Análise de sentimentos do Twitter em relação a covid-19 e a comparação de algoritmos classificadores

Rômulo Rodrigues Coutinho

Objetivo

Em um momento único, onde o mundo sofre a maior crise sanitária do século, as redes sociais se tornaram um campo fértil de dados, onde os usuários expressam seus sentimentos sobre a pandemia. Desta forma, surge a necessidade de extrair conhecimento sobre estes dados, bem como observar quais os métodos na programação se mostram mais eficientes e o comportamento dos mesmos neste tipo de análise.

O objetivo deste trabalho é observar como se comportam algoritmos classificadores a partir de sua aplicação em uma base rotulada, dessa forma, são utilizados técnicas de *Data Mining* e *Machine Learning* no desenvolvimento do trabalho.

Desenvolvimento

O desenvolvimento deste trabalho é composto pelas seguintes etapas:

- Extração de Tweets
- Rotulação do sentimento
- Limpeza de dados
- Vetorização e Tokenização dos dados
- Aplicação dos algoritmos classificadores
- Gerar métricas de avaliação dos algoritmos

Extração dos Tweets

Primeiramente, é preciso importar as bibliotecas necessárias para extração e manipulação dos dados

```
In [1]: import tweepy as tw
        import pandas as pd
```

Utilizando a biblioteca Tweepy, é feito a autenticação com a API do Twitter

```
In [44]: auth = tw.AppAuthHandler(consumerKey, consumerSecret)
         api = tw.API(auth)
```

É definido as palavras-chaves, remoção de retweets e utilização do método Cursor para extração dos tweets

```
In [53]: search_words = 'pandemia' + '-filter:retweets'
         language = 'pt'
```

```
In [54]: tweets = tw.Cursor(api.search,
                            q=search_words,
                            lang = language,
                            tweet_mode="extended").items(10)
```

Palavras-chaves utilizadas: pandemia, covid-19, #covid-19, coronavírus, corona vírus, corona, covid

```
In [55]: for tweet in tweets:
             print(tweet)
```

```
Status(_api=<tweepy.api.API object at 0x0000028A943E46D0>, _json={'created_at': 'Fri Oct 30 19:53:25 +0000 2020', 'id': 13222
65372091928578, 'id_str': '1322265372091928578', 'full_text': 'Eleições em meio a pandemia! Fica a pergunta! Vale a pena corr
er risco de infecção para votar nesta classe de políticos de merda que nos temos?', 'truncated': False, 'display_text_range':
[0, 143], 'entities': {'hashtags': [], 'symbols': [], 'user_mentions': [], 'urls': []}, 'metadata': {'iso_language_code': 'p
t', 'result_type': 'recent'}, 'source': '<a href="https://mobile.twitter.com" rel="nofollow">Twitter Web App</a>', 'in_reply_
to_status_id': None, 'in_reply_to_status_id_str': None, 'in_reply_to_user_id': None, 'in_reply_to_user_id_str': None, 'in_rep
ly_to_screen_name': None, 'user': {'id': 62364521, 'id_str': '62364521', 'name': 'Juba Galo', 'screen_name': 'JUBA_GALO', 'lo
cation': 'Brasil', 'description': 'Verás que um filho teu não foge à luta BR oh pátria amada!', 'url': None, 'entities': {'des
cription': {'urls': []}}, 'protected': False, 'followers_count': 1957, 'friends_count': 1919, 'listed_count': 6, 'created_a
t': 'Sun Aug 02 22:18:41 +0000 2009', 'favourites_count': 26085, 'utc_offset': None, 'time_zone': None, 'geo_enabled': True,
'verified': False, 'statuses_count': 49597, 'lang': None, 'contributors_enabled': False, 'is_translator': False, 'is_translat
ion_enabled': False, 'profile_background_color': '01070A', 'profile_background_image_url': 'http://abs.twimg.com/images/theme
s/theme1/bg.png', 'profile_background_image_url_https': 'https://abs.twimg.com/images/themes/theme1/bg.png', 'profile_backgro
und_tile': True, 'profile_image_url': 'http://pbs.twimg.com/profile_images/1056520922117607424/O-Gh6FSF_normal.jpg', 'profile
_image_url_https': 'https://pbs.twimg.com/profile_images/1056520922117607424/O-Gh6FSF_normal.jpg', 'profile_banner_url': 'htt
ps://pbs.twimg.com/profile_banners/62364521/1540732670', 'profile_link_color': '0084B4', 'profile_sidebar_border_color': 'FFF
FFF', 'profile_sidebar_fill_color': 'DDEEF6', 'profile_text_color': '333333', 'profile_use_background_image': True, 'has_exte
nded_profile': False, 'default_pr
e', 'notifications': None, 'translator_type': 'none', 'geo': None, 'coordinates': None, 'place': None, 'contributors': None,
```

