Instituto Federal do Tocantins - Campus Paraíso do Tocantins
Paraíso do Tocantins - TO - Brasil
Sistemas de Informação 6ºP

Twitter Sentiment Analysis

Jefté Lopes G, Lucas Lopes, Edivaldo Araújo Jr.

Resumo

O objetivo desse estudo de caso é criar um modelo que analisa um ou mais Tweets para prever o sentimento(Positivo ou Negativo) presente em cada Tweet é usado o Processamento de linguagem natural (NLP) juntamente com aprendizagem de maquina para construir esse modelo. Buscamos apresentar tudo com uma didática simplificada para maximizar a medida do possível o entendimento do assunto, que a princípio pode parecer complexo.

Abstract

The purpose of this case study is to create a model that analyzes one or more Tweets to predict the feeling (Positive or Negative) present in each Tweet. Natural language processing (NLP) is used in conjunction with machine learning to build this model. We seek to present everything with a simplified didactic to maximize the extent of the possible understanding of the subject, which at first may seem complex.

1. Problema

Classificar os sentimentos do Twitter como Negativo ou positivo. O objetivo desse estudo de caso é criar um modelo que analisa um ou mais Tweets para prever o sentimento(Positivo ou Negativo) presente em cada Tweet.

1.1. 1ºSolução e 2ºSolução

Escolhemos realizar uma segunda solução pelo fato de que os resultados obtidos e aprendidos durante a primeira solução não foram satisfatórios, portanto, buscamos uma nova solução que em tese traria um melhor resultado a explicção das duas soluções estão detalhadas no documento.

1º Solução	2º Solução
Base de dados	Base de dados
Idioma Inglês	Idioma Português-Brasil
Utilizando tweets	Utilizando tweets
Baixada no kaggle	Criada
NLP(Natural Language Processing)	NLP(Natural Language Processing)
Bag of Word	Bag of Word
Naive Baye	Naive Baye
Visualização dos Resultado	Visualização dos Resultado

1. **1º Solução**
    - Entender a Declaração do Problema e o caso de negócios.
    - Importar bibliotecas e conjuntos de dados.
    - Executar a análise exploratória dos dados.
    - Plotar a nuvem de palavras.
    - Executar a limpeza de dados - remover pontuação.
    - Executar a limpeza de dados - remover palavras de parada(stop words).
    - Executar vetorização de contagem (Tokenization).
    - Criar um pipeline para remover palavras irrelevantes, pontuação e realizar tokenização.
    - Compreender a teoria e a intuição por trás dos classificadores Naive Bayes.
    - Treinar um Classificador Naive Bayes.
    - Avaliar o desempenho do modelo treinado.
2. **2º Solução**
    - Entender a Declaração do Problema e o caso de negócios.
    - Importar bibliotecas e conjuntos dos dados.
    - Executar a análise exploratória de dados.
    - Plotar a nuvem de palavras.
    - Executar vetorização de contagem (Tokenization).
    - Treinar um Classificador Naive Bayes.
    - Avaliar o desempenho do modelo treinado.
    - Salvar o modelo treinado.

2. Entender a Declaração do Problema e o caso de negócios

Esse documento foi criado, para informar o passo a passo feito com base no nosso notebook que é usado o Processamento de linguagem natural(NLP) juntamente com aprendizagem de maquina para construir um modelo que analisa milhares de Tweets para prever o sentimentos das pessoas.

A Inteligencia artificial e a analise de sentimentos baseadas em aprendizado de maquina é crucial para empresas, visto que, os insight revelado pela analise visa indicar o grau de qualidade dos serviços e/ou produtos da empresa de acordo com os clientes.(Trabalhos Futuros)

Esse projeto é diretamente aplicável a praticamente qualquer empresa que disponha de meios onlines(Twitter, Instagran, Facebook, WebSite) para interagir com seus clientes.(Trabalhos Futuros)

Os algoritmos podem ser usados para detectar e possivelmente sinalizar automaticamente tweets de ódio e racismo.(Trabalhos Futuros)

2.1. Importar bibliotecas e conjuntos de dados

Com base no nosso notebook explicarei passo a passo sobre a 1º Solução, nela foram utilizados Pacotes essenciais para analise numéricas, manipulação de data frames e visualização de dados.

