_filtrados))

# Exibição da nuvem
plt.figure(figsize=(15, 7))
plt.imshow(wordcloud_filtrada, interpolation='bilinear')
plt.axis('off')
plt.title("Nuvem de Palavras (Sem Stopwords)")
plt.show()
```



```
#####
```

```
#####
```

```
# Verifica se df_clean está definido e mostra suas dimensões
try:
    print("✅ df_clean existe.")
    print(f"Formato: {df_clean.shape[0]} linhas x {df_clean.shape[1]} colunas")
except NameError:
    print("❌ df_clean não está definido no ambiente atual.")
```



```
✅ df_clean existe.
Formato: 2916 linhas x 4 colunas
```

2) Utilizando o dataset de produtos [1]:

```
import pandas as pd
```

```
df = pd.read_csv(
    "https://dados-ml-pln.s3-sa-east-1.amazonaws.com/produtos.csv",
    delimiter=";",
    encoding='utf-8' )
```

```
df.head()
```



	nome	descricao	categoria
0	O Hobbit - 7ª Ed. 2013	Produto NovoBilbo Bolseiro é um hobbit que lev...	livro
1	Livro - It A Coisa - Stephen King	Produto NovoDurante as férias escolares de 195...	livro
2	Box As Crônicas De Gelo E Fogo Pocket 5 Li...	Produto NovoTodo o reino de Westeros ao alcanç...	livro
3	Box Harry Potter	Produto Novo e Físico A série Harry Potter ch...	livro
4	Livro Origem - Dan Brown	Produto NovoDe Onde Viemos? Para Onde Vamos? R...	livro

2.1. Elimine linhas com valores nulos

```
import pandas as pd
```

```
# 📁 Carregando o arquivo CSV
```

```
df = pd.read_csv(
    "https://dados-ml-pln.s3-sa-east-1.amazonaws.com/produtos.csv",
    delimiter=";",
    encoding='utf-8'
)
```

```
# 🇧🇷 Contagem antes da limpeza
```

```
linhas_antes = df.shape[0]
```

```
# ✂ Remoção total de linhas com valores nulos
```

```
df = df.dropna()
```

```
# 🇧🇷 Contagem depois da limpeza
```

```
linhas_depois = df.shape[0]
```

```
# 🔍 Quantas foram removidas
```

```
print(f"Linhas removidas com valores nulos: {linhas_antes - linhas_depois}")
```

```
print(f"Total restante: {linhas_depois}")
```



```
Linhas removidas com valores nulos: 1164
Total restante: 2916
```

```
# resposta
```



2.2. Adicione uma nova coluna chamada texto, formada pela composição das colunas nome e descrição

```
# 📄 Criando a nova coluna 'texto' com a junção de 'nome' + 'descricao'
df['texto'] = df['nome'].astype(str) + ' ' + df['descricao'].astype(str)

# ✅ Visualizando as 5 primeiras linhas da nova coluna
df[['nome', 'descricao', 'texto']].head()
```



	nome	descricao	texto
0	O Hobbit - 7ª Ed. 2013	Produto NovoBilbo Bolseiro é um hobbit que lev...	O Hobbit - 7ª Ed. 2013 Produto NovoBilbo Bol...
1	Livro - It A Coisa - Stephen King	Produto NovoDurante as férias escolares de 195...	Livro - It A Coisa - Stephen King Produto No...
2	Box As Crônicas De Gelo E Fogo Pocket 5 Li...	Produto NovoTodo o reino de Westeros ao alcanc...	Box As Crônicas De Gelo E Fogo Pocket 5 Li...
3	Box Harry Potter	Produto Novo e Físico A série Harry Potter ch...	Box Harry Potter Produto Novo e Físico A sé...
4	Livro Origem - Dan Brown	Produto NovoDe Onde Viemos? Para Onde Vamos? R...	Livro Origem - Dan Brown Produto NovoDe Onde...

resposta

df.head()



	nome	descricao	categoria	texto
0	O Hobbit - 7ª Ed. 2013	Produto NovoBilbo Bolseiro é um hobbit que lev...	livro	O Hobbit - 7ª Ed. 2013 Produto NovoBilbo Bol...
1	Livro - It A Coisa - Stephen King	Produto NovoDurante as férias escolares de 195...	livro	Livro - It A Coisa - Stephen King Produto No...
2	Box As Crônicas De Gelo E Fogo Pocket 5 Li...	Produto NovoTodo o reino de Westeros ao alcanc...	livro	Box As Crônicas De Gelo E Fogo Pocket 5 Li...
3	Box Harry Potter	Produto Novo e Físico A série Harry Potter ch...	livro	Box Harry Potter Produto Novo e Físico A sé...
4	Livro Origem - Dan Brown	Produto NovoDe Onde Viemos? Para Onde Vamos? R...	livro	Livro Origem - Dan Brown Produto NovoDe Onde...



2.3. Conte quantos Unigramas existem antes e depois de remover stopwords (use a coluna texto)

