

## ▼ Classificação de textos - Análise de Sentimentos

### Processamento de Linguagem Natural

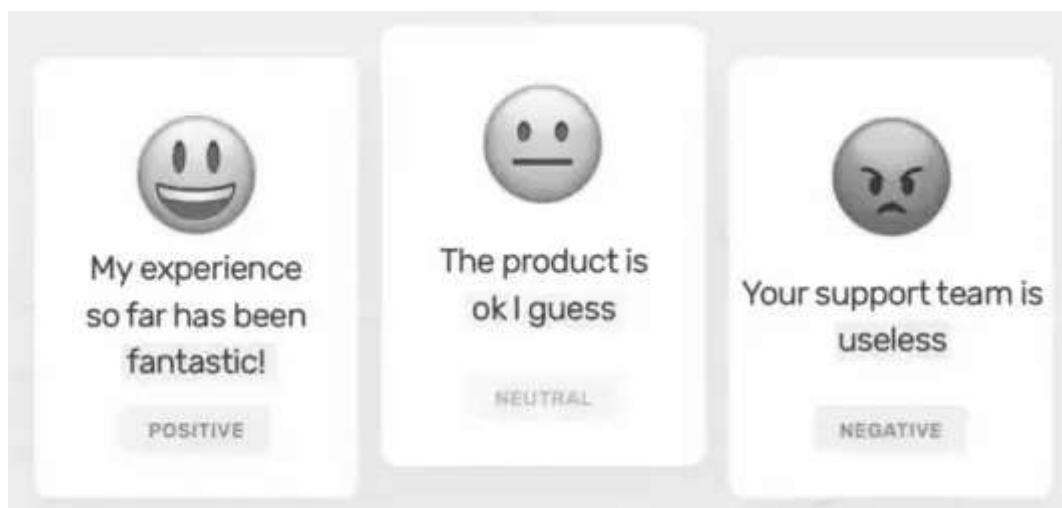
Nesta aula continuaremos trabalhando com Classificação de textos, porém agora focaremos em uma tarefa específica, chamada de **Análise de Sentimentos**.

O objetivo é que ao final desta aula você:

1. Entenda o que é a **Análise de Sentimentos**
2. Saiba treinar um classificador supervisionado para realizar análise de sentimentos usando um corpus anotado
3. Aprenda a utilizar a biblioteca Polyglot do Python que já possui métodos para identificar polaridade de textos em português
4. Acesse API do Twitter para obter dados em tempo real de redes sociais

### O que é a Análise de Sentimentos?

É a interpretação e classificação de emoções relativas a um texto, sendo estas emoções explícitas declaradas no texto ou implícitas.



É uma tarefa de PLN/Machine Learning que permite empresas identificarem os sentimentos dos consumidores em relação a produtos, marcas ou serviços, através de opiniões deixadas em redes sociais ou em canais de comunicação da empresa.

## ▼ Como treinar um classificador para realizar a Análise de Sentimentos?

Iremos desenvolver nosso classificador, utilizando uma abordagem supervisionada, ou seja, precisaremos de dados rotulados com suas respectivas emoções.

O pipeline de execução é muito similar aos exemplos de classificação realizados na última semana, o que muda efetivamente são as classes/categorias envolvidas.

### Dados

Para este exemplo iremos trabalhar com uma base de dados de notícias, rotulada com as emoções básicas de Ekman: **alegria, tristeza, raiva, medo, repugnância e surpresa**. Em caso de ausência de emoção, a categoria **neutro** foi aplicada.

**IMPORTANTE:** Faça o upload da base de dados para seu ambiente Google Colab ou Jupyter Notebook! Segue o [link](#) para acesso.

## ▼ Fluxo de execução

Vamos seguir o seguinte fluxo de processamento dos dados:

1. Abrir o corpus
2. Remover as stop-words
3. Aplicar stemmer
4. Gerar o Bag of Words
5. Treinar o modelo SVM
6. Predizer/Avaliar o modelo

```
import pandas as pd
from numpy import True_

f = open("/content/analise-sentimentos-2000-noticias.txt", "r", encoding="utf-8-sig")
linhas = f.readlines()

corpus_textos = []
rotulo = []

# Percorre as 2000 linhas
for linha in linhas:

    # Separa texto e rótulo/categoria/emoção
    item = linha.split(";;")
    if item[0] != 'surpresa':
        rotulo.append(item[0])
        corpus_textos.append(item[1])

    df = pd.Series(rotulo)

    rotulo2 = df.replace(['alegria','tristeza','raiva', 'medo', 'desgosto', 'surpresa'],['positivo', 'negativo','negativo', 'negativo'])

    corpus_rotulos = rotulo2

from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

#Comparando o tamanho dos corpus após edição do arquivo.

tamanho_rotulos = len(corpus_rotulos)
print('Esse é o tamanho do corpus de rótulos: ', tamanho_rotulos)
tamanho_textos = len(corpus_textos)
print('Esse é o tamanho do corpus de textos: ', tamanho_textos)
```

```
Esse é o tamanho do corpus de rótulos: 1748  
Esse é o tamanho do corpus de textos: 1748
```

```
# 5 primeiros textos  
corpus_textos[0:10]
```