1. Bibliotecas de apoio:

```
#Pacotes essenciais para analise numericas
#manipulação de data frames e visualização de dados.
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

pandas para carregar e manipular os dados.

Numpy é usado para trabalhar com matrizes. Também possui funções para trabalhar no domínio da álgebra linear, transformada de Fourier e matrizes.

pyplot from matplotlib é usado para visualizar os resultados.

Seaborn é uma biblioteca de visualização de dados Python baseada em matplotlib. Ele fornece uma interface de alto nível para desenhar gráficos estatísticos atraentes e informativos.

2. Pacote para a criação de nuvens de palavras:

```
In [ ]: # Pacote para a criação de nuuvens de palavras
        !pip install WordCloud
        from wordcloud import WordCloud
```

A base de dados desse exemplo está no meu repositório do github e pode ser baixada no link abaixo.

`https://raw.githubusercontent.com/JefteLG/Twitter_Sentiment_Analysis/main/Data_Set/data_base_estudos/twitter.csv`

3. Utilizando o pandas para ler o arquivo CSV

```
#estruturando em um data frame na variavel `tweets_df` por meio do metodo read_csv.
tweets_df = pd.read_csv('/tmp/twitter.csv')

#visualização1.
tweets_df.head()

#visualização2.
tweets_df

#Informações sobre a quantidade de Tweets, memoria, tipo de dados e dados faltantes
tweets_df.info()
```

3. Executar a análise exploratória de dados

Esta é uma função no nível dos eixos e desenhará o mapa de calor para os eixos ativos no momento. Nesse caso a função verifica se existe dados faltantes.

```
sns.heatmap(tweets_df.isnull(), yticklabels=False, cbar=False, cmap="Blues")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3fdbd2e208>
```

O método sns.countplot() é usado para mostrar as contagens de observações em cada categoria categórica usando barras(Histograma).

```
sns.countplot(data=tweets_df, x='label', palette='Set2')
plt.show()
```

```
#Nova coluna com o tamanho dos tweets
tweets_df['length'] = tweets_df['tweet'].apply(len)
```

1. Teste:

```
from nltk.tokenize import TweetTokenizer
tweets_df[tweets_df['id']==31958]['tweet'].iloc[0]
tweetTK = TweetTokenizer()
tweetTK.tokenize(tweets_df[tweets_df['id']==31958]['tweet'].iloc[0])
```

2. Fim Teste:

```
#Descobrir o tamanho maximo, minimo e medio dos tweets.
tweets_df['length'].plot(bins=100, kind='hist', figsize=(12,8), color='g')
tweets_df.describe()

```

```
# Selecionar a menor frase
tweets_df[tweets_df['length']==11]['tweet'].iloc[0]

# Separa o DataFrame em dois Dataframes, um com sentimentos
#positivos e o outro com sentimentos negativos.
positive_df = tweets_df[tweets_df['label']==0]
negative_df = tweets_df[tweets_df['label']==1]

positive_df

negative_df
```

3.1. Plotar a nuvem de palavras

Wordcloud – em português, nuvem de palavras ou nuvem de tags – é um tipo de visualização de dados muito poderoso e ferramenta de Data Science usado quando estamos trabalhando com textos, documentos, pesquisas, entre outras.

Resumidamente, é como se você estivesse contando a frequência com que cada palavra aparece em um texto. Com essa frequência, você define tamanhos proporcionais às palavras e quanto maior for a ocorrência da palavra no texto maior será na nuvem ou desenho.

Segue abaixo a implementação:

1. Criar uma lista de tweets:

```
sentences = tweets_df['tweet'].tolist()
```

2. O método join() pega todos os itens em um iterável e os une em uma string é usado o espaço como separador

```
sentences_as_one_string = " ".join(sentences)
```

3. Nuvem de palavras de todos os tweets NEGATIVOS:

```
negative_sentences = negative_df['tweet'].tolist()
negative_as_one_string = " ".join(negative_sentences)

plt.figure(figsize=(10,10))
plt.imshow(WordCloud().generate(negative_as_one_string))
```

4. Nuvem de palavras de todos os tweets POSITIVOS:

```
positive_sentences = positive_df['tweet'].tolist()
positive_as_one_string = " ".join(positive_sentences)

plt.figure(figsize=(10,10))
plt.imshow(WordCloud().generate(positive_as_one_string))
```

4. Processamento de Linguagem Natural

Processamento de linguagem natural (PLN) é uma vertente da inteligência artificial que ajuda computadores a entender, interpretar e manipular a linguagem humana.

A linguagem humana é surpreendentemente complexa e diversa. Nós nos expressamos de infinitas maneiras, tanto verbalmente quanto por escrito. Não apenas existem centenas de idiomas e dialetos, como há também um conjunto único de regras gramaticais e de sintaxe, expressões e gírias dentro de cada um deles. Quando escrevemos, costumamos cometer erros ou abreviar palavras, ou omitimos pontuações quando falamos, carregamos sotaques regionais, tendemos a murmurar e emprestamos termos de outros idiomas.

Embora o aprendizado supervisionado, o aprendizado não-supervisionado e, especificamente, o deep learning sejam hoje amplamente utilizados para modelar a linguagem humana, há também a necessidade de compreensão sintática e semântica, além de domínio, que não estão necessariamente presentes nessas abordagens de machine learning. O PLN é importante porque ajuda a resolver a ambiguidade na linguagem e adiciona uma estrutura numérica útil aos dados para muitas aplicações downstream, como reconhecimento de fala ou análise de texto.

4.1. Executar a limpeza de dados - remover pontuação

Vamos criar uma função para tratar nossos dados, removendo símbolos e caracteres especiais, essa remoção tem como objetivo ter uma melhor precisão nas previsões.

Nesta etapa vamos começar a “limpar” nosso texto, removendo tudo que não tem valor para nosso objetivo.

Para fazermos essa limpeza exitem vários métodos, nós utilizamos cinco que será mostrado a implementação logo abaixo:

```
import string
text = 'Good morning beautiful people :)...
'I am having fun learning Machine learning and artificial intelligence'
```

1. Método Nº1:

```
test_punc_remove = [char for char in text if char not in
   string.punctuation]
test_punc_remove
test_punc_remove = ''.join(test_punc_remove)
test_punc_remove
```

2. Método Nº1.1:

```
text_remove_punct = []
 for char in text:
  if char not in string.punctuation:
     text_remove_punct.append(char)

new_text = ''.join(text_remove_punct)

new_text
```

3. Método Nº2:

```
punct = string.punctuation + string.digits
table_tst = str.maketrans('','',punct)
newtext = text.translate(table_tst)
newtext
```

4. Método Nº3:

```
punct = string.punctuation + string.digits
table_ = str.maketrans(punct, ' '*len(punct))
newtext = ' '.join(text.translate(table_).split())
newtext
```

5. Método Nº4:

```
punct = string.punctuation + string.digits
   for s in punct:
    text = text.replace(s, '')
text
```

6. Método Nº5:

```
import re
newtext = re.sub(r'[^A-Za-z]+', ' ', text)
newtext
'Good morning beautiful people I am having fun learning
'Machine learning and artificial intelligence'
```

4.2. Executar a limpeza de dados - remover palavras de parada(stop words)

As palavras irrelevantes são frequentemente removidas do texto antes do treinamento de modelos de aprendizado profundo e de aprendizado de máquina, já que as palavras irrelevantes ocorrem em abundância, fornecendo, portanto, pouca ou nenhuma informação exclusiva que pode ser usada para classificação ou agrupamento.

Usando a biblioteca NLTK do Python é uma das bibliotecas Python mais antigas e mais comumente usadas para processamento de linguagem natural. O NLTK oferece suporte à remoção de palavras de interrupção e você pode encontrar a lista de palavras de interrupção no corpusmódulo. Para remover palavras irrelevantes de uma frase, você pode dividir seu texto em palavras e, em seguida, remover a palavra se ela sair da lista de palavras irrelevantes fornecida pelo NLTK.

Vamos ver na prática:

```
#Pacotes para facilitar o processamento de linguagem natural
import nltk
import string
nltk.download('stopwords')

from nltk.corpus import stopwords
stopwords.words('english')

```

```
text = 'Good morning beautiful people :)... I am having fun
'learning Machine learning and artificial intelligence in 2020'
#caracteres para remover
puncts = string.digits + string.punctuation

#remoção de caracteres
text_remove_punct = [char for char in text if char not in puncts]

#pega todos os itens em um iterável e os une em uma string.
text_remove_punct = ''.join(text_remove_punct)

text_remove_punct

#remoção de stopword
text_remove_stopword = [word for word in text_remove_punct.split()
if word.lower() not in stopwords.words('english')]

text_remove_stopword
```

Pipeline remover pontuaçoes e stopwords, Pipelines são interessantes para reduzir código e automatizar fluxos é também executa uma série de ações que normalmente são definidas por funções e/ou métodos, a execução ocorre de maneira linear.