```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
```

```
# 📄 Garante que as stopwords estejam disponíveis
```

```
nltk.download('stopwords')
```

```
stopwords_pt = stopwords.words('portuguese')
```

```
# 📄 CountVectorizer sem remoção de stopwords
```

```
vectorizer_sem_stop = CountVectorizer(token_pattern=r'\b\w\w\w+\b') # palavras com 3+ letras
```

```
X_sem_stop = vectorizer_sem_stop.fit_transform(df['texto'])
```

```
# 📄 CountVectorizer com stopwords
```

```
vectorizer_com_stop = CountVectorizer(
    stop_words=stopwords_pt,
    token_pattern=r'\b\w\w\w+\b' # 3+ letras
)
```

```
X_com_stop = vectorizer_com_stop.fit_transform(df['texto'])
```

```
# 📄 Contagens
```

```
print("### Unigramas únicos com CountVectorizer")
```

```
print(f"📄 Antes de remover stopwords: {len(vectorizer_sem_stop.get_feature_names_out())}")
```

```
print(f"📄 Depois de remover stopwords: {len(vectorizer_com_stop.get_feature_names_out())}")
```

```
➡ [nltk_data] Downloading package stopwords to /root/nltk_data...
```

```
[nltk_data] Package stopwords is already up-to-date!
```

```
### Unigramas únicos com CountVectorizer
```

```
📄 Antes de remover stopwords: 34994
```

```
📄 Depois de remover stopwords: 34854
```

```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
nltk.download('stopwords')

# 🚩 Parâmetros comuns
token_pattern = r'\b\w\w\w+\b'
stopwords_pt = stopwords.words('portuguese')
```

```
# 📄 CountVectorizer sem stopwords
vsct = CountVectorizer(
    token_pattern=token_pattern,
    ngram_range=(1, 1)
)
vsct.fit(df['texto'])
total_unigrams_sem_stop = len(vsct.get_feature_names_out())
```

```
# 📄 CountVectorizer com stopwords
vsct = CountVectorizer(
    stop_words=stopwords_pt,
    token_pattern=token_pattern,
    ngram_range=(1, 1)
)
vsct.fit(df['texto'])
total_unigrams_com_stop = len(vsct.get_feature_names_out())
```

```
# 🖨️ Resultado final
print("#### 2.7. Quantidade de Unigramas na coluna 'texto'\n")
print(f"📄 Antes de remover stopwords: {total_unigrams_sem_stop}")
print(f"📄 Depois de remover stopwords: {total_unigrams_com_stop}")
```

```
🔄 [nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
#### 2.7. Quantidade de Unigramas na coluna 'texto'

📄 Antes de remover stopwords: 34994
📄 Depois de remover stopwords: 34854
```

```

import nltk
from nltk.tokenize import TreebankWordTokenizer
from nltk.corpus import stopwords

