Universidade Estadual de Montes Claros - Unimontes
Centro de Ciências Exatas e Tecnológicas - CCET
Engenharia de Sistemas

OBSERVATÓRIO DO TWITTER PARA ANÁLISE DE SENTIMENTOS DOS TWEETS BRASILEIROS

CRISTIANO FERREIRA ANTUNES
WALKER FELIPE RODRIGUES DE JESUS
Engenharia de Sistemas
9º Período

Montes Claros, 2018

Sumário

1 INTRODUÇÃO **3**

2 DESENVOLVIMENTO **3**
2.1 Base de dados 3
2.2 TextBlob 4
2.3 NLTK Natural Language Toolkit 6

3 ANALISE DE RESULTADOS **7**
3.1 Exemplos de classificações 7
3.2 Demonstração de uso 8
3.3 Explicando os gráficos 10
3.4 Discutindo resultados e comparando ferramentas utilizadas 10

4 CONCLUSÕES **20**

5 CÓDIGO **20**
5.1 Script de busca (buscatweets.py) 20
5.2 Script de análise (classification.py) 22
5.3 Script calcula médias e salva gráficos (calcmedias.py) 26

6 REFERÊNCIAS **30**

1 INTRODUÇÃO

O objetivo do trabalho é obter e analisar o sentimento de textos publicados no Twitter por brasileiros. Para realizar o trabalho utilizamos a api do Twitter *tweepy*, que requer uma conta *developer*(conta de desenvolvedor). Após buscar vários tweets com palavras e combinação de palavras específicas, utilizamos duas ferramentas para processamento de linguagem natural, realizando a analise de sentimento de cada tweet com objetivo de detectar qual sentimento que cada tema pesquisado gera em quem fez o tweet.

2 DESENVOLVIMENTO

2.1 Base de dados

Para realização do trabalho, escrevemos 3 scripts em python, sendo que o primeiro utiliza a api *tweepy* para realizar as buscas dos tweets e salvar em um banco de dados local, o segundo analisa o sentimento de cada texto dos tweets e salva o valor no mesmo banco de dados, o terceiro calcula a media de cada métrica dos classificadores.
Foi proposto fazer a busca de no mínimo 1500 tweets com combinações das palavras:

- facismo, ditadura, Brasil, medo.
- eleições, bolsonaro, voto, desenvolvimento.
- haddad, ladrão, socialismo, lula.
- fake news, governo, povo, whatsapp, espalhar.
- general, manoela, stf, militar, mourão.

Realizamos 22 combinações:

```
(ditadura) (facismo) (Brasil,medo)  (eleições,medo) (facismo,brasil)
(ditadura,medo) (medo,bolsonaro) (eleições,bolsonaro)
(voto,eleições) (voto,bolsonaro)
(bolsonaro,desenvolvimento)(haddad,ladrão)(lula,ladrão)
(haddad,socialismo)(fake news)(whatsapp,espalhar)
(espalhar,fake news)(governo,povo)(mourão,militar)
(manuela d\'ávila)(stf)(mourão)
```

A API de busca permite especificar uma data para busca dos tweets, ao informar a data, a api tenta buscar tweets feitos anterior a data informada, porém há um limite de tempo, na documentação informa que o limite é de uma semana. Não foi possível buscar tweets próximos a data de eleição conforme desejado. Se não for especificada a data, o api busca os tweets mais recentes, o que pode resultar em conclusões totalmente diferentes dependendo dos acontecimentos recentes ou mesmo do horário em que os tweets foram buscados.

Após buscar os tweets teremos todos salvos em um banco dedados, ficam armazenados da seguinte forma :

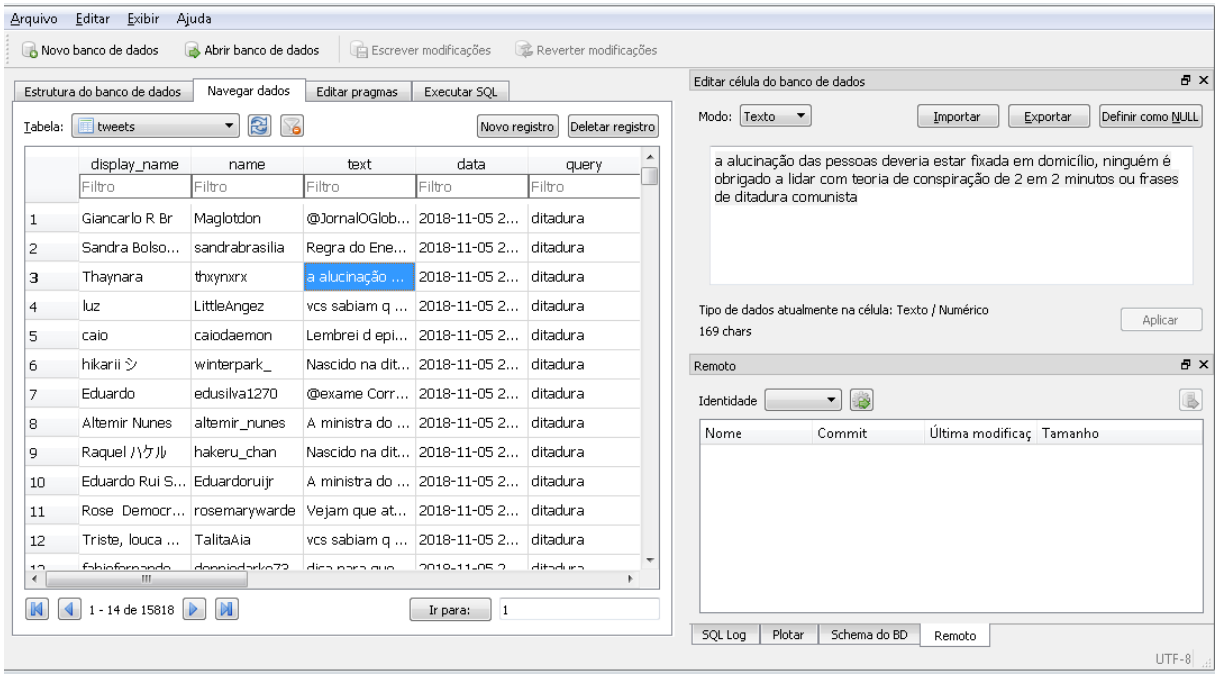


Figura 2.1: Tabela de banco de dados

A tabela armazena o *display Name* do usuário (nome exibido para outros usuarios), nome do twitter (@fulano), data da postagem, texto da postagem, combinação de palavras usadas para encontrar o tweet e id de identificação do tweet que é chave primaria da tabela, para impedir que em futuras buscas por palavras repetidas o memso tweet seja inserido.