```
def pipeline(text):
  #frase sem nenhum tratamento
  print(text)
  #remover as pontuações
  text_remove_punct = [char for char in text if char not in puncts]
  #unir todos os elemntos da lista
  text_remove_punct = ''.join(text_remove_punct)
  #frase com pontuações removidas
  print(text_remove_punct)

  # remover as stopword
  text_remove_stopword = [word for word in text_remove_punct.split()
  if word.lower() not in stopwords.words('english')]
  text_remove_stopword = ' '.join(text_remove_stopword)
  #frase com stopwords removidas
  print(text_remove_stopword)

  pipeline(text)

```

4.3. Executar vetorização de contagem(Tokenization)

A tokenização, também conhecida como segmentação de palavras, quebra a sequência de caracteres em um texto localizando o limite de cada palavra, ou seja, os pontos onde uma palavra termina e outra começa [Palmer, 2010]. Para fins de linguística computacional, as palavras assim identificadas são frequentemente chamadas de *tokens*.

Quando a linguagem escrita é armazenada em um arquivo de computador, ela é normalmente representada por meio de uma sequência ou string (do inglês, "cadeia") de caracteres. Isto é, em um arquivo de texto padrão, as palavras são strings, as sentenças são strings e o próprio texto não passa, no fundo, de uma longa string. Os caracteres de uma string não precisam ser necessariamente alfanuméricos; estas também podem incluir caracteres especiais que representem os espaços, as tabulações, os sinais de nova linha, etc.

Muito do processamento computacional é realizado acima do plano dos caracteres. Quando compilamos uma linguagem de programação, por exemplo, o compilador espera que o input (os dados de entrada) seja uma sequência de tokens com os quais ele seja capaz de lidar; por exemplo, as classes dos identificadores, constantes textuais e alfanuméricas. De forma análoga, um parser espera que seu input seja uma sequência de tokens de palavras e não uma sequência de caracteres individuais.

Em sua forma mais simples, a tokenização de um texto envolve a busca das posições da string que contem os chamados "caracteres em branco"(espaços, tabulações ou sinais de nova linha) ou sinais de pontuação específicos, separando a string em tokens nestas posições.

Metodologia: Bag of Word, Bag of words é um método usado para extrair recursos de documentos de texto. Esses recursos são utilizados para treinar algoritmos de aprendizagem de máquina. Ele cria um vocabulário de todas as palavras do conjunto de treinamento.

Um dos maiores problemas com o texto é que ele é confuso e não estruturado, e os algoritmos de aprendizado de máquina preferem entradas de comprimento fixo bem definidas e estruturadas, utilizando a técnica Bag-of-Words, podemos converter textos, em um comprimento fixo vetor. Além disso, os modelos de aprendizado máquina funcionam com dados numéricos em vez de dados textuais, então para ser mais específico, usando a técnica do saco de palavras (BoW), é possível converter texto em vetores para facilitar o trabalho com números equivalentes.

Veremos agora na prática:

```
#Classe responsável por Converter uma coleção de documentos de texto
#em uma matriz de contagens de tokens
from sklearn.feature_extraction.text import CountVectorizer
sample_data=[
             'Porcaria de produto',
             'Obrigado pelo retorno estou muito satisfeito com seu trabalho?',
             'Voces ainda vão me responder?',
             'O pior serviço de atendimento de todos!!!'
             ]

sample_data2=[
```

```
                'Hello World',
                'Hello Hello hello World',
                'Hello World world world'
                ]

# Aprender o vocabulario do texto e retornar um array, [n_samples, n_features]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(sample_data)
print(vectorizer.get_feature_names())