A API do Twitter retorna diversos atributos para cada tweet capturado

É selecionado apenas os atributos de nome de usuário e o tweet completo, desta forma, é criado um data frame com os dados capturados e são salvos em um arquivo CSV.

```
In [75]: tweet_detalhes= [[tweet.user.screen_name, tweet.full_text] for tweet in tweets]
```

```
In [76]: tweet_df = pd.DataFrame(data=tweet_detalhes, columns=["user", "text"])
```

```
In [23]: tweet_df.to_csv('BaseDeDadosTweets.csv', mode='a')
```

Rotulação do Sentimento

Após os tweets serem capturados e salvos em um arquivo CSV, é feito a rotulação manual dos sentimentos expressos nos mesmos de acordo com os seguintes critérios:

Positivo	Negativo	Neutro
Esperança	Raiva	Informações gerais
Otimismo	Pessimismo	Propaganda
Felicidade	Tristeza	Curiosidades
Sarcasmos positivo	Sarcasmos negativos	Conselho ou dicas
Ironias positivas	Ironias negativas	Ações governamentais
Recuperados da covid	Mortes	*Tweets* que não expressa sentimento
Desejar o bem e empenho contra a pandemia	Indignação e críticas	

Após a extração dos tweets, é criado um novo notebook para desenvolver os processos de Machine Learning. A imagem a seguir contém todas as bibliotecas necessárias para os processos computacionais que serão realizados.

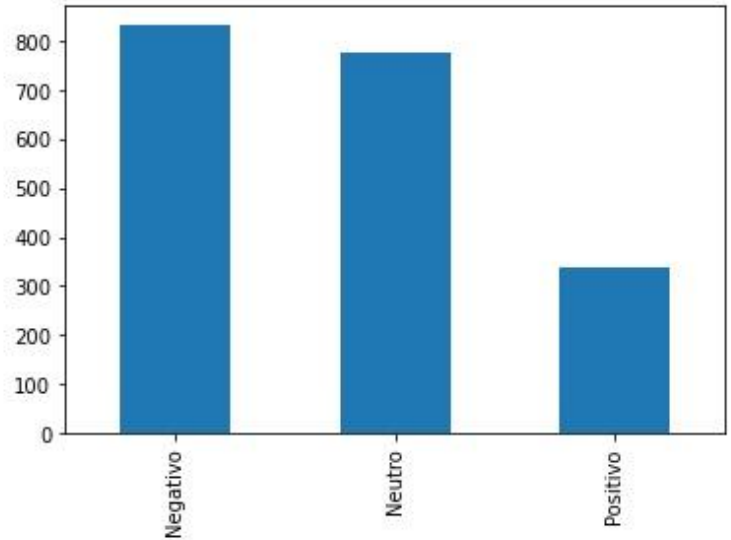
In [1]:
```
from nltk import word_tokenize
from nltk.tokenize import TweetTokenizer
import nltk
import numpy as np
import matplotlib.pyplot as plt
import re
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn import svm
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import ComplementNB
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.model_selection import cross_val_predict, train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

Após a rotulação de sentimentos de um total de 1950 tweets, a base de dados se encontra desbalanceada. Como esperado, a maioria dos tweets relacionado a pandemia da covid-19 eram negativos e neutros, com poucas classificações positivas.

Qtd. *Tweets*	1950
Positivo	340
Negativo	833
Neutro	777