- ⇨ ['irã liberta viúva condenada ao apedrejamento, diz comitê internacional. a viúva iraniana sakineh mohammadi-ashtiani, condenada à morte por apedrejamento, foi libertada juntamente com o seu filho e o seu advogado, anunciou nesta quinta-feira (9) o comitê internacional contra apedrejamento, sediado na alemanha.\n',  
'haiti sofre com cólera depois da passagem do furacão sandy. após a passagem do furacão sandy pelo haiti, o país vive as dificuldades da reconstrução, além de problemas causados pela contaminação da bactéria que causa o cólera.\n',  
'enchentes causam a morte de 4 pessoas em al: vítimas são mãe, filho, idosa de 100 anos e adolescente. defesa civil da capital já registrou mais de 30 ocorrências.\n',  
'sem detalhes do que será discutido, revisão do plano diretor começa na segunda (17). dez oficinas, 19 audiências públicas e canal interativo serão os meios de participação popular; ippuc pretende enviar plano revisado à câmara até o início de dezembro\n',  
'chávez apresenta neto de allende como companheiro da filha: pablo sepúlveda allende é médico e neto de salvador allende. maría, de 29 anos, é a segunda filha do primeiro casamento de chávez.\n',  
'chefe do pc chinês de xinjiang ameaça executar envolvidos em protestos: forças do governo tomaram as ruas da capital da província, urumqi. confrontos entre uigures, chineses han e a polícia já mataram 156.\n',  
'jovem fingia ser modelo para aliciar crianças na web. guilherme donin é suspeito de chantagear menina de 12 anos na internet.\n',  
'colômbia se diz disposta a atuar como mediadora na venezuela. presidente colombiano disse que poderia fazer este papel. ele destacou que evitará provocar caracas com seus comentários.\n',  
'carlinhos brown e guilherme arantes cantam para namorados na bahia: apresentação aconteceu na costa do sauípe nesta sexta-feira. brown surpreendeu com músicas românticas na madrugada de sábado.\n',  
'juiz aponta advogado para cuidar das finanças de óctuplos nos eua: mãe, nadya soleman, foi à audiência, mas não se pronunciou. reality show que filmaria sua rotina depende de aprovação da justiça.\n']

```
# 5 primeiros rótulos  
corpus_rotulos[0:10]
```

0	positivo
1	negativo
2	negativo
3	neutro
4	neutro
5	negativo

```
6    negativo
7    neutro
8    positivo
9    neutro
dtype: object
```

Em nossos exemplos de classificação anteriores, separamos parte do banco de dados para **TREINAMENTO** e outra parte para **TESTE**, nesse tipo de avaliação que chamamos de **hold-out**.

Existem outras formas de realizar a avaliação, inclusive mais indicadas de acordo com a situação, mas isto não está no escopo de nossa disciplina, caso queira saber mais métodos de avaliação como o **cross-validation**, leia [este post](#).

```
from sklearn.model_selection import train_test_split

# O próprio sklearn tem um método para dividir a base de dados em treinamento e teste
# Neste caso estamos deixando 90% para treinamento e 10% para testes

corpus_treinamento, corpus_teste, rotulos_treinamento, rotulos_teste = train_test_split(corpus_textos, corpus_rotulos, test_size=0.10

tamanhoTextoTreinamento = len(corpus_treinamento)
print('Esse é o tamanho do corpus de texto para treinamento: ', tamanhoTextoTreinamento)
tamanhoRotuloTreinamento = len(rotulos_treinamento)
print('Esse é o tamanho do corpus de rótulos para treinamento: ', tamanhoRotuloTreinamento)
tamanhoTextoTeste = len(corpus_teste)
print('\nEsse é o tamanho do corpus de textos para teste: ', tamanhoTextoTeste)
tamanhoRotulosTeste = len(rotulos_teste)
print('Esse é o tamanho do corpus de rótulos para teste: ', tamanhoRotulosTeste)

Esse é o tamanho do corpus de texto para treinamento: 1573
Esse é o tamanho do corpus de rótulos para treinamento: 1573

Esse é o tamanho do corpus de textos para teste: 175
Esse é o tamanho do corpus de rótulos para teste: 175
```

Vamos deixar preparada uma função para pré-processar os textos, utilizando uma lista de stop-words com novos itens, o stemming e normalização dos textos.

```
import nltk
from nltk import tokenize
nltk.download('stopwords')
nltk.download('rslp')
nltk.download('punkt')

stopwords = nltk.corpus.stopwords.words('portuguese') #carrega stopwords da lingua portuguesa disponíveis no NLTK
stopwords += (',','.',',','(',')','"','"','`','`','!','$','%','&','...','-',':',';','?','`','`','\``') #acrescenta simbolos
stopwords += ('a','e','i','o','u','A','E','I','O','U') #acrescenta também vogais

stemmer = nltk.stem.RSLPStemmer()

def my_preprocessor(text):

    # Faz a remoção de tudo que não seja letra (essa adição é resposta ao exercício número 3)
    text.replace("[^a-zA-Z#]", " ")

    # Normaliza para minúsculas
    text=text.lower()

    # Tokeniza
    words = tokenize.word_tokenize(text, language='portuguese')
    # Remove stop-words
    words_no_stopwords = [word for word in words if not word in stopwords]
    # Aplica stemming
    stemmed_words=[stemmer.stem(word=word) for word in words_no_stopwords]
    return ' '.join(stemmed_words)

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package rslp to /root/nltk_data...
[nltk_data]   Unzipping stemmers/rslp.zip.
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]  Unzipping tokenizers/punkt.zip.
```