# versao tokenizada das strings no dataset
# (id palavra, id frase)  frequencia da palavra na frase
c = X
print(c)

print(vectorizer.vocabulary_)
{'porcaria': 11, 'de': 4, 'produto': 12, 'barato': 2, 'ruim': 15, 'obrigado': 8,
'pelo': 9, 'retorno': 14, 'estou': 5, 'muito': 7, 'satisfeito': 16, 'com': 3,
'seu': 18, 'trabalho': 20, 'voces': 21, 'ainda': 0, 'vão': 22, 'me': 6,
'responder': 13, 'pior': 10, 'serviço': 17, 'atendimento': 1, 'todos': 19}

#Retorne uma representação densa desta matriz.
print(X.toarray())
```

4.4. Criar um pipeline para remover palavras irrelevantes, pontuação e realizar tokenização

1. Limpar as mensagens:

```
def message_cleaning(msg):
```

2. Remoção das pontuações:

```
text_remove_punct = [char for char in msg if char not in char_remove]
text_remove_punct = ''.join(text_remove_punct)
```

3. Remoção das palavras de parada:

```
text_remove_stopword = [word for word in text_remove_punct.split()
if word.lower() not in stopwords.words('english')]
# text_remove_stopword = ' '.join(text_remove_stopword)
return text_remove_stopword
```

4.5. Treinar um Classificador Naive Bayes

Em termos simples, um classificador Naive Bayes assume que a presença de uma característica particular em uma classe não está relacionada com a presença de qualquer

outro recurso. O modelo Naive Bayes é fácil de construir e particularmente útil para grandes volumes de dados.

Classificadores Naive Bayes utilizados principalmente em classificação de textos (devido a um melhor resultado em problemas de classes múltiplas e regra de independência) têm maior taxa de sucesso em comparação com outros algoritmos. E bastante utilizado em Análise de Sentimento que e o nosso caso, para identificar sentimentos positivos e negativos.

Segue a implementação de treino:

```
X.shape
(31962, 47386)
y.shape
(31962,)
```

1. Separação dos dados de treino e teste:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

2. Utilização do algoritmo de Naive Bayes para o treinamento:

```
from sklearn.naive_bayes import MultinomialNB
NB_classifier = MultinomialNB()
NB_classifier.fit(X_train, y_train)
```

5. Avaliar o desempenho do modelo treinado