Ferramenta usada para visualizar o banco de dados fora do programa: DB Browser for SQLite.

2.2 TextBlob

Um dos classificadores é o TextBlob, ele fornece uma API simples para tarefas comuns de processamento de linguagem natural (NLP), como marcação de parte da fala, extração de frase de substantivo, classificação, tradução e análise de sentimento que é o que nos interessa. O recurso de análise de sentimentos, de forma nativa opera apenas com a língua inglesa, porém é possível treinar a ferramente para analisar textos em português ou traduzir para o inglês. O trenamento feito tem uma base de dados disponível em ReLi (REsenha de LIvros) que possui 1600 resenhas de livros, contendo a opinião sobre o livro resenhado e sua polaridade. A analise feita com o texto em português é treinado com classificador Bayesiano disponível em *textblob.classifiers*. O trenamento funciona bem, porém retorna apenas 'pos' para positivo e 'neg' para negativo.

Fazendo a analise de sentimento do TextBlob temos 2 indicadores: *subjectivity* (subjetividade),*polarity* (polaridade). POLARITY - é um valor contínuo que varia de -1.0 a 1.0, sendo -1.0 referente a 100% negativo e 1.0 a 100% positivo. SUBJECTIVITY - que também é um valor contínuo que varia de 0.0 a 1.0, sendo 0.0 referente a 100% objetivo e 1.0 a 100% subjetivo. O problema em utilizar a tradução e depois aplicar a analise de sentimentos está no fato de o TextBlob utilizar api do Google Translate para traduzir o texto e o Google bloqueia o tradutor apos muitos requests, pois para grandes quantidades de requests o Google cobra pelo serviço, utilizamos as duas analises a que precisa de trenamento e a nativa do TextBlob e armazenamos os resultados no banco de dados. Como a outra ferramenta NLTK também opera com textos na língua inglesa, foi inserida mais uma coluna no banco de dados que armazena o tweet traduzido para o inglês.

Teremos 4 novas colunas armazenadas no banco de dados. Um exemplo analisando dados obtidos com a tradução e o TextBlob:

