UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
ESCOLA DE CIÊNCIAS E TECNOLOGIA

Análise de sentimentos em posts do Twitter

url: https://github.com/alyssonolima/Projeto-Virtual-Mood-Identifier

Discente: Alysson Rafael Oliveira de Lima
Docente: Orivaldo Santana Jr

Natal, 29 de novembro de 2019

Introdução

As redes sociais são utilizadas por milhares de pessoas ao redor do mundo para diversos fins. Dentre vários tipos e funções existentes a rede social **twitter** se destaca por ser uma ferramenta de compartilhamento de mensagens curtas, inicialmente 140 caracteres, o que lhe dá o nome de microblog.

Criado em Março de 2006, por Jack Dorsey, Evan Williams, Biz Stone e Noah Glass, o twitter conta atualmente com mais de 284 milhões de usuários registrados.

Devido ao grande alcance e a liberdade que os usuários têm nas postagens, é possível extrair bastante informação pessoal circulando nas redes. Diante disto o presente projeto busca desenvolver uma ferramenta com o uso de machine learn para identificar sentimentos nas postagens e classificá-las em três grupos: Positivas, Negativas e Neutras.

Para identificar os sentimentos utilizaremos o processamento de linguagem natural (PNL) que trata do desenvolvimento de aplicativos e serviços capazes de entender idiomas humanos.

Após os dados serem categorizados sentimentalmente, serão aplicados a um modelo de rede neural SOM, para ser extraído as características e ser possível algum agrupamento automático.

Desenvolvimento

Foi escolhida a linguagem **Python,** para desenvolvimento do projeto, por se tratar de uma linguagem bastante utilizada, e que possui diversas bibliotecas que auxiliam na manipulação de base de dados.

Todo o código foi desenvolvido e armazenado no Google Colab ou "Colaboratório" que é um serviço de nuvem gratuito hospedado pelo Google que permite escrever e executar Python no navegador, sem necessitar configuração.

Para categorizar e realizar um tratamento nos textos foi utilizado o modelo Processamento de Linguagem Natural NLP.

Posteriormente foi utilizado a rede SOM. A rede SOM é uma rede neural de 2 camadas que aceita padrões de N-dimensões como entrada e os mapeia para um conjunto de neurônios de saída, o qual representa o espaço dos dados a serem agrupados. O mapa (camada) de saída, que é tipicamente bi-dimensional, representa as posições dos neurônios em relação aos seus vizinhos.

O desenvolvimento foi dividido em três principais partes: Mineração dos dados, manipulação através da aplicação do processamento de linguagem Natural e por fim a aplicação da rede SOM para obter resultados identificáveis.

Mineração de Dados

Os dados, da rede social, que foram utilizados no projeto estão disponíveis de forma pública e visível a qualquer usuário da rede, conforme indica a política de privacidade da empresa.

O twitter fornece uma plataforma, para desenvolvedores, onde é disponibilizado uma API de serviços e informações. Através dessa API é possível obter, por exemplo, todos os tweets públicos de um perfil qualquer na rede.

Para facilitar a configuração e uso da API, foi utilizado a biblioteca para python **Tweepy,** que oferece diversas funções que simplificam bastante a interação com as informações disponibilizadas na plataforma da rede social.

O código abaixo, por exemplo, em poucas linhas realiza a autenticação e consegue fazer o download dos tweets da linha do tempo da conta conectada e imprime cada um de seus textos no console.

```
import tweepy

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)

public_tweets = api.home_timeline()
for tweet in public_tweets:
    print(tweet.text)
```

Algorítmo: Exemplo de uso da biblioteca tweepy

Com o uso do tweepy foi realizado o download das postagens dos usuários desejados e essas informações foram salvas em um arquivo no formato .csv.

```
def obter_tweets(usuario, limite=500):
      resultados = api.user_timeline(screen_name=usuario, count=limite,
tweet_mode='extended')
      tweets = []
      for r in resultados:
            tweet = re.sub(r'http\S+', '', r.full_text)
            tweets.append(tweet.replace('\n', ' '))
return tweets
```

Algorítmo: Método que faz a busca dos tweets.