```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
y_predict_test = NB_classifier.predict(X_test)
cm = confusion_matrix(y_test, y_predict_test)
cm
# sns.heatmap(cm, annot=True)
print(classification_report(y_test, y_predict_test))
```

6. Tópicos relacionado a 2º Solução

A partir de agora serão explicados os tópicos relacionados apenas a 2º solução, os tópicos abordados serão Criação da base de dados (feita por nós), Análise exploratória dos dados, Avaliar o desempenho do modelo treinado e Salvar o modelo treinado.

6.1. Criação da base de dados utilizando a API do Twitter com python pelo tweepy

Para ser possível acessar a api é necessário ter uma conta de desenvolvedor e a conta em questão foi criada por fins acadêmicos no intuito de desenvolver um modelo de aprendizagem para reconhecimentos de sentimentos positivos e negativos a partir de tweets.

Sobre o acesso às informações coletadas segundo as diretrizes de privacidade do twitter não é possível disponibilizar dados referentes aos usuários sendo que o não cumprimento do mesmo é passivo a revogação da licença de desenvolvedor é perdendo assim o acesso a api.

6.2. Sobre o desenvolvimento do algoritmo para a criação do dataset:

Todo o algoritmo foi elaborado em python, e primeiramente foi elaborado com uma integração a um banco de dados mysql, porém pela demora na inserção e validação dos dados decidimos utilizar-se do pandas para a criação do dataset em um documento CSV.

Sobre o método de tratamento dos dados após ser coletados do twitter utilizamos de expressões regulares para fazer uma limpeza em cada tweet retirando links, nomes de usuários, pontuações, espaços em excesso e por fim transformando cada caractere para caixa baixa(lower case).

6.3. Definição dos termos a serem buscados no twitter:

A princípio foi definido somente emojis para a classificação de tweets sendo :) para positivo e :(para negativos e com a melhor compreensão da api as buscas foram feitas em cima das seguintes combinações:

1. Para tweets positivos:
    - :) feliz
    - :) alegre
    - \ø/ empolgado
    - :) amor
    - \ø/ confiante
    - :) apaixonado
    - \ø/ otimista
    - resiliência
2. Para tweets negativos
    - :(triste
    - :-(triste

- :(mau
- :(morrer

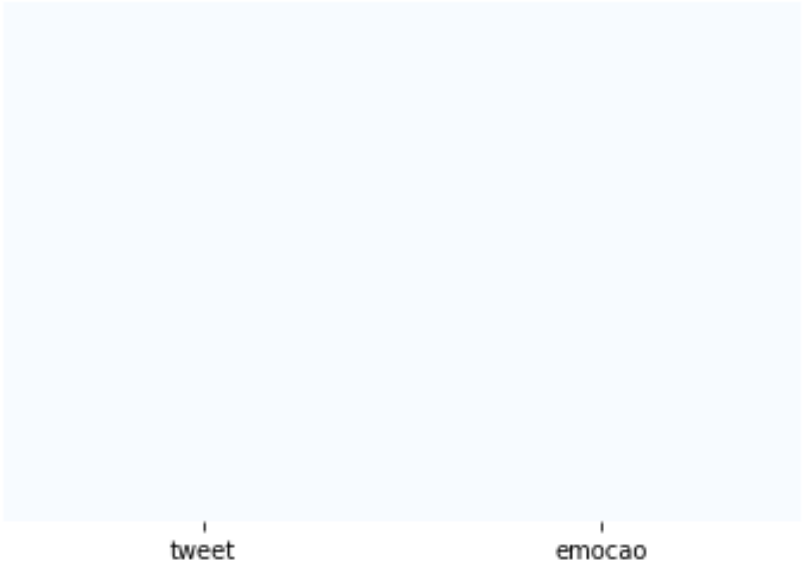
Grande parte dos tweets eram em sua maioria negativos, sendo necessário incluir mais tweets positivos para igualar a base de dados e acabando tendo mais instruções de pesquisa positivas do que negativas.

7. Análise exploratória dos dados

1. Esta é uma função no nível dos eixos e desenhará o mapa de calor para os eixos ativos no momento. Nesse caso a função verifica se existe dados faltantes.

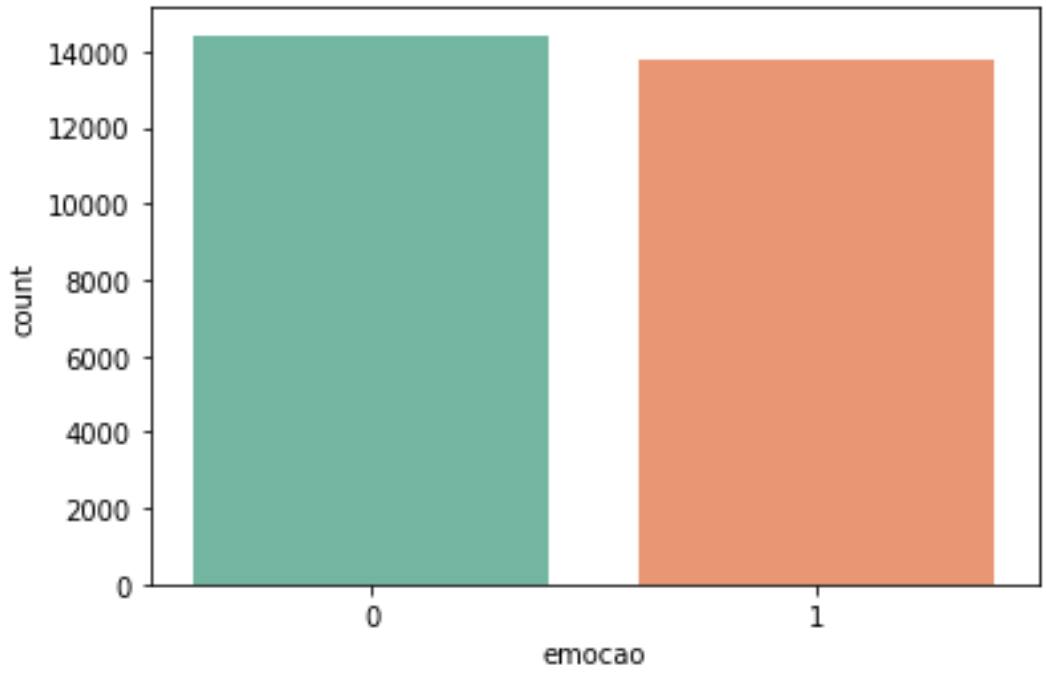
```
sns.heatmap(tweets_df_dataframe.isnull(), yticklabels=False, cbar=False, cmap="
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f459cc58438>
```



2. O método sns.countplot() é usado para mostrar as contagens de observações em cada categoria categórica usando barras(Histograma).