Agora vamos extrair os atributos do texto (gerar a representação vetorial - bag of words) e criar nosso pipeline de classificação usando o classificador SVM.

```
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

# Primeiro aplica o BoW, depois envia dados ao classificador SVM
# (SEM retirada de stop-words e stemming)
#sent_clf = Pipeline([('vect', CountVectorizer()), ('clf', SVC(kernel='linear', C=1))])

# Depois de executar uma vez, verifique os resultados e compare-os depois de descomentar a linha abaixo, onde retiramos as stop-words
# (COM retirada de stop-words e stemming)

# Em resposta ao exercício número 2: foi adicionado a remoção de acentos e "ngram_range". Para o classificador, foi
sent_clf = Pipeline([('vect', CountVectorizer(strip_accents ='unicode', ngram_range=(1,2), preprocessor = my_preprocessor)), ('clf', S

from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB

#Essa é a resposta ao exercício 4, onde pede-se que seja testado com outro modelo de classificação. Neste caso, Naive Bayes.
sent_clf2 = Pipeline([('vect', CountVectorizer(strip_accents ='unicode', ngram_range=(1,2), preprocessor = my_preprocessor)), ('clf', 

# Inicia treinamento
sent_clf = sent_clf.fit(corpus_treinamento, rotulos_treinamento)

# Inicia treinamento do modelo editado (clf2)
sent_clf2 = sent_clf2.fit(corpus_treinamento, rotulos_treinamento)

/usr/local/lib/python3.7/dist-packages/sklearn/naive_bayes.py:557: UserWarning: alpha too small will result in numeric errors,
```

```
% _ALPHA_MIN
```

Já temos nosso modelo treinado! Agora vamos predizer a base de teste e avaliar os resultados.

```
# Prediz base de teste
rotulos_preditos = sent_clf.predict(corpus_teste)
```

```
# Prediz base de teste para o modelo editado
rotulos_preditos = sent_clf2.predict(corpus_teste)
```

```
#Apresenta o resultado de acuracidade do modelo original (SVM)
```

```
from sklearn.metrics import accuracy_score
rotulos_preditos = sent_clf.predict(corpus_teste)
print("Acurácia do modelo original: ",accuracy_score(rotulos_preditos, rotulos_teste))
```

```
Acurácia do modelo original: 0.7942857142857143
```

```
#Apresenta o resultado de acuracidade do modelo editado (Naive Bayes)
```

```
from sklearn.metrics import accuracy_score
rotulos_preditos = sent_clf2.predict(corpus_teste)
print("Acurácia do modelo editado: ",accuracy_score(rotulos_preditos, rotulos_teste))
```

```
Acurácia do modelo editado: 0.7771428571428571
```

```
from sklearn.metrics import classification_report
```

```
# Mostra relatório completo de avaliação
print(classification_report(rotulos_teste, rotulos_preditos))
```

```
precision    recall    f1-score    support
```

negativo	0.76	0.97	0.85	103
neutro	0.83	0.60	0.70	58
positivo	1.00	0.07	0.13	14
accuracy			0.78	175
macro avg	0.86	0.55	0.56	175
weighted avg	0.80	0.78	0.74	175

## Modelo original do exercício

	precision	recall	f1-score	support
alegria	0.00	0.00	0.00	19
desgosto	0.44	0.26	0.33	27
medo	0.32	0.39	0.35	18
neutro	0.52	0.73	0.61	51
raiva	0.29	0.29	0.29	7
surpresa	0.42	0.47	0.44	32
tristeza	0.66	0.59	0.62	46
accuracy			0.48	200