		Tweets
2	0	95% Approval Rating in the Republican Party. Thank you!
3	1	New Stock Market Record today, AGAIN. Congratulations USA!
4	2	People now realize it is a Democrat Hoax!
5	3	RT @piersmorgan: 🤣🤣
6	4	RT @EricTrump: What an amazing night in Florida! #USAUSAUSA
7	6	RT @GOPChairwoman: Overwhelmingly strong support for @realDonaldTrump in a DEM county in battleground Florida. His supporters are already...
8	7	GOD BLESS THE U.S.A.! #MAGA
9	9	THANK YOU FLORIDA! #KAG2020
10	10	"OMB Official: Ukraine Aid Held Up Because Other Countries Weren't Giving"
11	12	RT @WhiteHouse: LIVE: President @realDonaldTrump Pardons the National Thanksgiving Turkey
12	13	...love to have Mike Pompeo, Rick Perry, Mick Mulvaney and many others testify about the phony Impeachment Hoax. It is a Democrat Scam that is going nowhere but, fu
13	14	...lawyer has already stated that I did nothing wrong. John Bolton is a patriot and may know that I held back the money from Ukraine because it is considered a corrupt c
14	15	The D.C. Wolves and Fake News Media are reading far too much into people being forced by Courts to testify before Congress. I am fighting for future Presidents and th
15	16	RT @DailyCaller: President @realDonaldTrump signs the Women's Suffrage Centennial Commemorative Coin Act which will direct the U.S. Treasur...

Figura: tweets salvos em formato csv.

Aplicação do modelo de Linguagem Natural

Após estar de posse dos dados é necessário realizar algumas melhorias nos tweets e aplicá-los ao modelo NLP, e assim, extrair as características dos dados nos textos.

Fez-se a escolha por realizar a análise da mensagens em inglês devido a um melhor aproveitamento das ferramentas existentes.

Utilizamos a **NLTK** (Natural Language Toolkit) que é a biblioteca mais popular para processamento de linguagem natural, escrita em Python e que conta uma grande quantidade de informações e exemplos disponíveis.

A classificação do modelo NLP foi escolhido o **Naive Bayes classifier**, que classifica os dados através de um modelo probabilístico de aprendizado, baseado no teorema de Bayes.

Para auxiliar o modelo utilizamos o **Vader Lexicon**, que é uma ferramenta de análise de sentimentos baseada em regras e léxico e que está sintonizada especificamente com os sentimentos expressos nas mídias sociais. É totalmente de código aberto. Através dessa ferramenta se obtém como resultado do processamento uma composição como a segue no exemplo:

```
“VADER is smart, handsome, and funny.”
{'pos': 0.746, 'compound': 0.8316, 'neu': 0.254, 'neg': 0.0}
```

```
“VADER is smart, handsome, and funny!”
{'pos': 0.752, 'compound': 0.8439, 'neu': 0.248, 'neg': 0.0}
“VADER is VERY SMART, handsome, and FUNNY.”
{'pos': 0.754, 'compound': 0.9227, 'neu': 0.246, 'neg': 0.0}
```

onde:

A pontuação **compound** é calculada somando as pontuações de valência de cada palavra no léxico, ajustadas de acordo com as regras e, em seguida, normalizadas para estar entre -1 (máximo extremo negativo) e +1 (máximo extremo positivo).

As pontuações **pos**, **neu** e **neg** são razões para proporções de texto que se enquadram em cada categoria (assim que estes devem todos adicionar até ser 1 ou próximo a ela com a operação float).

Após o treinamento foi obtido a seguinte acurácia:

```
Training classifier
Evaluating NaiveBayesClassifier results...
Accuracy: 0.8
```

Com essas ferramentas é possível configurar esse código:

```
sentim_analyzer = SentimentAnalyzer()
all_words_neg = sentim_analyzer.all_words([mark_negation(doc) for doc in
training_docs])

unigram_feats = sentim_analyzer.unigram_word_feats(all_words_neg, min_freq=4)
len(unigram_feats)

sentim_analyzer.add_feat_extractor(extract_unigram_feats,
unigrams=unigram_feats)

training_set = sentim_analyzer.apply_features(training_docs)
test_set = sentim_analyzer.apply_features(testing_docs)

trainer = NaiveBayesClassifier.train
classifier = sentim_analyzer.train(trainer, training_set)

for key,value in sorted(sentim_analyzer.evaluate(test_set).items()):
  print('{0}: {1}'.format(key, value))
```

Aplicado ao modelo essas configurações citadas e exibidas nesse trecho de código acima, é aplicado à tabela de tweets brutos, extraídos do twitter, são tratados e geram o seguinte resultado, que são salvos em arquivo no formato csv.