# 📥 Garantindo download das stopwords
nltk.download('stopwords')

# 🛠 Inicializando tokenizer e stopwords
tokenizer = TreebankWordTokenizer()
stopwords_pt = set(stopwords.words('portuguese'))

# 📦 Armazenamento dos tokens
tokens_todos = []
tokens_filtrados = []

# 🔄 Tokenização e limpeza
for texto in df['texto']:
    tokens = tokenizer.tokenize(texto.lower())
    tokens = [t for t in tokens if t.isalpha()]

    tokens_todos.extend(tokens)
    tokens_filtrados.extend([t for t in tokens if t not in stopwords_pt])

# 🇧🇷 Contagem de unigrams únicos
print(f"📦 Unigramas únicos antes das stopwords: {len(set(tokens_todos))}")
print(f"📉 Unigramas únicos depois das stopwords: {len(set(tokens_filtrados))}")

🔄 [nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
📦 Unigramas únicos antes das stopwords: 27784
📉 Unigramas únicos depois das stopwords: 27620

```

📁 2.3. (Corrigido) Quantidade de Unigramas com e sem Stopwords

Recontamos agora os unigramas (palavras únicas com 3+ letras) diretamente da coluna

`df_clean['texto']`, com e sem a remoção de stopwords, garantindo consistência na origem dos dados.

```

import nltk
from nltk.tokenize import TreebankWordTokenizer
from nltk.corpus import stopwords

# 📄 Baixar stopwords (caso ainda não esteja feito)
nltk.download('stopwords')

# ✂ Inicializa tokenizer e lista de stopwords
tokenizer = TreebankWordTokenizer()
stopwords_pt = set(stopwords.words('portuguese'))

# 📋 Listas de tokens
tokens_todos = []
tokens_filtrados = []

# 🔄 Tokenização linha a linha
for texto in df['texto']:
    tokens = tokenizer.tokenize(texto.lower())
    tokens = [t for t in tokens if t.isalpha() and len(t) >= 3] # só palavras com 3+ letras
    tokens_todos.extend(tokens)
    tokens_filtrados.extend([t for t in tokens if t not in stopwords_pt])

# 🇧🇷 Contagem de unigrams únicos
print("## 📄 2.3. (Corrigido) Quantidade de Unigramas com e sem Stopwords\n")
print(f"📋 📋 Unigramas únicos (3+ letras) ANTES de remover stopwords: {len(set(tokens_todos))}")
print(f"▼ 📋 Unigramas únicos (3+ letras) DEPOIS de remover stopwords: {len(set(tokens_filtrados))}")

🔄 [nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
## 📄 2.3. (Corrigido) Quantidade de Unigramas com e sem Stopwords

📋 📋 Unigramas únicos (3+ letras) ANTES de remover stopwords: 27532
▼ 📋 Unigramas únicos (3+ letras) DEPOIS de remover stopwords: 27392

```



```

from nltk.corpus import stopwords
import re
import unicode

# Reunindo todos os textos
texto_total = " ".join(df['texto'].astype(str))

# Pré-processamento comum: minúsculas e sem acento
texto_normalizado = unicode.unidecode(texto_total.lower())

# Tokenização simples: apenas palavras com 3 ou mais letras
tokens_gerais = re.findall(r'\b[a-z]{3,}\b', texto_normalizado)

# Versão com e sem stopwords
stopwords_pt = set(stopwords.words('portuguese'))

# Set de unigramas únicos
unigramas_antes = set(tokens_gerais)
unigramas_depois = set([t for t in tokens_gerais if t not in stopwords_pt])

# Contagem
print(f"📊 Unigramas únicos antes das stopwords: {len(unigramas_antes)}")
print(f"▼ Unigramas únicos depois das stopwords: {len(unigramas_depois)}")

🔄 📊 Unigramas únicos antes das stopwords: 28653
▼ Unigramas únicos depois das stopwords: 28527