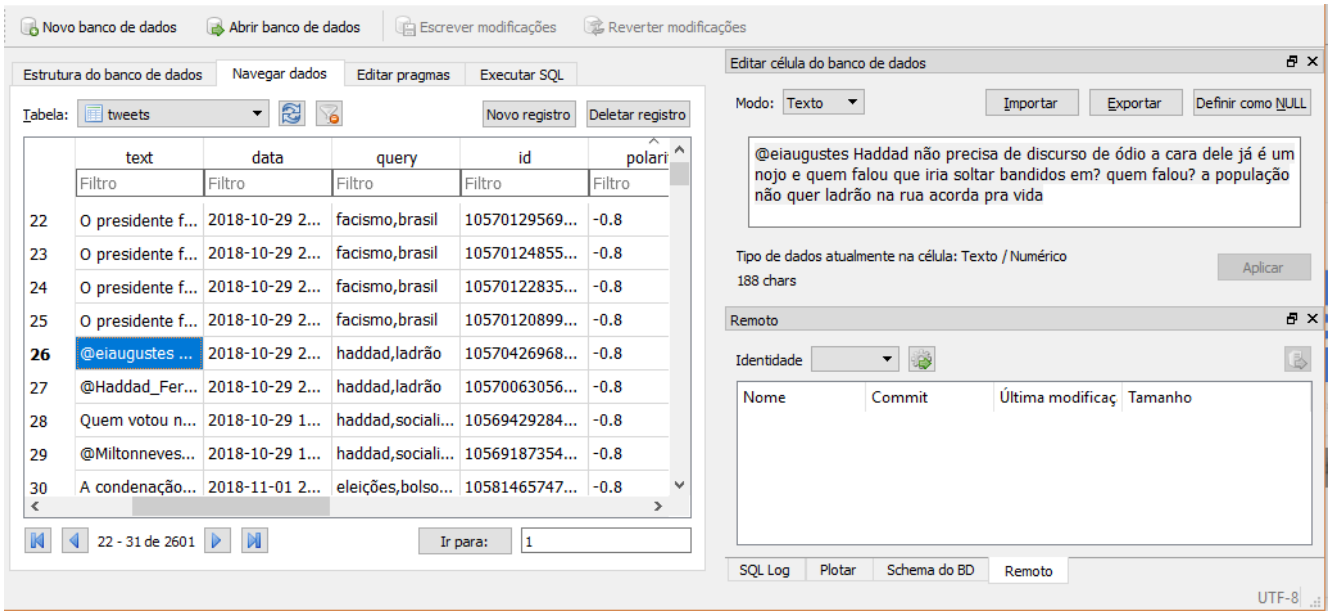


Figura 2.2: Exemplo de resultado, Texto original

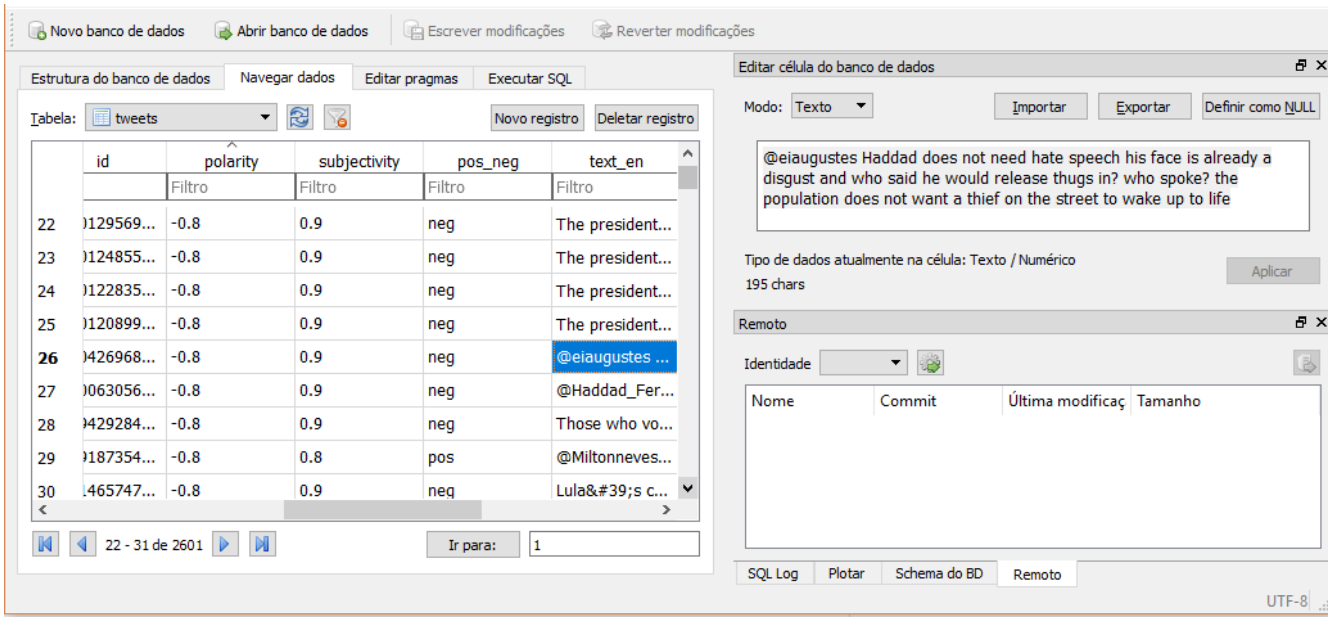


Figura 2.3: Exemplo de resultado, sentimento e texto traduzido

Podemos verificar que o classificador treinado classificou como negativo o sentimento expressado no tweet, o texto traduzido manteve o mesmo sentido e a analise da polaridade também mostrou um valor negativo. Traduzindo o texto de volta, também temos o mesmo sentido, bem semelhante ao texto original.

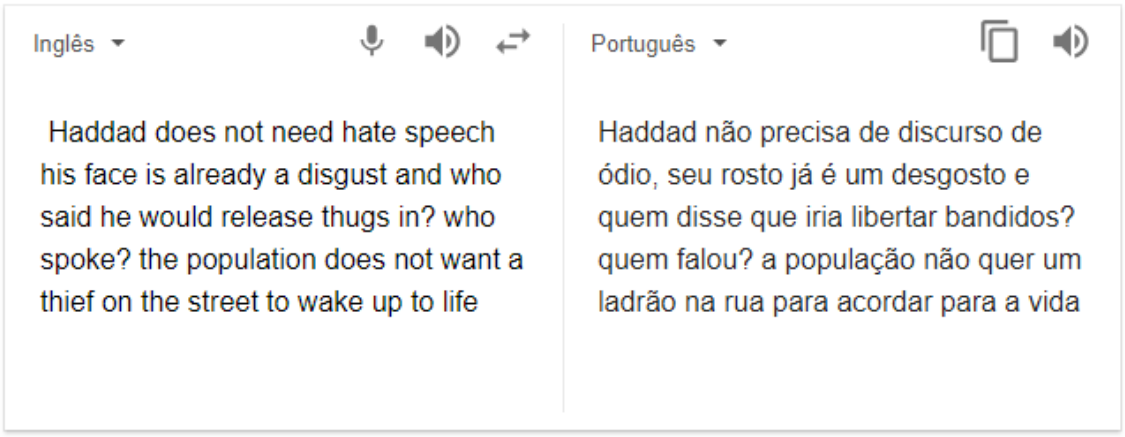


Figura 2.4: Teste de tradução

2.3 NLTK Natural Language Toolkit

NLTK, é um conjunto de bibliotecas e programas para processamento de linguagem natural simbólica e estatística para o inglês. O NLTK vem com um módulo interno de análise de sentimentos - nltk.sentiment.vader - que pode analisar um trecho de texto e classificar as sentenças em polaridade positiva, negativa e neutra de sentimentos.
O VADER produz quatro métricas de sentimento a partir dessas classificações de palavras. Os três primeiros, positivo, neutro e negativo, representam a proporção do texto que se enquadra nessas categorias. Por exemplo em uma sentença foi classificada como 45% positiva, 55% neutra e 0% negativa. A métrica final, a pontuação composta, é a soma de todas as classificações de léxico que foram padronizadas para variar entre -1 e 1.
Como já possuímos todos tweets traduzidos para o inglês, acrescentamos mais 4 colunas e analisamos todos tweets novamente com o NLTK. No final teremos a base de dados com os novos campos conforme mostra a imagem:

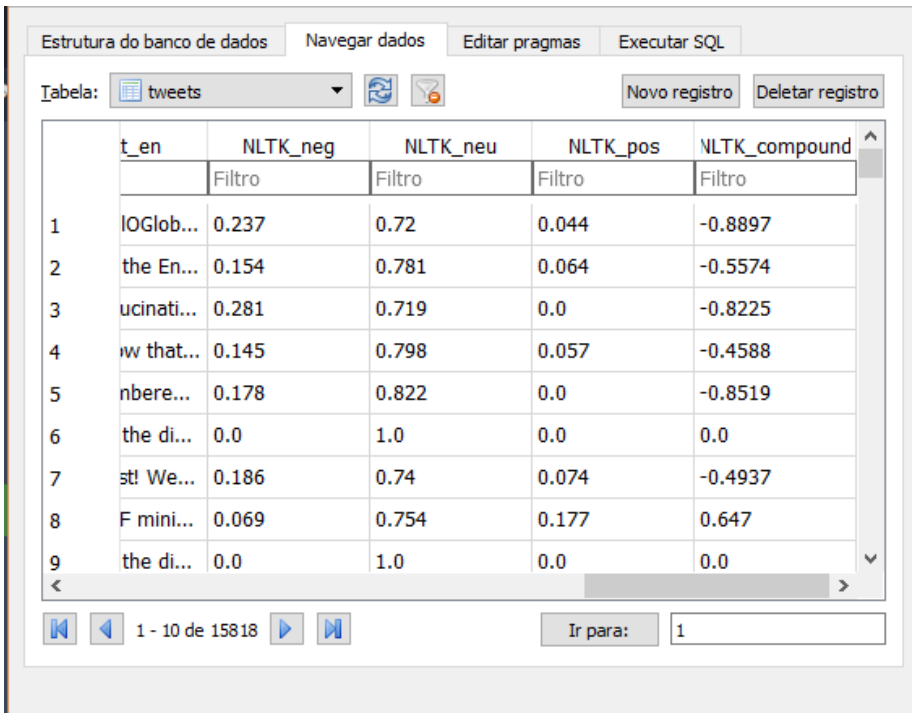


Figura 2.5: Colunas NLTK

3 ANALISE DE RESULTADOS

3.1 Exemplos de classificações

Foram salvos 15.818 tweets em uma base unica, com as combinações citadas em 2.1, algumas outras combinações pensadas depois foram salvas em outras bases totalizando mais de **23 mil** tweets, desenvolvemos o terceiro script em Python que calcula a média para cada combinação buscada, em cada métrica das duas ferramentas NLTK e TextBlob. São gerados gráficos para melhor visualização.
Para a saida com a base de dados treinada temos :

```
>>> blob=TextBlob(rows[5466][2],classifier=cl)
>>> blob
TextBlob("O STF vai ou não vai julgar Bolsonaro nos processos que
   ele é réu? Um réu pode tomar posse? A Constituição permite isso?
    Atenção @MPF_PGR Será que estão com medo? Vamos julgar e
   absolver ou condenar. A lei é para todos.")
>>> blob.classify()
'neg'
>>>
```