```
In [19]: %matplotlib inline
         dataset.classe.value_counts().plot(kind='bar')

Out[19]: <AxesSubplot:>
```



Limpeza de Dados (Pré-Processamento)

In [13]:
```
def RemoveStopWords(instancia):
    stopwords = set(nltk.corpus.stopwords.words('portuguese'))
    palavras = [i for i in instancia.split() if not i in stopwords]
    return (" ".join(palavras))
text = [RemoveStopWords(i) for i in text]
```

In [14]:
```
def limpar_dados(text):
    text = re.sub("RT[\s]+", " ", text).lower()
    text = re.sub("https?://[A-Za-z0-9./]*", " ", text)
    text = re.sub("&amp", " ", text)
    text = re.sub("\n", " ", text)
    dataset.drop_duplicates(['text'], inplace=True)
    return text
```

Nesta etapa é feito a limpeza dos dados, descartando o que é irrelevante para a classificação e que não possui nenhum valor semântico. As funções de pré-processamento, inclui: Remoção de stopwords, converter todas as letras em minúsculo, remoção de links, remoção de caracteres não alfabéticos e remoção de linhas duplicadas na base.

Vetorização e Tokenização

Tokenização dos dados, utilizando o tokenizador de tweets da biblioteca NLTK

In [17]:
```
from nltk.tokenize import TweetTokenizer
```

In [18]:
```
tweet_tokenizer = TweetTokenizer()
```

Vetorização dos tweets

In [19]:
```
vectorizer = CountVectorizer(analyzer="word", tokenizer=tweet_tokenizer.tokenize)
```

Aplicar o vetorizador no Texto

In [20]:
```
tweets_vetorizados = vectorizer.fit_transform(text)
type(tweets_vetorizados)
```

Out[20]:
```
scipy.sparse.csr.csr_matrix
```

In [21]:
```
tweets_vetorizados.shape
```

Out[21]:
```
(1950, 9907)
```

Algoritmos classificadores implementados:

- MULTINOMIAL NAIVE BAYES
- BERNOULLI NAIVE BAYES
- COMPLEMENT NAIVE BAYES
- GAUSSIAN NAIVE BAYES
- SVM - SUPPORT VECTOR MACHINES
- KNN - VIZINHOS PRÓXIMOS

Aplicação dos algoritmos classificadores

Antes de gerar um modelo classificador pelos métodos de Machine Learning, a base é dividida em 30% para teste e 70% para treinamento.

In [140]:
```
X = vectorizer.fit_transform(dataset["text"])
y = dataset["classe"]
```

In [118]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1352, 8379)
(580, 8379)
(1352,)
(580,)
```

Multinomial Naive Bayes

MULTINOMIAL NAIVE BAYES

Modelo com o sklearn Multinomial Naive Bayes e Acurácia/Predição do modelo

Acurácia pelo método Predict (30%Teste e 70%Treinamento)

In [166]:
```
NB_model = MultinomialNB()
NB_model.fit(X_train, y_train)
y_predict_nb = NB_model.predict(X_test)
print(accuracy_score(y_test, y_predict_nb))
```

```
0.6775862068965517
```

Acurácia com validação cruzada de 10 páginas

Acurácia por Validação Cruzada de 10 partes (toda as amostras)

In [169]:
```
y_predict_nb= cross_val_predict (NB_model, X, y, cv=10)
print(metrics.accuracy_score(y, y_predict_nb))
```

```
0.6480331262939959
```