macro avg 0.38 0.39 0.38 200 weighted avg 0.45 0.47 0.45 200

```
from sklearn.metrics import confusion_matrix

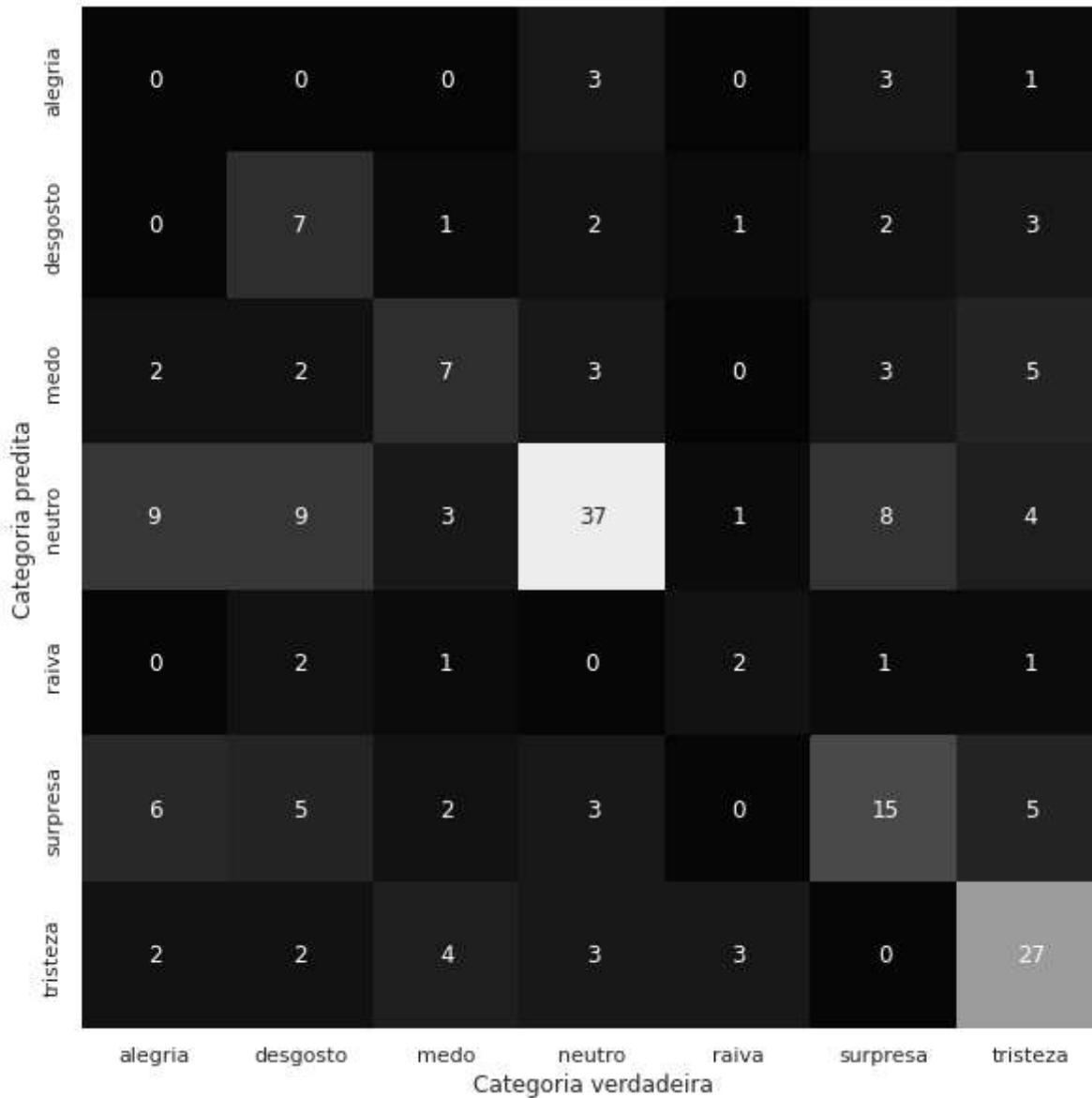
# Podemos imprimir a matriz de confusão para tentar entender melhor os resultados
mat = confusion_matrix(rotulos_teste, rotulos_preditos)

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
```

```
rotulos_nomes = ['positivo', 'neutro', 'negativo']

fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False, xticklabels=rotulos_nomes, yticklabels=rotulos_nomes )
```

## Modelo original do exercício



**CONCLUSÃO:** Apesar de verificarmos melhoria ao pré-processarmos o texto, os resultados ainda são regulares.

## ▼ Detalhamento das atividades realizadas.

1. **Atividade 1:** Criamos um script que ignorava as linhas que continham "surpresa" e então realizamos as mudanças dos labels das emoções através de um "replace".
2. **Atividade 2:** Para o exercício 2, adicionamos dois novos atributos: "strip\_accents" para remoção da acentuação, mas sem mudanças de acuracidade; e "n\_gram(1,2)", que melhorou um pouco a acuracidade do modelo. Já para o classificador, no original tentei mudar o kernel para "sigmoid" mas houve a piora da acuracidade em 0.03% (de 0.8 para 0.77) e então foi removido e voltei ao original, além da mudança do kernel, tentei a adição do atributo "probability", que não mudou a acuracidade e foi mantido.
3. **Atividade 3:** A troca do para TfidfVectorizer na realidade apresentou queda na acuracidade de predição das emoções, e então mantive o "countvectorize", mas adicionei na etapa de pré-processamento a remoção de tudo que não era letra (text.replace("[^a-zA-Z#]", " ")), embora isso não tenha apresentado melhorias significativas no modelo.
4. **Atividade 4:** Utilizamos o classificador Naive Bayes. Testamos com outros classificadores também, que embora nenhum tenha chego na acuracidade do modelo original (SVC -> 0.8), o Naive Bayes apresentou acuracidade de 0.77. Alguns dos demais testes: RandomForest -> 0.71, SGD -> 0.77 e KNeighborsClassifier -> 0.6.

## Atividade Somativa 2 - Como podemos tentar melhorar os resultados?

**1) Redução da granularidade dos sentimentos** Como podemos ver na matriz de confusão o classificador tem vários pontos de erro, em todas emoções. E se modificarmos o corpus para ao invés de 7 emoções, trabalhar com as 3 clássicas (positivo, neutro, negativo)?

Vamos então adotar a seguinte sistemática de atualização da base de dados:

A classe "**positivo**" será obtida utilizando-se as instâncias da base original rotuladas como "alegria".