Para cada termo buscado, será exibido a % de 'neg' e de 'pos'. Traduzindo e analisando em inglês teremos as seguintes saidas:

```
>>> blob=blob.translate(to="en")
>>> blob
TextBlob("The STF will or will not judge Bolsonaro in the lawsuits
   he is guilty of? Can a defendant take over? Does the
   Constitution allow this? Attention @MPF_PGR Are they afraid? Let
    us judge and acquit or condemn. The law is for everyone.")
>>> blob.sentiment
Sentiment(polarity=-0.55, subjectivity=0.95)
>>>
```

POLARITY - é um valor contínuo que varia de -1.0 a 1.0, sendo -1.0 referente a 100% negativo e 1.0 a 100% positivo, realizamos a média de todos os negativos e todos os positivos e foi feito o plot. SUBJECTIVITY - que também é um valor contínuo que varia de 0.0 a 1.0, sendo 0.0 referente a 100% objetivo e 1.0 a 100% subjetivo, também foi feita a média e plotado.
Para o NLTK temos as seguintes saidas:

```
>>>
>>> text_en
'The guys are trying to change our past. Worse than that there are
   people believing that there was no Dictatorship, and that
   torture, death and exile applied to (and only) thugs. It is the
   historical cyclicality asking for passage for humanity to evolve
   . Grow for love! Do not wait for the pain'
>>> sid.polarity_scores(text_en)
{'neg': 0.25, 'neu': 0.688, 'pos': 0.063, 'compound': -0.9098}
>>>
```

Temos quatro métricas: positivo, neutro e negativo e métrica compound que a pontuação composta, é a soma de todas as classificações, retorna um valor entre -1 e 1, sendo -1 muito negativo e 1 muito positivo. É a métrica mais importante a ser analisada do NLTK. Para representação no gráfico, compound será multiplicado por -1 e plotado em vermelho caso seja negativo.

3.2 Demonstração de uso

Aqui será demonstrado como é o uso do *script* em versão final.
No arquivo *buscatweets.py* é necessário informar a base de dados, pode ser uma que não existe (será criada no mesmo diretório) ou alguma que já exista.

```
...
conn = sqlite3.connect('tweets_1.db')
...
```

Ainda no arquivo Nesse outro trecho definimos quais termos serão buscados e a quantidade máxima para cada termo.

```
...
query=['stan lee','marvel']
max_tweets = 10
...
```

Apos alterar essas linhas, podemos executar o script.

```
C:\Users\user\Desktop>python buscatweets.py
Buscando por stan lee -- 1 de 2
Buscando por marvel -- 2 de 2

C:\Users\user\Desktop>
```

Depois executamos o script classification.py

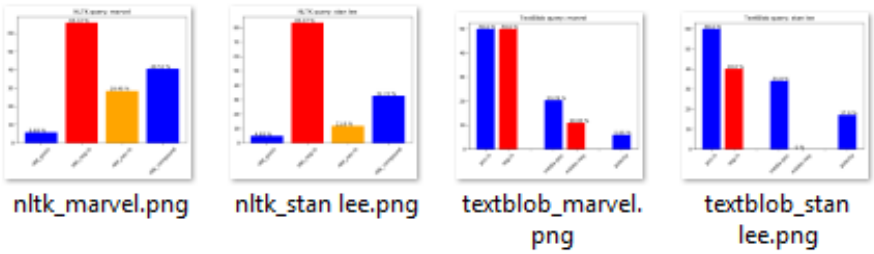
```
C:\Users\user\Desktop>python classification.py
C:\Users\user\AppData\Local\Programs\Python\Python36\lib\site-
   packages\nltk\twitter\__init__.py:20: UserWarning: The twython
   library has not been installed. Some functionality from the
   twitter package will not be available.
  warnings.warn("The twython library has not been installed. "
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
Processando Polarity 'pos' ou 'neg':
0 de 20
1 de 20
2 de 20
.
```

```
.
.
.

18 de 20
19 de 20
Processando subjectivity e polarity:
0 de 20
1 de 20
2 de 20
.
.
.
18 de 20
19 de 20
Processing NLTK:
0 de 20
1 de 20
2 de 20
.
.
.

17 de 20
18 de 20
19 de 20

C:\Users\User>
```

Com essas execuções já possuímos a tabela de banco de dados preenchida com todos os valores referente ao TextBlob e NLTK. Para concluir executamos o ultimo script para calcular as médias e salvar os gráficos em formato png.

```
C:\Users\user>python calcmedias.py

C:\Users\user>
```

Após essa execução será criada a pasta *images* com os gráficos.