Tabela de *Classification Report* e Matriz de Confusão - **MultinomialNB**

```
In [42]: sentimento =['Positivo','Negativo','Neutro']
         print (metrics.classification_report(y,y_predict_nb,sentimento))
```

	precision	recall	f1-score	support
Positivo	0.63	0.35	0.45	334
Negativo	0.63	0.84	0.72	828
Neutro	0.68	0.57	0.62	770
accuracy			0.65	1932
macro avg	0.65	0.59	0.60	1932
weighted avg	0.65	0.65	0.63	1932

Classification Report **- Precisão, revocação e F-measure.**

```
In [145]: print (pd.crosstab(y, y_predict_nb2, rownames=['Real'], colnames=['Predito'], margins=True))
```

Predito	Negativo	Neutro	Positivo	All
Real				
Negativo	699	112	17	828
Neutro	283	437	50	770
Positivo	123	95	116	334
All	1105	644	183	1932

Matriz de Confusão - Número exato de classificações para cada classe

Bernoulli Naive Bayes

BERNOULLI NAIVE BAYES

Modelo com o sklearn Bernoulli Naive Bayes e Acurácia/Predição do modelo

Acurácia pelo método Predict (30%Teste e 70%Treinamento)

In [147]:
```
NBBE_model = BernoulliNB()
NBBE_model.fit(X_train, y_train)
y_predict_nbbe = NBBE_model.predict(X_test)
print(accuracy_score(y_test, y_predict_nbbe))
```

```
0.6689655172413793
```

Acurácia com validação cruzada de 10 páginas

Acurácia por Validação Cruzada de 10 partes (toda as amostras)

In [149]:
```
y_predict_nbbe = cross_val_predict (NBBE_model, X, y , cv=10)
print(metrics.accuracy_score(y, y_predict_nbbe))
```

```
0.6273291925465838
```

Tabela de *Classification Report* e Matriz de Confusão - **BernoulliNB**

```
In [352]: sentimento =['Positivo','Negativo','Neutro']
          print (metrics.classification_report(y,y_predict_nbbe,sentimento))
```

	precision	recall	f1-score	support
Positivo	0.77	0.10	0.18	334
Negativo	0.62	0.85	0.72	828
Neutro	0.63	0.61	0.62	770
accuracy			0.63	1932
macro avg	0.67	0.52	0.51	1932
weighted avg	0.65	0.63	0.59	1932

***Classification Report* - Precisão, revocação e F-measure.**

```
In [150]: print (pd.crosstab(y, y_predict_nbbe, rownames=['Real'], colnames=['Predito'], margins=True))
```

Predito	Negativo	Neutro	Positivo	All
Real				
Negativo	707	120	1	828
Neutro	290	471	9	770
Positivo	146	154	34	334
All	1143	745	44	1932

Matriz de Confusão - Número exato de classificações para cada classe

Complement Naive Bayes

COMPLEMENT NAIVE BAYES

Modelo com o sklearn Complement Naive Bayes e Acurácia/Predição do modelo

Acurácia pelo método Predict (30%Teste e 70%Treinamento)

In [153]:
```
CNB_model = ComplementNB()
CNB_model.fit(X_train, y_train)
y_predict_cnb = CNB_model.predict(X_test)
print(accuracy_score(y_test, y_predict_cnb))
```

```
0.6896551724137931
```

Acurácia por Validação Cruzada de 10 partes (toda as amostras)

In [154]:
```
y_predict_cnb = cross_val_predict (CNB_model, X, y , cv=10)
print(metrics.accuracy_score(y, y_predict_cnb))
```

```
0.6573498964803313
```

Tabela de *Classification Report* e Matriz de Confusão - **ComplementNB**

```
In [356]: sentimento =['Positivo','Negativo','Neutro']
          print (metrics.classification_report(y,y_predict_cnb,sentimento))
```

	precision	recall	f1-score	support
Positivo	0.50	0.60	0.54	334
Negativo	0.68	0.81	0.74	828
Neutro	0.73	0.52	0.61	770
accuracy			0.66	1932
macro avg	0.64	0.64	0.63	1932
weighted avg	0.67	0.66	0.65	1932