A classe "**negativo**" será obtida utilizando-se as instâncias rotuladas como "raiva", "medo", "desgosto" e "tristeza".

Já a classe "**neutro**", utilizará as instâncias da base original rotuladas como "neutro".

As instâncias rotuladas como "surpresa" não serão utilizadas.

Crie um novo arquivo para esta base de dados atualizada, carregue-o em nosso notebook e compare os resultados.

- 2) Configuração dos parâmetros de extração de atributos e do classificador** Altere a configuração de pelo menos um atributo gerado pela vetorização das palavras (i.e., CountVectorizer) e também pelo classificador utilizado.
- 3) Adicione novas etapas de extração de atributos ou de pré-processamento** Incorpore ao menos uma nova etapa de pré-processamento ou extração de atributos (e.g., TF-IDF) ao pipeline atual.
- 4) Utilize outro classificador de texto** Treine ao menos um novo modelo utilizando um classificador de textos diferentes. Ele pode pertencer à biblioteca sklearn ou qualquer outra.

## ▼ Existe alguma ferramenta pronta que realize análise de sentimentos?

Não é sempre que teremos um corpus anotado para utilizar em nossas tarefas, neste caso, existem bibliotecas que incorporam funcionalidades simples que podem ser usadas para detectar a polaridade de textos (negativo, positivo).

No caso da língua portuguesa, a [Polyglot](#) tem um léxico de polaridade das palavras, que pode ser usado para detecção simples de sentimentos.

```
# Ela tem algumas dependências que devem ser instaladas
!pip install -U git+https://github.com/aboSamoor/polyglot.git@master
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
Collecting git+https://github.com/aboSamoor/polyglot.git@master
  Cloning https://github.com/aboSamoor/polyglot.git (to revision master) to /tmp/pip-req
    Running command git clone -q https://github.com/aboSamoor/polyglot.git /tmp/pip-req-bu
Collecting futures>=2.1.6
  Downloading futures-3.0.5.tar.gz (25 kB)
WARNING: Discarding https://files.pythonhosted.org/packages/55/db/97c1ca37edab586a1ae03d
  Downloading futures-3.0.4.tar.gz (25 kB)
WARNING: Discarding https://files.pythonhosted.org/packages/8d/73/b5fff618482bc06c9711e7
  Downloading futures-3.0.3.tar.gz (24 kB)
WARNING: Discarding https://files.pythonhosted.org/packages/4c/dc/f9473006d4c9c52d4a4e97
  Downloading futures-3.0.2.tar.gz (24 kB)
WARNING: Discarding https://files.pythonhosted.org/packages/f8/e7/fc0fcbeb9193ba2d4de00b
  Downloading futures-3.0.1.tar.gz (24 kB)
WARNING: Discarding https://files.pythonhosted.org/packages/b2/2c/6b6a57379e47031c6f52e6
  Downloading futures-3.0.0.tar.gz (24 kB)
WARNING: Discarding https://files.pythonhosted.org/packages/ea/c9/35287369718fc05059e7a9
  Downloading futures-2.2.0-py2.py3-none-any.whl (16 kB)
Collecting morfessor>=2.0.2a1
  Downloading Morfessor-2.0.6-py3-none-any.whl (35 kB)
Requirement already satisfied: wheel>=0.23.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.7/dist-packages (f
Collecting pyclld2>=0.3
  Downloading pyclld2-0.41.tar.gz (41.4 MB)
    |██████████| 41.4 MB 1.2 MB/s
Requirement already satisfied: six>=1.7.3 in /usr/local/lib/python3.7/dist-packages (fro
Collecting PyICU>=1.8
  Downloading PyICU-2.9.tar.gz (305 kB)
    |██████████| 305 kB 51.1 MB/s
Installing build dependencies ... done
Getting requirements to build wheel ... done
  Preparing wheel metadata ... done
Building wheels for collected packages: polyglot, pyclld2, PyICU
  Building wheel for polyglot (setup.py) ... done
  Created wheel for polyglot: filename=polyglot-16.7.4-py2.py3-none-any.whl size=70673 s
  Stored in directory: /tmp/pip-ephem-wheel-cache-txpku4g/wheels/6d/b6/83/526ac20beb2e8
  Building wheel for pyclld2 (setup.py) ... done
  Created wheel for pyclld2: filename=pyclld2-0.41-cp37-cp37m-linux_x86_64.whl size=983425
  Stored in directory: /root/.cache/pip/wheels/ed/e4/58/ed2e9f43c07d617cc81fe7aff0fc6e42
  Building wheel for PyICU (PEP 517) ... done
  Created wheel for PyICU: filename=PyICU-2.9-cp37-cp37m-linux_x86_64.whl size=1375525 s
```

```
Stored in directory: /root/.cache/pip/wheels/28/88/93/6c1b06361e4cbd4e7f793fb456729f69

import polyglot
from polyglot.text import Text
# Baixa o léxicos da lingua portuguesa
!polyglot download LANG:pt

[polyglot_data] Downloading collection 'LANG:pt'
[polyglot_data]   | Downloading package sgns2.pt to
[polyglot_data]   |       /root/polyglot_data...
[polyglot_data]   | Downloading package unipos.pt to
[polyglot_data]   |       /root/polyglot_data...
[polyglot_data]   | Downloading package ner2.pt to
[polyglot_data]   |       /root/polyglot_data...
[polyglot_data]   | Downloading package counts2.pt to
[polyglot_data]   |       /root/polyglot_data...
[polyglot_data]   | Downloading package transliteration2.pt to
[polyglot_data]   |       /root/polyglot_data...
[polyglot_data]   | Downloading package embeddings2.pt to
[polyglot_data]   |       /root/polyglot_data...
[polyglot_data]   | Downloading package uniemb.pt to
[polyglot_data]   |       /root/polyglot_data...
[polyglot_data]   | Downloading package pos2.pt to
[polyglot_data]   |       /root/polyglot_data...
[polyglot_data]   | Downloading package sentiment2.pt to
[polyglot_data]   |       /root/polyglot_data...
[polyglot_data]   | Downloading package tsne2.pt to
[polyglot_data]   |       /root/polyglot_data...
[polyglot_data]   | Downloading package morph2.pt to
[polyglot_data]   |       /root/polyglot_data...
[polyglot_data]   | Done downloading collection LANG:pt

text = Text("O filme que vimos é realmente muito bom!")
print("{:<16}{}".format("Word", "Polarity")+"\n"+"-"*30)
for w in text.words:
    print("{:<16}{:>2}{}".format(w, w.polarity))
```