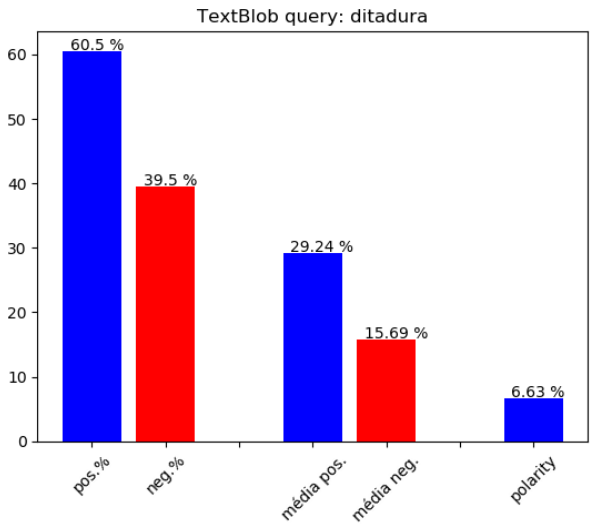
3.3 Explicando os gráficos

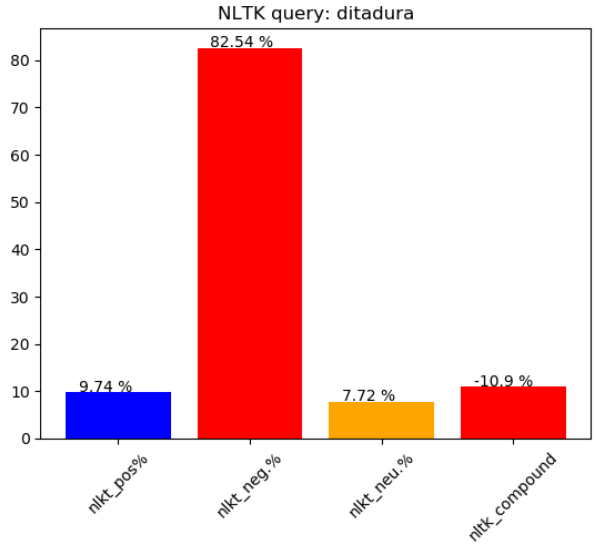
No primeiro gráfico temos a representação do TextBlob com 5 indicadores (combinação da analise em português e em inglês) , conforme mostrado nos exemplos anteriormente, treinamos o textblob para analisar textos em português que retornam resultado da analise 'pos' ou 'neg', as **duas primeiras** barras são referente a porcentagem de classificação para 'pos' e 'neg', a soma dos 2 primeiros é 100%. O **terceiro** e o **quarto** são a polaridade que retorna valores entre -1 e 1, fizemos a média entre os positivos e entre os negativos, a soma dos dois não é 100%. O **quinto** é a subjetividade que varia de 0.0 a 1.0, sendo 0.0 referente a 100% objetivo e 1.0 a 100% subjetivo, também foi feita a média.

O **segundo gráfico é o do NLTK** que é mais fácil de ser interpretado. para todas métricas foi calculado a média. Aqui é importante citar novamente que o ultimo valor nltk_compound é o resultado da combinação dos outros 3 (pos, neg e neu). Para valores negativos, a barra do compound é plotada em vermelho, quer dizer sentimentos negativos, para sentimentos positivos o valor do compound será positivo e a barra será plotada em azul.

3.4 Discutindo resultados e comparando ferramentas utilizadas

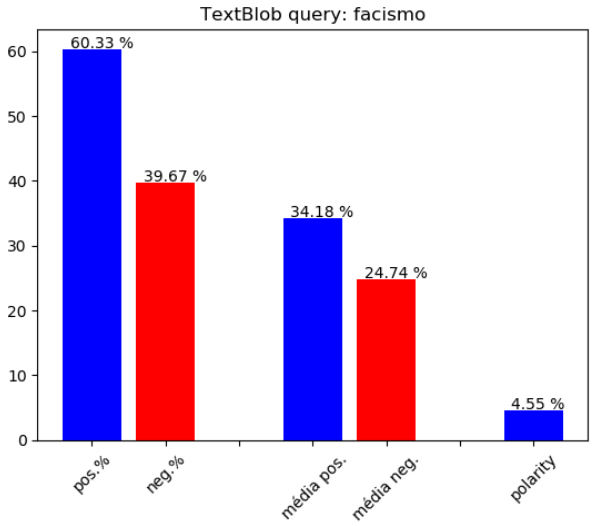
Query: "ditadura"

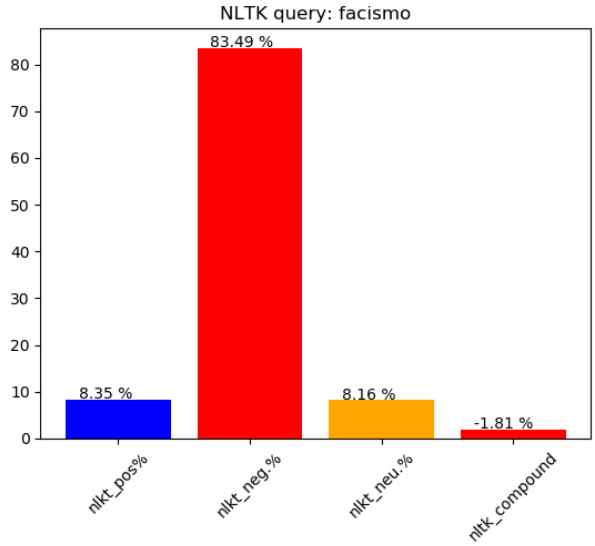




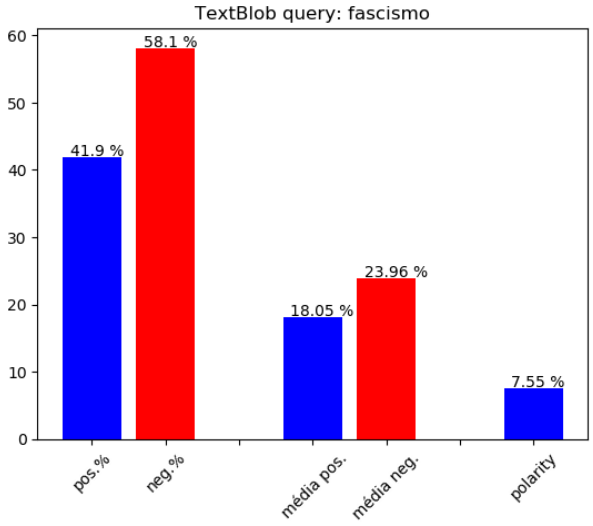
Na base de dados possui 2000 tweets encontrados com a busca "ditaduta" no dia 05-11-2018 . Analisando os resultados das duas ferramentas, podemos verificar que o resultado foi divergente, o TextBlob aponta que os tweets tem sentimento positivo, já o NLTK aponta negativo. O NLTK aparentemente mostra resultados que retratam melhor a realidade.

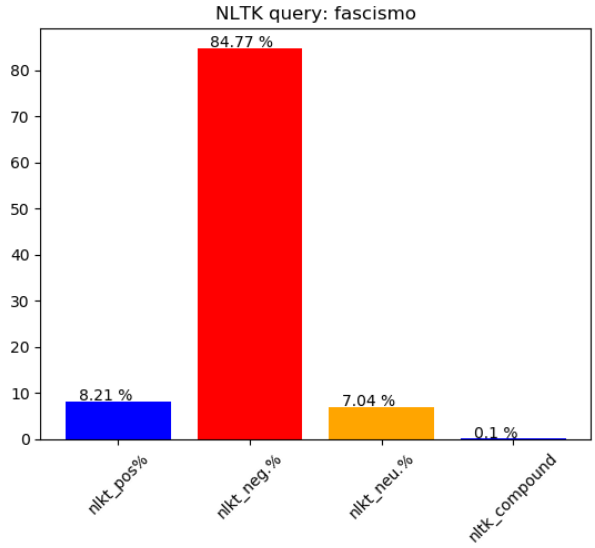
Query: "facismo"e "fascismo"