# resposta

```

✓ 2.4. Conte quantos Bigramas existem antes e depois de remover stopwords (use a coluna texto)

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
from nltk.corpus import stopwords
```

```
nltk.download('stopwords')
```

```
# 🚩 Parâmetros comuns
```

```
token_pattern = r'\b\w\w\w+\b' # Apenas palavras com 3+ letras
```

```
stopwords_pt = stopwords.words('portuguese')
```

```
# 🔗 Bigrams SEM stopwords
```

```
vsct = CountVectorizer(  
    token_pattern=token_pattern,  
    ngram_range=(2, 2) # Bigrams  
)
```

```
vsct.fit(df['texto'])
```

```
total_bigrams_sem_stop = len(vsct.get_feature_names_out())
```

```
# 🔗 Bigrams COM stopwords
```

```
vsct = CountVectorizer(  
    stop_words=stopwords_pt,  
    token_pattern=token_pattern,  
    ngram_range=(2, 2)  
)
```

```
vsct.fit(df['texto'])
```

```
total_bigrams_com_stop = len(vsct.get_feature_names_out())
```

```
# 🖨️ Resultado final
```

```
print("#### 2.8. Quantidade de Bigramas na coluna 'texto'\n")
```

```
print(f"📊 Antes de remover stopwords: {total_bigrams_sem_stop}")
```

```
print(f"▼ Depois de remover stopwords: {total_bigrams_com_stop}")
```

```
🔄 [nltk_data] Downloading package stopwords to /root/nltk_data...
```

```
[nltk_data] Package stopwords is already up-to-date!
```

```
#### 2.8. Quantidade de Bigramas na coluna 'texto'
```

```
📊 Antes de remover stopwords: 151560
```

```
▼ Depois de remover stopwords: 140103
```

```
from nltk.util import bigrams
```

```
# 📄 Listas para armazenar bigramas
```

```
bigrams_todos = []
```

```
bigrams_filtrados = []
```

```
# 🔄 Geração de bigramas linha a linha
```

```
for texto in df['texto']:
```

```
    tokens = tokenizer.tokenize(texto.lower())
```

```
    tokens = [t for t in tokens if t.isalpha() and len(t) >= 3]
```

```
    # Bigramas antes de remover stopwords
```

```
    bigrams_todos.extend(list(bigrams(tokens)))
```

```
    # Remoção de stopwords para bigramas filtrados
```

```
    tokens_sem_stop = [t for t in tokens if t not in stopwords_pt]
```

```
    bigrams_filtrados.extend(list(bigrams(tokens_sem_stop)))
```

```
# 🇧🇷 Contagem de bigramas únicos
```

```
print("## 🔗 2.4. Quantidade de Bigramas com e sem Stopwords\n")
```

```
print(f"📄 Bigramas únicos (3+ letras) ANTES de remover stopwords: {len(set(bigrams_todos))}")
```

```
print(f"▼ Bigramas únicos (3+ letras) DEPOIS de remover stopwords: {len(set(bigrams_filtrados))}")
```



```
## 🔗 2.4. Quantidade de Bigramas com e sem Stopwords
```

```
📄 Bigramas únicos (3+ letras) ANTES de remover stopwords: 132607
```


```
▼ Bigramas únicos (3+ letras) DEPOIS de remover stopwords: 120990
```

```
# resposta
```



2.5. Conte quantos Trigramas existem antes e depois de remover stopwords (use a coluna texto)

```
from nltk.util import trigrams
```

```
#  Listas para armazenar trigramas
```

```
trigrams_todos = []
```

```
trigrams_filtrados = []
```

```
# resposta
```

```
tokens = tokenizer.tokenize(texto.lower())
```

```
tokens = [t for t in tokens if t.isalpha() and len(t) >= 3]
```


✓ 2.6. Conte quantos unigramas existem na coluna texto após aplicar Stemmer (utilize rslp)

```
# Trigramas antes de remover stopwords
```

```
from nltk.stem import RSLPStemmer
```

```
#  Baixando o stemmer para português
```

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

```
#  Inicializa o Stemmer
```

```
stemmer = RSLPStemmer()
```