Word	Polarity
0	0
filme	0
que	0
vimos	0
é	0
realmente	0
muito	1
bom	1
!	0

```
text = Text("As notas foram ruins.")
print("{:<16}{:}>".format("Word", "Polarity")+"\n"+"-"*30)
for w in text.words:
    print("{:<16}{:>2}>".format(w, w.polarity))
```

Word	Polarity
As	0
notas	0
foram	0
ruins	-1
.	0

```
text = Text("Não sei o que pensar.")
print("{:<16}{:}>".format("Word", "Polarity")+"\n"+"-"*30)
for w in text.words:
    print("{:<16}{:>2}>".format(w, w.polarity))
```

Word	Polarity
Não	0
sei	0
o	0
que	0
pensar	0
.	0

## ▼ Como obter dados em tempo real de redes sociais?

Já vimos na disciplina que é possível utilizar o web-scraping para obter páginas da web e percorrer a estrutura HTML em busca das informações. Porém, em alguns casos temos opções mais rápidas e viáveis para recolher dados. Alguns sites oferecem o serviço de [API](#) para acesso aos dados, onde podemos autenticar e utilizar funções para buscar dados em tempo real.

Em nosso exemplo, utilizaremos a [API do Twitter](#) para buscar tweets sobre determinado assunto.

Neste exemplo, utilizaremos as chaves de acesso da conta do professor, mas quando você for desenvolver seu script, você deve criar seu próprio token de acesso. Basta acessar sua conta no Twitter, na [página do desenvolvedor](#) você deve ir em Apps > Create an app.

```
import tweepy
from tweepy import OAuthHandler

# As chaves e tokens que você receberá ao criar um App
consumer_key = 'iUpbJi0v2LGFZdRTSq00Fndf7'
consumer_secret = 'AVQQS4UcSI6dlRkwO3PdrNo6Iuw19k5yE4oae18yNvACnhREzs'
access_token = '12948382-GTwhQS1j2y1AsMxsm2TT8ecYQ8fUteAfNGTqJutAP'
access_token_secret = 'U1HOAmQWkSwfiGEacEqDL59QZ8dvpBhG03yiusjyhtI2M'

api = None
try:
    # Cria um objeto de autenticação (OAuthHandler)
    auth = OAuthHandler(consumer_key, consumer_secret)
    # Define o token e senha de acesso
    auth.set_access_token(access_token, access_token_secret)
    # Cria um novo objeto API para acessar os tweets
    api = tweepy.API(auth)
except:
    print("Erro: Falha de autenticação no Twitter")
```

Uma vez que temos um objeto API autenticado, podemos fazer nossas buscas. A API oferecer vários métodos e parâmetros para buscar dados, basta olhar na [documentação da biblioteca tweepy](#). Nós iremos utilizar o método **search()**, que funciona de maneira similar a própria

caixa de busca no site do Twitter.

try:

```
# Busca até 200 tweets utilizando a query informada
tweets = api.search(q = 'Bolsonaro', count = 200)

except tweepy.TweepError as e:
    print("Erro : " + str(e))
```

**DICA:** Estamos colocando todos trechos do processo entre blocos try/except, assim em caso de erro, temos certeza de qual ponto do processo está com problemas.