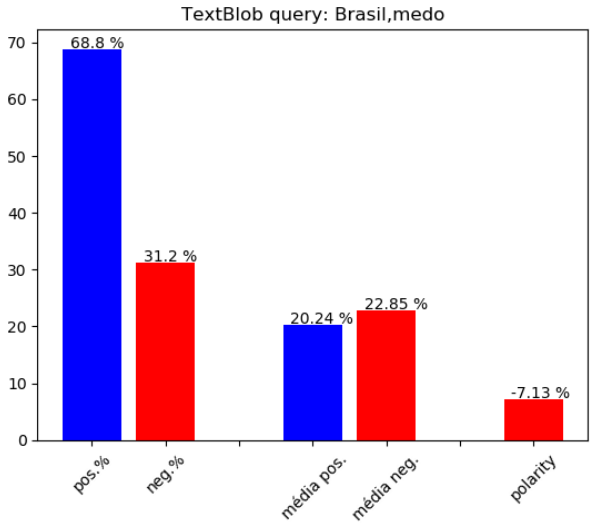
Novamente o NLTK identificou mais negativo e o TextBlob positivo. Importante também observar aqui que a palavra "facismo" está errada, o correto seria fascismo e mesmo assim ainda encontramos muitos tweets (cerca de 489). Mesmo após ter concluído a busca de todas as combinações anteriores, realizamos mais uma busca com a palavra escrita da forma correta. Lembrando que o resultado final pode flutuar bastante dependendo dos acontecimentos recentes.

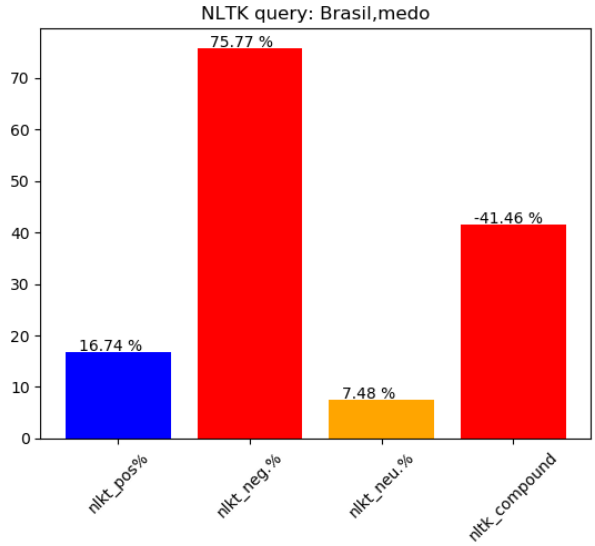




NLTK continua com classificador tendendo mais para o negativo, e o TextBlob mudou para negativo. Foram adquiridos 2000 tweets para essa busca.

Query: "Brasil,medo"

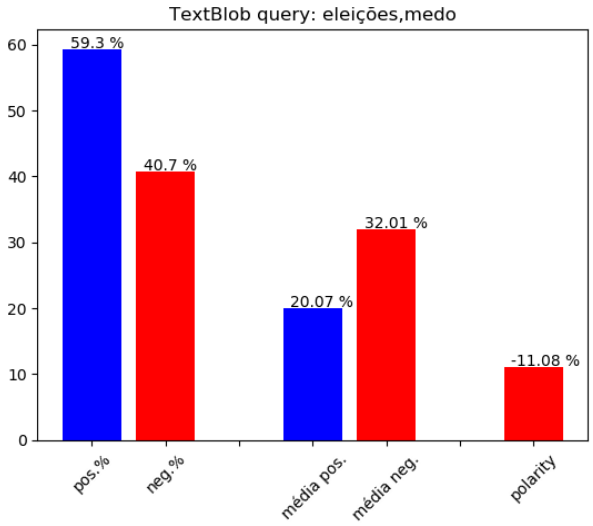


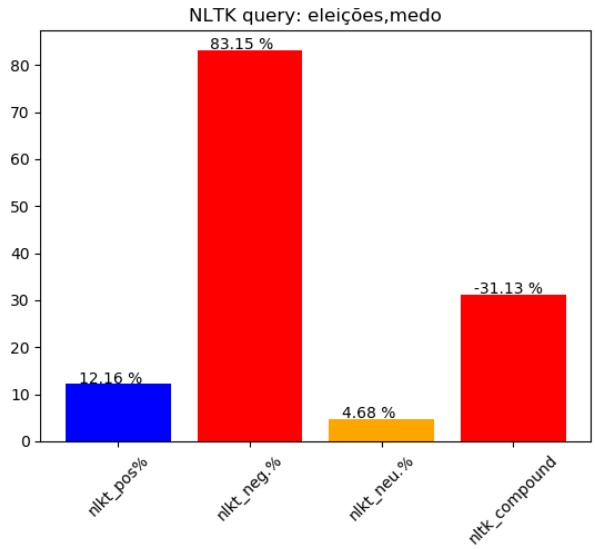


Aqui podemos identificar um interessante a ser observado sobre a análise do TextBlob, a maioria dos tweets são positivos, porém os negativos, mesmo em menor quantidade, tem intensidade maior.

O NLTK também aponta para a mesma conclusão, textos com sentimentos negativos com alta intensidade.

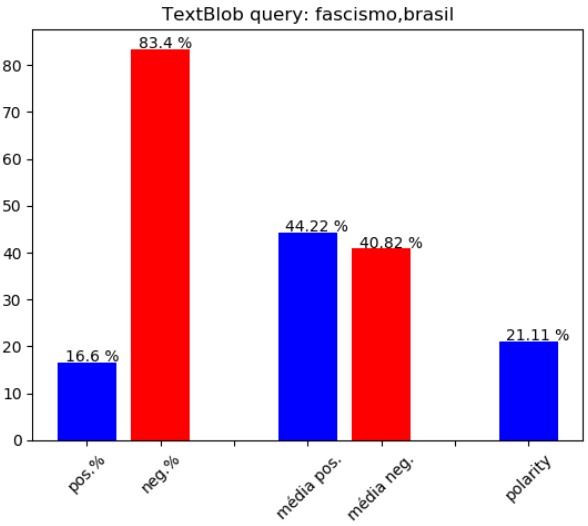
Query: "eleições,medo"

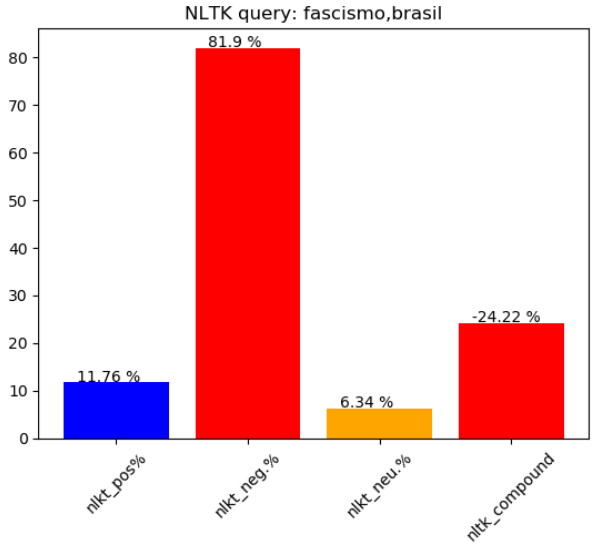




Novamente para uma combinação semelhante, a quantidade de textos positivos são maiores, porém os negativos tem mais intensidade. O resultado do NLTK também foi semelhante ao resultado anterior.