Após esse procedimento temos disponível uma tabela com as seguintes informações:

Unnamed: 0	tweets	compound	negativos	neutro	positivo
0	Before arguing with friends or family around t...	0.6341	0.115	0.671	0.214
1	Politicians shouldn't be picking their voters....	0.3182	0.000	0.944	0.056
2	Take a look at this piece to understand the ve...	0.2500	0.000	0.950	0.050
3	Young leaders like these are powering the prom...	0.8553	0.000	0.722	0.278
4	Proud to see Oluwaseun @AyodejiOsowobi in this...	0.1779	0.087	0.789	0.124
5	Stories like this are a reminder of our duty, ...	0.3818	0.079	0.754	0.167
6	This Veterans Day, here's a moving portrait of...	0.7579	0.000	0.866	0.134
7	No one says it better than @MichelleObama — th...	0.1779	0.069	0.841	0.090
8	Proud of all the Americans who showed up to vo...	0.8271	0.145	0.565	0.290
9	Proud to endorse an outstanding group of Virgi...	0.9684	0.000	0.603	0.397

Aplicação do modelo SOM

Com as informações geradas no passo anterior, onde os tweets foram categorizados de acordo com os sentimentos extraídos das palavras, decidimos utilizar a rede neural auto-organizada - SOM, uma vez que não temos as saídas desejadas, para poder explorar as semelhanças entre os posts e agrupar em padrões parecidos.

A rede SOM foi configurada da seguinte forma:

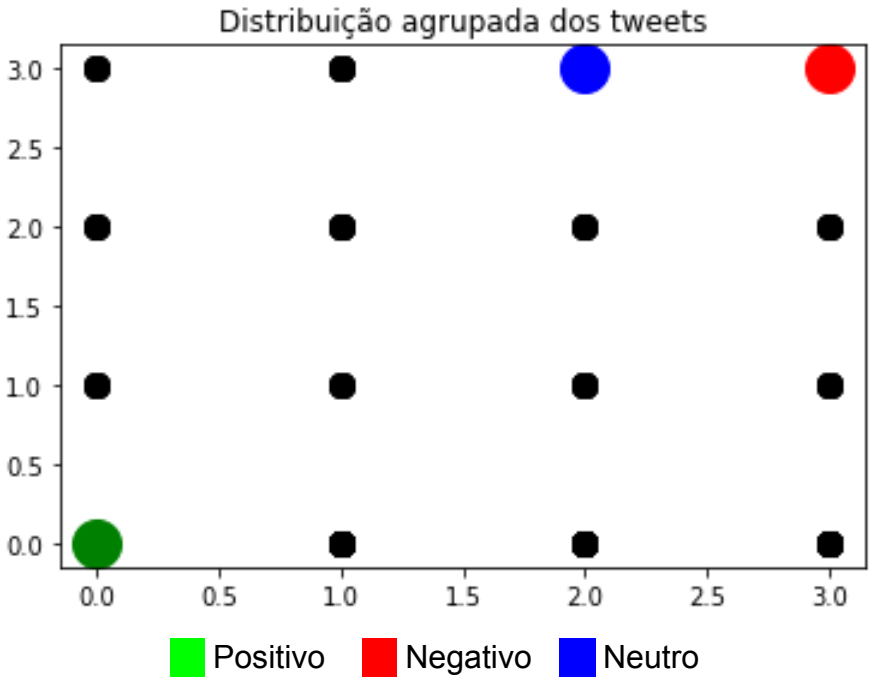
```
#tamanho da rede
tx = 4
ty = 4

# Training the SOM
from minisom import MiniSom
som = MiniSom(x = tx, y = ty, input_len = 4, sigma = 1.0, learning_rate = 0.5)
som.random_weights_init(X)
som.train_random(data = X, num_iteration = 1000)
```

Algoritmo: configuração da rede SOM.

Resultados

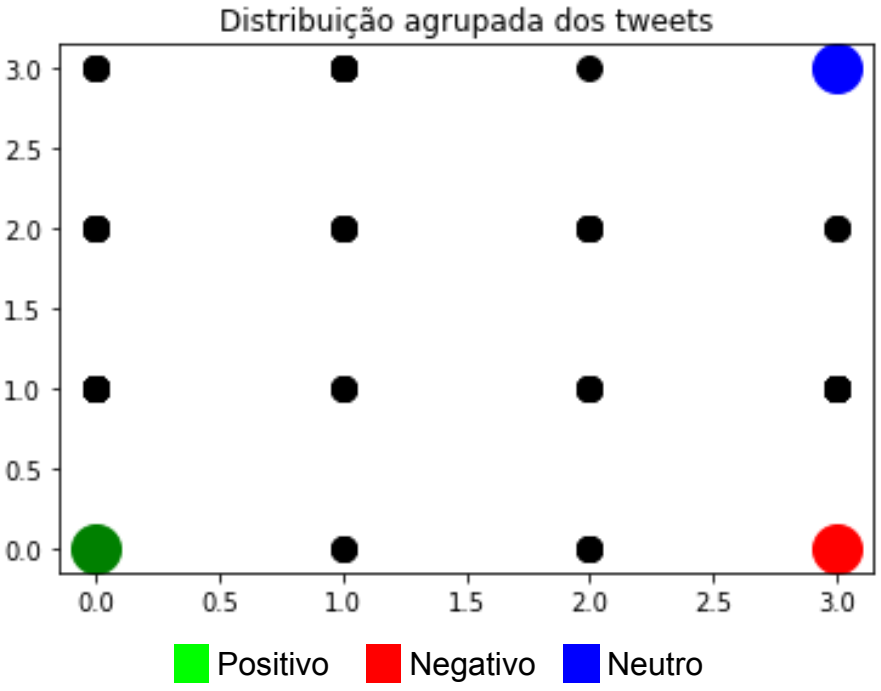
Para analisar o processo foi utilizado os perfis de dois presidentes dos Estados Unidos da América, Donald Trump e Barack Obama, segue o resultado para os dois.