***Classification Report* - Precisão, revocação e F-measure.**

```
In [155]: print (pd.crosstab(y, y_predict_cnb, rownames=['Real'], colnames=['Predito'], margins=True))
```

Predito	Negativo	Neutro	Positivo	All
Real				
Negativo	670	95	63	828
Neutro	229	398	143	770
Positivo	80	52	202	334
All	979	545	408	1932

Matriz de Confusão - Número exato de classificações para cada classe

Gaussian Naive Bayes

GAUSSIAN NAIVE BAYES

Modelo com o sklearn Gaussian Naive Bayes e Acurácia/Predição do modelo

Acurácia pelo método Predict (30%Teste e 70%Treinamento)

In [156]:
```
GNB_model = GaussianNB()
GNB_model.fit(X_train.todense(), y_train)
y_predict_gnb = GNB_model.predict(X_test.todense())
print(accuracy_score(y_test, y_predict_gnb))
```

```
0.6275862068965518
```

Acurácia por Validação Cruzada de 10 partes (toda as amostras)

In [158]:
```
y_predict_gnb = cross_val_predict (GNB_model, X.todense(), y , cv=10)
print(metrics.accuracy_score(y, y_predict_gnb))
```

```
0.5952380952380952
```

Tabela de *Classification Report* e Matriz de Confusão - **GaussianNB**

```
In [360]: sentimento =['Positivo','Negativo','Neutro']
          print (metrics.classification_report(y,y_predict_gnb,sentimento))
```

	precision	recall	f1-score	support
Positivo	0.43	0.44	0.44	334
Negativo	0.62	0.73	0.67	828
Neutro	0.64	0.52	0.57	770
accuracy			0.60	1932
macro avg	0.57	0.56	0.56	1932
weighted avg	0.60	0.60	0.59	1932

***Classification Report* - Precisão, revocação e F-measure.**

```
In [159]: print (pd.crosstab(y, y_predict_gnb, rownames=['Real'], colnames=['Predito'], margins=True))
```

Predito	Negativo	Neutro	Positivo	All
Real				
Negativo	605	140	83	828
Neutro	259	397	114	770
Positivo	104	82	148	334
All	968	619	345	1932

Matriz de Confusão - Número exato de classificações para cada classe

Support Vector Machine - SVM

SUPPORT VECTOR MACHINES - SVM

Modelo com o SVM - Support Vector Machines e Acurácia/Predição do modelo

Acurácia pelo método Predict (30%Teste e 70%Treinamento)

In [80]:
```
SVM_model = svm.SVC(C=1.0)
SVM_model.fit(X_train, y_train)
y_predict_svm = SVM_model.predict(X_test)
print(accuracy_score(y_test, y_predict_svm))
```

```
0.646551724137931
```

Acurácia por Validação Cruzada de 10 partes (toda as amostras)

In [83]:
```
y_predict_svm = cross_val_predict (SVM_model, X, y, cv=10)
print(metrics.accuracy_score(y, y_predict_svm))
```

```
0.6185300207039337
```

Tabela de *Classification Report* e Matriz de Confusão - **svm.SVC**

```
In [41]: sentimento =['Positivo','Negativo','Neutro']
         print (metrics.classification_report(y,y_predict_svm,sentimento))
```

	precision	recall	f1-score	support
Positivo	0.86	0.19	0.31	334
Negativo	0.62	0.76	0.69	828
Neutro	0.59	0.65	0.62	770
accuracy			0.62	1932
macro avg	0.69	0.53	0.54	1932
weighted avg	0.65	0.62	0.59	1932

***Classification Report* - Precisão, revocação e F-measure.**

```
In [160]: print (pd.crosstab(y, y_predict_svm, rownames=['Real'], colnames=['Predito'], margins=True))
```

Predito	Negativo	Neutro	Positivo	All
Real				
Negativo	633	194	1	828
Neutro	263	498	9	770
Positivo	120	150	64	334
All	1016	842	74	1932