Vamos fazer uma função que através da pontuação de polaridade das palavras do tweet, obtido através do léxico do polyglot, calculamos a polaridade do texto. Utilizaremos uma fórmula bem simples, onde somaremos todas polaridades, se o resultado por positivo o sentimento é positivo, se for zero é neutro, caso menor que zero, negativo.

```
# Você pode fazer download dos léxicos de sentimento de cada idioma separadamente
!polyglot download sentiment2.en
!polyglot download sentiment2.tr
!polyglot download sentiment2.es
!polyglot download sentiment2.tk
!polyglot download sentiment2.fi
!polyglot download sentiment2.pt
!polyglot download sentiment2.de
!polyglot download sentiment2.it
!polyglot download sentiment2.lv
!polyglot download sentiment2.ca
!polyglot download sentiment2.la

[polyglot_data] Downloading package sentiment2.en to
[polyglot_data]      /root/polyglot_data...
[polyglot_data] Package sentiment2.en is already up-to-date!
```

```
[polyglot_data] Downloading package sentiment2.tr to
[polyglot_data]      /root/polyglot_data...
[polyglot_data]      Package sentiment2.tr is already up-to-date!
[polyglot_data] Downloading package sentiment2.es to
[polyglot_data]      /root/polyglot_data...
[polyglot_data]      Package sentiment2.es is already up-to-date!
[polyglot_data] Downloading package sentiment2.tk to
[polyglot_data]      /root/polyglot_data...
[polyglot_data]      Package sentiment2.tk is already up-to-date!
[polyglot_data] Downloading package sentiment2.fi to
[polyglot_data]      /root/polyglot_data...
[polyglot_data]      Package sentiment2.fi is already up-to-date!
[polyglot_data] Downloading package sentiment2.pt to
[polyglot_data]      /root/polyglot_data...
[polyglot_data]      Package sentiment2.pt is already up-to-date!
[polyglot_data] Downloading package sentiment2.de to
[polyglot_data]      /root/polyglot_data...
[polyglot_data]      Package sentiment2.de is already up-to-date!
[polyglot_data] Downloading package sentiment2.it to
[polyglot_data]      /root/polyglot_data...
[polyglot_data]      Package sentiment2.it is already up-to-date!
[polyglot_data] Downloading package sentiment2.lv to
[polyglot_data]      /root/polyglot_data...
[polyglot_data]      Package sentiment2.lv is already up-to-date!
[polyglot_data] Downloading package sentiment2.ca to
[polyglot_data]      /root/polyglot_data...
[polyglot_data]      Package sentiment2.ca is already up-to-date!
[polyglot_data] Downloading package sentiment2.la to
[polyglot_data]      /root/polyglot_data...
```

```
qtdeTweetsPos = 0
qtdeTweetsNeg = 0
qtdeTweetsNeu = 0
tweets_str_list = []

# Percorre os tweets encontrados
for tweet in tweets:

    #ATENÇÃO: Tenha certeza que já instalou e importou o polyglot acima!
```

```
text = Text(tweet.text)

somaP = 0

# Percorre as palavras do tweet
for palavra in text.words:
    # Soma polaridade da palavra
    somaP += palavra.polarity

# Faz contagem das polaridades
if somaP > 0:
    qtdeTweetsPos += 1
elif somaP < 0:
    qtdeTweetsNeg += 1
else:
    qtdeTweetsNeu += 1
```

```
# Imprime o tweet
print("Polaridade: " + str(somaP) + " >>> " + tweet.text + "\n")
```

```
tweets_str_list.append(tweet.text)
```

Polaridade: 0 >>> RT @FlavioBolsonaro: Cavalgada a favor de Bolsonaro, em Canhotinho (Pernambuco), no dia 25/Set/22.  
O Nordeste é Bolsonaro 22!

#BrasilVota2...

Polaridade: 0 >>> RT @siteptbr: 🚨 No Ceará, eleitor de Bolsonaro invade bar, pergunta quem vota no Lula e logo após mata pet  
A vítima era pa...

Polaridade: -1 >>> RT @GeorgMarques: BOMBA: Rachadinha presidencial!

A Polícia Federal encontrou no telefone do principal ajudante de ordens de Jair Bolsonar...

Polaridade: -2 >>> RT @GuilhermeBoulos: BOMBA! PF encontrou no telefone do principal ajudante de ordens de Bolsonaro indícios

Polaridade: 0 >>> RT @\_Estragao: 🚨#URGENTE: Após denúncia da ativista vegana cristã conservadora dos animais Luisa Mel, o TS

Polaridade: 0 >>> @GuilhermeBoulos @felipeneto Presidente Bolsonaro vence no Primeiro turno neste domingo e Ciro alcançará 30

Polaridade: -5 >>> RT @BlogdoNoblat: PF vê transações suspeitas em gabinete de Bolsonaro, e Alexandre de Moraes quebra sigilo

Polaridade: 0 >>> RT @revistaforum: Tico Santa Cruz posta foto com Lula: “É um voto para que possamos sobreviver”

Músico, que era um dos principais aliados...

Polaridade: 0 >>> Bolsonaro e Anderson Ferreira participam de motociata em Petrolina, nesta terça-feira  
#Eleicoes2022

<https://t.co/rYeJ0kaVlb>

Polaridade: -4 >>> RT @folha: EXCLUSIVO | PF vê transações suspeitas em gabinete de Bolsonaro, e Moraes quebra sigilo de asse

Polaridade: 3 >>> se tudo der certo:

o brasa é hexa

Bolsonaro no primeiro turno

são paulo ganha a sula



Polaridade: -2 >>> RT @GuilhermeBoulos: BOMBA! PF encontrou no telefone do principal ajudante de ordens de Bolsonaro indícios

Polaridade: 0 >>> 'Veio observar o quê?', provoca Bolsonaro sobre observadores da OEA que vieram ao Brasil para acompanhar el

Polaridade: 1 >>> RT @taoquei1: Xandeeeeee, seu abuso de autoridade te levou para o New York Times. Você tá internacional!!!!