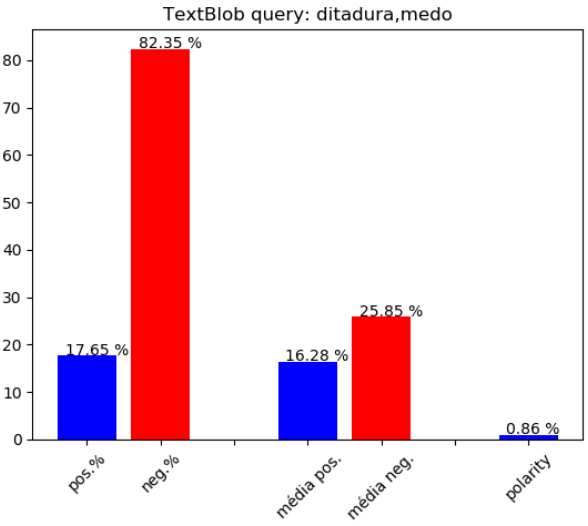
Query: "fascismo,brasil"

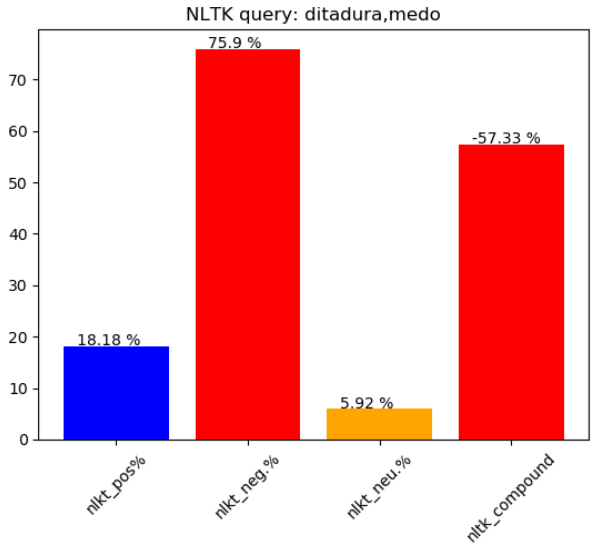




Resultado semelhante ao da query 'fascismo'. Para esse resultado foi necessário fazer outra busca, pois com "facismo,brasil" não encontramos nada, mudamos para "fascismo,brasil".

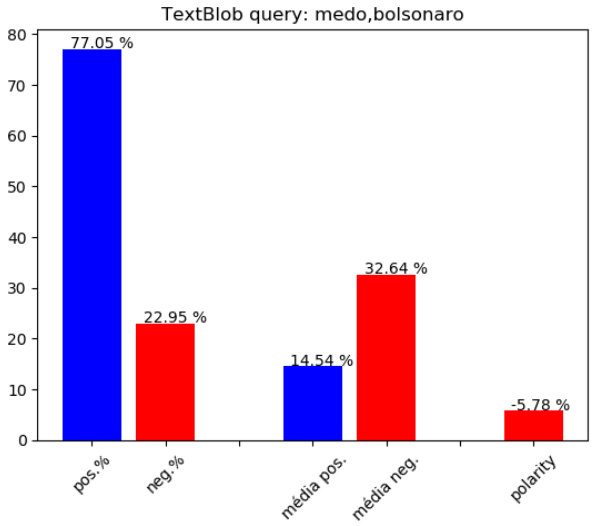
Query: "ditadura,medo"

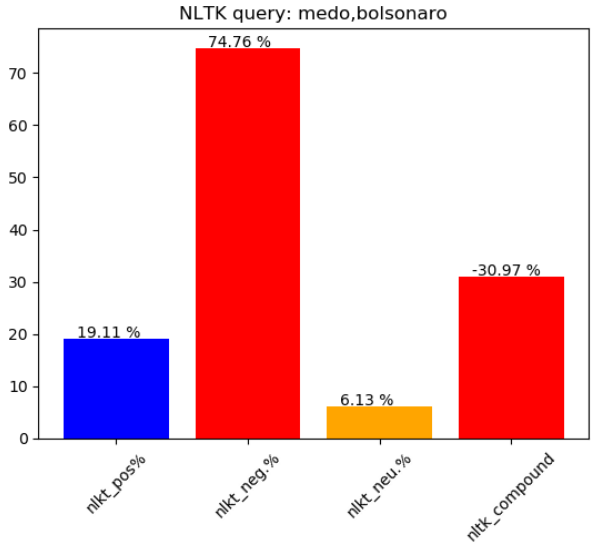




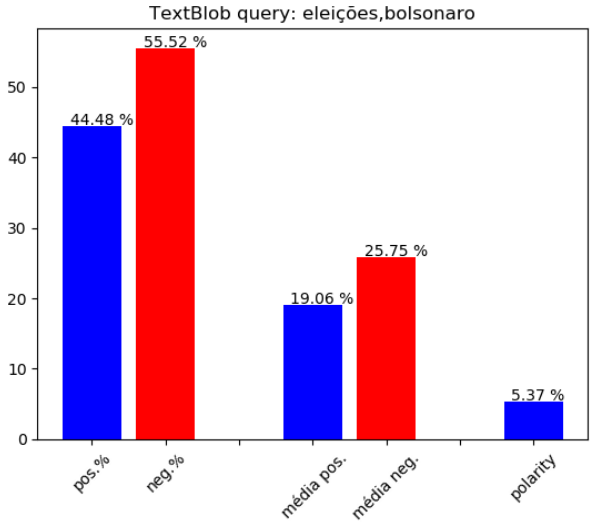
Ambas ferramentas de análise indicam um sentimento muito negativo relacionado a textos que mencionam "ditadura" e "medo". Conclusão pertinente e bastante plausível.