O resultado da análise dos tweets de Barack Obama foi:



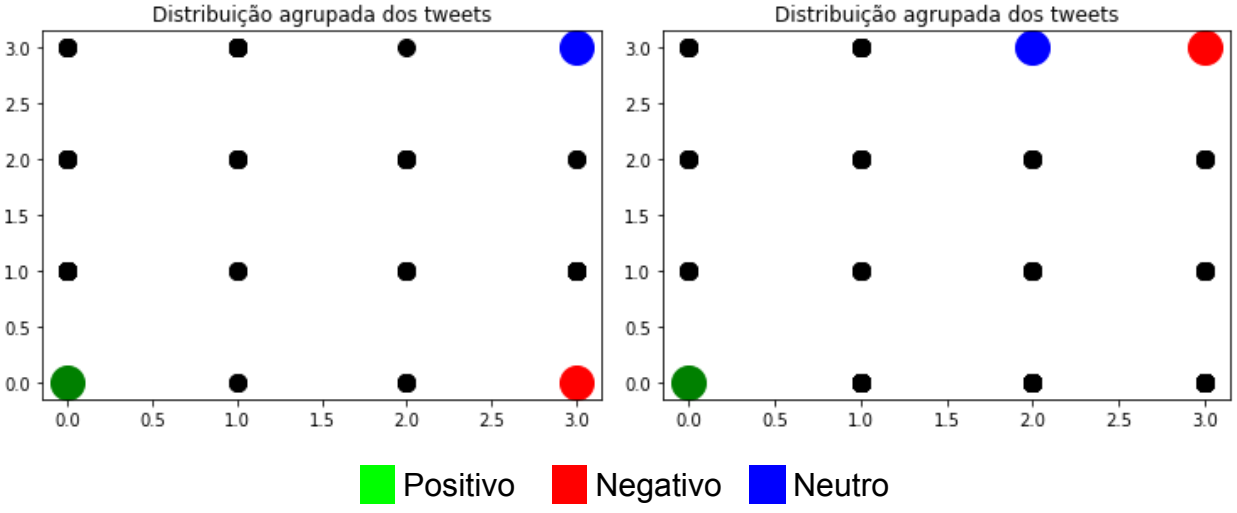

```
Total Barack Obama:
[[27.  7.  6.  6.]
 [25.  7. 10. 11.]
 [22. 10.  7. 16.]
 [19.  9.  7. 10.]]
```

Resultados para Donald Trump:




```
Total Donald Trump:
[[17. 19. 10.  6.]
 [ 3.  4.  7. 13.]
 [ 5.  7.  8.  1.]
 [19. 20.  3. 44.]]
```

Trump X Obama



Conclusão

Através das combinação de Programação de Linguagem Natural com outro modelo de rede neural é possível obter bons resultados no reconhecimento de padrões em textos. Padrões esses que podem ser ajustados para fins mais específico, como reconhecimento de tendências de posts de ódios ou até mesmo suicidas. É uma área de conhecimento bastante promissora.

Referências

Api twitter developer: Acessado em 22/11/2019. <https://developer.twitter.com/en/docs/tweets/curate-a-collection/api-reference/get-collections-show>

Obtendo tweets de usuário com Python e tweepy. Acessado em 25/11/2019. <https://gist.github.com/marcoscastro/bc43e1741b4af47fda0ef289093aae01>

MatPlotLib. Acessado em 27/11/2019. <https://matplotlib.org/gallery/index.html>

Redes Neurais Auto-Organizáveis - SOM. Acessado em 28/11/2019. <http://aimotion.blogspot.com/2009/04/redes-neurais-auto-organizaveis-som.html>

Natural Language Toolkit. Acessado em 22/11/2019. <https://www.nltk.org/>