Matriz de Confusão - Número exato de classificações para cada classe

Vizinhos Próximos - KNN

VIZINHOS MAIS PRÓXIMOS CLASSIFICADOR

Modelo com o sklearn K-Neighbors (N-Vizinhos próximos) e Acurácia/Predição do modelo

Acurácia pelo método Predict (30%Teste e 70%Treinamento)

In [106]:
```
n_neighbors=10
kvizinhos_model = KNeighborsClassifier(n_neighbors=n_neighbors)
kvizinhos_model.fit(X_train, y_train)
y_predict_kvimodel=kvizinhos_model.predict(X_test)
print(accuracy_score(y_test, y_predict_kvimodel))
```

```
0.5344827586206896
```

Acurácia por Validação Cruzada de 10 partes (toda as amostras)

In [127]:
```
y_predict_kvimodel = cross_val_predict (kvizinhos_model, X, y , cv=10)
print(metrics.accuracy_score(y, y_predict_kvimodel))
```

```
0.532608695652174
```

Tabela de *Classification Report* e Matriz de Confusão - **KNeighborsClassifier (Neighbors=10)**

```
In [93]: sentimento =['Positivo','Negativo','Neutro']
         print (metrics.classification_report(y,y_predict_kvimodel,sentimento))
```

	precision	recall	f1-score	support
Positivo	0.59	0.19	0.28	334
Negativo	0.56	0.59	0.58	828
Neutro	0.50	0.62	0.56	770
accuracy			0.53	1932
macro avg	0.55	0.47	0.47	1932
weighted avg	0.54	0.53	0.52	1932

Classification Report **- Precisão, revocação e F-measure.**

```
In [161]: print (pd.crosstab(y, y_predict_kvimodel, rownames=['Real'], colnames=['Predito'], margins=True))
```

Predito	Negativo	Neutro	Positivo	All
Real				
Negativo	496	319	13	828
Neutro	265	481	24	770
Positivo	105	177	52	334
All	866	977	89	1932

Matriz de Confusão - Número exato de classificações para cada classe

Acurácia dos modelos

MÉTODO	AVALIAÇÃO PREDICT	VALIDAÇÃO CRUZADA
MULTINOMIAL NB	67,75%	64,80%
BERNOULLI NB	66,89%	62,73%
COMPLEMENT NB	68,96%	65,73%
GAUSSIAN NB	62,75%	59,52%
SVM	64,65%	61,85%
KNEIGHBORS	53,44%	53,26%

Conclusão

O classificador Complement Naive Bayes, indicado para bases desbalanceadas, obteve a maior acurácia de classificação do modelo em relação às outras variantes do algoritmo Naive Bayes, bem como supera também os classificadores SVM e Vizinhos Próximos. A base de tweets, contém um baixo número de classificações de 'tweets positivos', por ser um classificador indicado para dados desbalanceados, o Complement NB obteve o melhor resultado na medida F-measure para tweets positivos (0.54), contribuindo para a média de acurácia geral do modelo.

O classificador Vizinhos Próximos, gerou o pior modelo de predição, não sendo de certa maneira surpreendente, devido ao mesmo não ser o mais indicado para classificação de texto. O K Neighbor Classifier é amplamente utilizado em técnicas para reconhecimento de faces.

O Multinomial Naive Bayes, é a variação Naive Bayes mais conhecida e amplamente utilizada para classificação de textos e análise de sentimentos, nesta aplicação, obteve resultados semelhantes ao Complement Naive Bayes, se posicionando como a 2º melhor acurácia geral. Em uma base de dados propriamente balanceada, a tendência é de que alcançaria o melhor resultado geral.

Análise de sentimentos do Twitter em relação a covid-19 e a comparação de algoritmos classificadores

Rômulo Rodrigues Coutinho

CENTRO UNIVERSITÁRIO CARIOCA

CIÊNCIA DA COMPUTAÇÃO