```
sns.countplot(data=tweets_df_dataframe, x='emocao', palette='Set2')
plt.show()
```



3. Separar o DataFrame em dois Dataframes, um com sentimentos positivos e o outro com sentimentos negativos.

```
positive_df = tweets_df_dataframe[tweets_df_dataframe['emocao']==1]
negative_df = tweets_df_dataframe[tweets_df_dataframe['emocao']==0]
```

8. Avaliar o desempenho do modelo treinado

```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
```

1. Visualização da matriz de confusão:

```
y_predict_test = NB_classifier.predict(X_test)
cm = confusion_matrix(y_test, y_predict_test)
cm
# sns.heatmap(cm, annot=True)
array([[2779,   73],
       [ 201, 2585]])
```

2. Métricas:

```
print(classification_report(y_test, y_predict_test))
 precision    recall  f1-score   support

           0       0.93      0.97      0.95      2852
           1       0.97      0.93      0.95      2786

    accuracy                           0.95      5638
   macro avg       0.95      0.95      0.95      5638
weighted avg       0.95      0.95      0.95      5638
```

8.1. Testes e Classificações

1. Quantidade de classificações certas:

```
tweet_correto = (y_test == NB_classifier.predict(X_test)).sum()
print(f'{tweet_correto} classificações certas')

5364 classificações certas
```

2. Quantidade de classificações incorretas:

```
tweet_incorreto = (y_test.size - tweet_correto)
print(f'{tweet_incorreto} classificações incorretas')
274 classificações incorretas
```

3. Porcentagem de accuracy:

```
error = tweet_incorreto/(tweet_correto + tweet_incorreto)
print(f'acuraccy do modelo {1-error:.2%}')
acuraccy do modelo 95.14%
```

4. Accuracy:

```
NB_classifier.score(X_test, y_test)
0.9547711954593827
```

Exemplos

```
exemplos = [
      'O meu dia hoje foi pessimo',
      'Minha cabeça está doendo muito',
      'Eu acho que me dei muito mal na minha entrevista de emprego hoje',
      'acordei feliz mas meu dia acabou muito triste',
      'eu quero morrer',
      'vou matar o',
      'acordei mó triste mas meu dia acabou muito feliz',
      'estou bastante otimista hoje',
      'Vou ter um irmãozinhooooooo eba',
      'Vou ser mamãe, estou muito contente',
      'Me desejem sorte',
      'o é meu melhor amigo',
      'fiquei na bad',
      'estou hypado',
      'estou no hype'
      ]

exemplos2 = [
            'Gosto de um bom churrasco'
]
```

1. Tokenizar os exemplos:

```
exemplos = vectorizer.transform(exemplos)
exemplos2 = vectorizer.transform(exemplos2)
print(exemplos.toarray())

t = exemplos.toarray()
print(t[0][10:40])
```

2. classificar o sentimento dos exemplos:

```
NB_classifier.predict(exemplos)
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1])

NB_classifier.predict(exemplos)
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1])
```

3. **TESTE 1:**

```
[39] y_predict_test = NB_classifier.predict(X_test)
     cm = confusion_matrix(y_test, y_predict_test)
     cm
     # sns.heatmap(cm, annot=True)

     array([[2790,   76],
            [ 170, 2606]])

[33] print(classification_report(y_test, y_predict_test))

                  precision    recall  f1-score   support

               0       0.94      0.98      0.96      2881
               1       0.98      0.94      0.96      2761

        accuracy                           0.96      5642
       macro avg       0.96      0.96      0.96      5642
    weighted avg       0.96      0.96      0.96      5642
```

```
[44] NB_classifier.predict(exemplos)

     array([1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1])