Polaridade: -2 >>> Caso segundo turno for esse ... Vou de Leite por eliminação Onix é Bolsonaro e  
BOLSONARO NUNCA MAIS !!!... <https://t.co/FIIZhLgqLb>

Polaridade: 0 >>> RT @FlavioBolsonaro: Cavalgada a favor de Bolsonaro, em Canhotinho (Pernambuco), no dia 25/Set/22.  
O Nordeste é Bolsonaro 22!

#BrasilVota2...

Polaridade: -2 >>> RT @BrazilFight: BOLSONARO

"Eu vou estar agora com o chefe dos observadores Da Missão de Observação Eleitoral, que vai observar as eleiçõe...

Polaridade: -1 >>> RT @SandraNavajas: E seguimos resistindo ao fascismo de Bolsonaro e aos ataques covardes dessa tal esquerda

Polaridade: 2 >>> @thiagosgc2022 @OpiPolitical1 Eu também não gosto. Pro Lula tá na frente, ele tem que estar muito bem no Nor

**ATENÇÃO:** As vezes, o polyglot pode identificar de maneira errada o idioma do tweet, e neste caso tentará utilizar um recurso léxico que ainda não foi baixado. Você pode corrigir de duas maneiras, ou usa o try/except e ignora este tweet, ou baixa o léxico dos idiomas envolvidos e deixa que ele calcule de maneira errada mesmo.

```
# Agora calculamos as porcentagens
print("POSITIVOS: {}".format(100*qtdeTweetsPos/len(tweets)))
print("NEGATIVOS: {}".format(100*qtdeTweetsNeg/len(tweets)))
print("NEUTROS: {}".format(100*qtdeTweetsNeu/len(tweets)))

POSITIVOS: 17.0%
NEGATIVOS: 51.0%
NEUTROS: 32.0%

import matplotlib.pyplot as plt

# Podemos plotar um gráfico também
fig=plt.figure(figsize=(6,6))
plt.pie([qtdeTweetsPos, qtdeTweetsNeg, qtdeTweetsNeu],labels=['Positivo', 'Negativo', 'Neutro'],colors=['green', 'red', 'blue'])
plt.ylabel('')
plt.title('Polaridade', fontsize='18')
plt.show()
```

## Polaridade



Poderíamos também construir uma WordCloud para entender melhor quais as palavras que estão mais associadas ao tema buscado.

Para isso iremos remover as stop-words e chamar uma função de WordCloud.

```
import nltk
nltk.download('stopwords')
import matplotlib.pyplot as plt
from wordcloud import WordCloud

stopword= nltk.corpus.stopwords.words('portuguese')

# Retiramos ocorrências de links e retweets
stopword.append('https')
stopword.append('RT')
stopword.append('co')

# Cria uma única string com todos tweets
str_tweets = " ".join(tweets_str_list)

# Gera a wordcloud
wordcloud = WordCloud(max_words=2000, max_font_size=90, stopwords=stopword, height=400, width=800).generate(str_tweets)

fig = plt.figure(figsize=[20,10])
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title('WordCloud Twitter', fontsize='18')
plt.show()
```

[nltk\_data] Downloading package stopwords to /root/nltk\_data...

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

WordCloud Twitter



## Considerações finais

Aqui utilizamos uma abordagem BEM simples, baseada em um léxico de polaridade da biblioteca polyglot. Os textos provindos de redes sociais são muito difíceis de processar, o ideal seria uma abordagem mais complexa, de preferência com aprendizado supervisionado, capaz de lidar com algumas características e desafios destes textos:

- Uso de gírias
- Ausência de regras ortográficas
- Uso de símbolos (emojis, hashtags)
- Presença de ironia
- Uso de hiperlinks
- entre outros

## Atividade complementar

Você pode deixar o algoritmo mais robusto, ao treinar um classificador utilizando as bases rotuladas de tweets em português, listadas nas referências deste notebook.

## ▼ Referências e Material complementar

- [Portuguese Tweets for Sentiment Analysis - Corpus anotado](#)
- [Portuguese Tweets for Sentiment Analysis using nltk and sklearn](#)
- [tweetSentBR - Corpus anotado](#)
- [Anotando um Corpus de Notícias para a Análise de Sentimento: um Relato de Experiência](#)
- [Twitter Sentiment Analysis using NLTK, Python](#)
- [Creating The Twitter Sentiment Analysis Program in Python with Naive Bayes Classification](#)
- [Sentiment Analysis with Python \(Part 1\) - Classifying IMDb Movie Reviews](#)
- [Twitter Developer API](#)
- [Sentiment Classification using Word Embeddings \(Word2Vec\)](#)
- [Sentiment Analysis on Tweets in Portuguese - Transformer and BERT](#)

Esta aula foi inspirada no trabalho do [Prof. Emerson Cabrera Paraiso](#), a quem agradecemos pela permissão de uso de sua [base de dados](#).

Este notebook foi produzido por Prof. [Lucas Oliveira](#).

Produtos pagos do Colab - Cancelar contratos

✓ 4s conclusão: 19:28