[24] NB_classifier.predict(exemplos)

     array([0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1])
```

4. **TESTE 2:**

```
[80] y_predict_test = NB_classifier.predict(X_test)
     cm = confusion_matrix(y_test, y_predict_test)
     cm
     # sns.heatmap(cm, annot=True)

     array([[2828,   70],
            [ 174, 2570]])
```

```
[88] print(classification_report(y_test, y_predict_test))

                  precision    recall  f1-score   support

               0       0.94      0.98      0.96      2898
               1       0.97      0.94      0.95      2744

        accuracy                           0.96      5642
       macro avg       0.96      0.96      0.96      5642
    weighted avg       0.96      0.96      0.96      5642
```

```
[94] NB_classifier.predict(exemplos)

     array([0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1])
```

```
[24] NB_classifier.predict(exemplos)

     array([0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1])
```

9. Salvar o modelo treinado

Salvamos o modelos de machine learning com Pickle, ele permite a serialização de objetos, ou seja, os transforma em sequências de bytes. Armazenando toda informação necessária, consegue reconstruir objetos aplicando uma lógica interna.

Para salvar um modelo, usa-se o método dump() – basta informar a variável referente ao objeto e o nome do arquivo a ser criado.

```
import pickle
#salvar o modelo Naive Bayes (NB_classifier) no arquivo model_1.pkl
with open('/tmp/model_2.pkl', 'wb') as file:
    pickle.dump(NB_classifier, file)
```

1. Carregando modelos de machine learning com Pickle, já que o modelo está treinado e validado, uma vez carregado ele já esta pronto para ser usado. No caso da célula abaixo, o modelo de machine learning foi atribuído à variável model.

```
with open('/tmp/model_1.pkl', 'rb') as f:
    model = pickle.load(f)
```

2. Classificar o sentimento:

```
model.predict(exemplos2)
```

10. Conclusão

Neste trabalho foi apresentada uma ferramenta para análise de sentimentos em mensagens enviadas através do Twitter para conhecer o que os usuários da rede social estão sentindo sobre determinados assuntos em relação aos sentimentos positivos e negativos presentes nas mensagens.A principal função da ferramenta e comparar os sentimentos das mensagens entre duas consultas sobre o sentimento presente nas mensagens(positivo e negativo).

Ainda existe muito espaço para a ferramenta evoluir, mas existem dois pontos principais que precisam ser melhorados. O primeiro está relacionado a qualidade do classificador. Um estudo maior precisa ser realizado para que se consiga resultados melhores, pois a análise de sentimentos ainda é um problema difícil para os classificadores que nós utilizamos neste estudo. O segundo ponto que precisa ser melhorado é a infraestrutura para suportar muitas consultas, pois o Twitter impõe limitações de uso de sua API.

11. Possíveis trabalhos futuros

A Inteligência artificial e a analise de sentimentos baseadas em aprendizado de maquina é crucial para empresas, visto que, os insight revelado pela analise visa indicar o grau de qualidade dos serviços e/ou produtos da empresa de acordo com os clientes.

Esse projeto é diretamente aplicável a praticamente qualquer empresa que disponha de meios online(Twitter, Instagram, Facebook, Website) para interagir com seus clientes.

Os algoritmos podem ser usados para detectar e possivelmente sinalizar automaticamente tweets de ódio e racismo.

12. Bibliografia

- **Pandas Disponível em:**

https://pandas.pydata.org/docs/

- **Numpy Disponível em:**

https://numpy.org/doc/stable/

- **Seaborn Disponível em:**

https://seaborn.pydata.org/introduction.html

- **Matplotlib Disponível em:**

https://matplotlib.org/3.3.3/contents.html

- **Scikit-Learn Disponível em:**

https://scikit-learn.org/stable/tutorial/index.html

- **Natural Language Toolkit (NLTK) Disponível em:**

https://www.nltk.org/

- **pickle Disponível em:**

https://docs.python.org/3/library/pickle.html

- **python Disponível em:**

https://www.python.org/doc/

- **Bag of Words (BoW) Disponível em:**

https://www.mygreatlearning.com/blog/bag-of-words/

- **Naive Bayes Disponível em:**

https://scikit-learn.org/stable/modules/naive_bayes.html