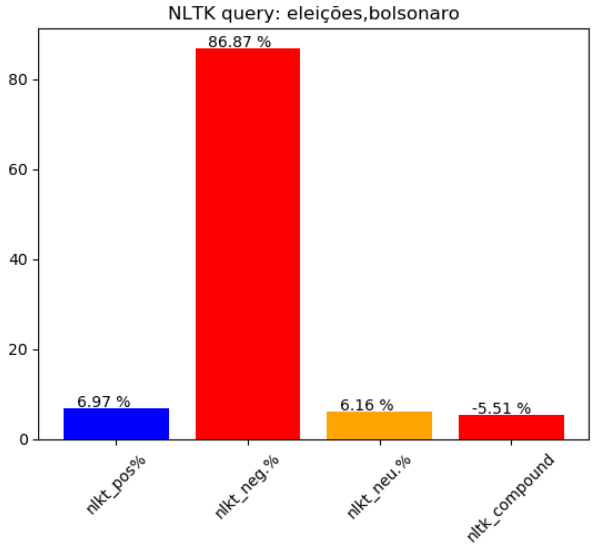
Query: "medo,bolsonaro"





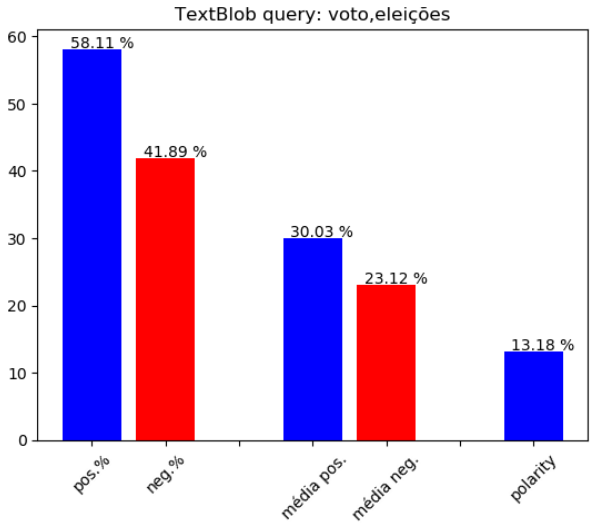
Query: "eleições,bolsonaro"

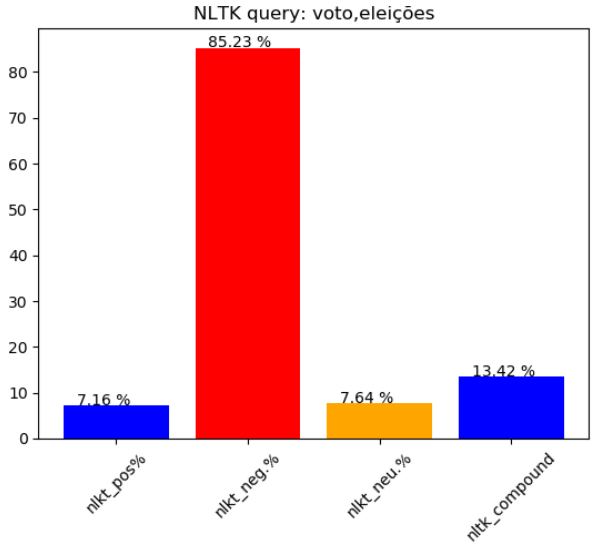




Esses tweets foram obtidos no dia 05/11/2018 por volta das 20 hs (informação no banco de dados) foram adquiridos 1976 tweets com essa combinação. As duas ferramentas apontam um sentimento negativo em relação a isso.

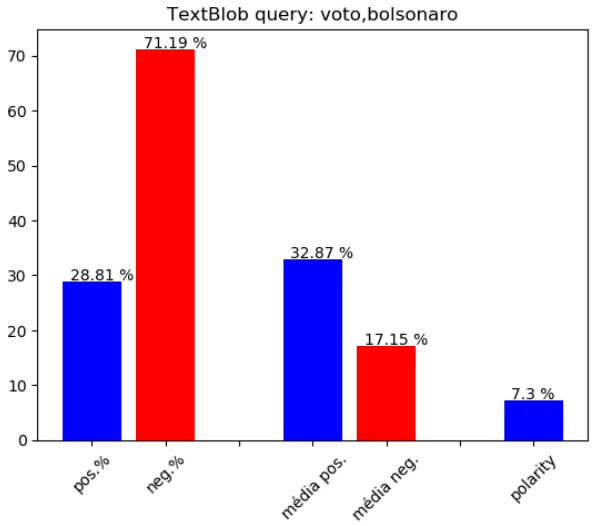
Query: "voto,eleições"

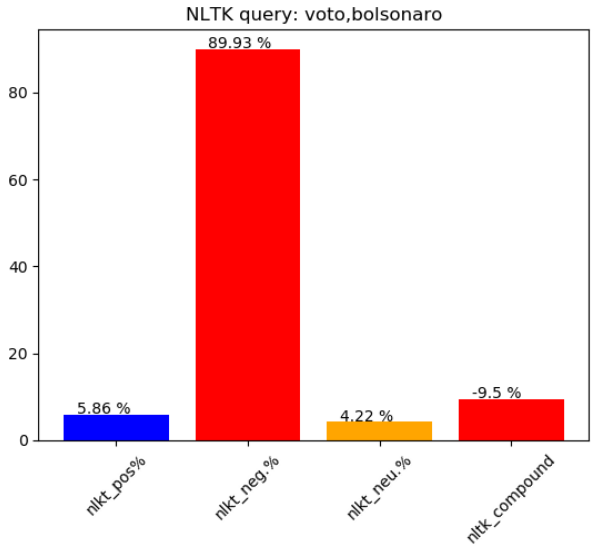




As ferramentas de analise do sentimento, mostraram conclusões diferentes TextBlob aponta para sentimentos mais positivos que negativos, e NLTK aponta para sentimentos negativos. Encontramos 869 tweets com essa combinação.

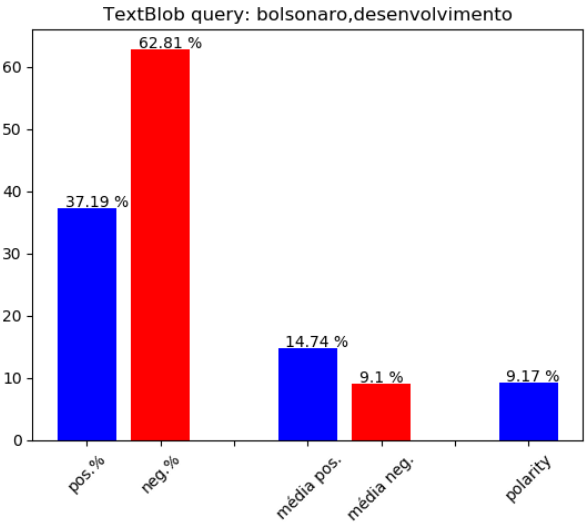
Query: "voto,bolsonaro"

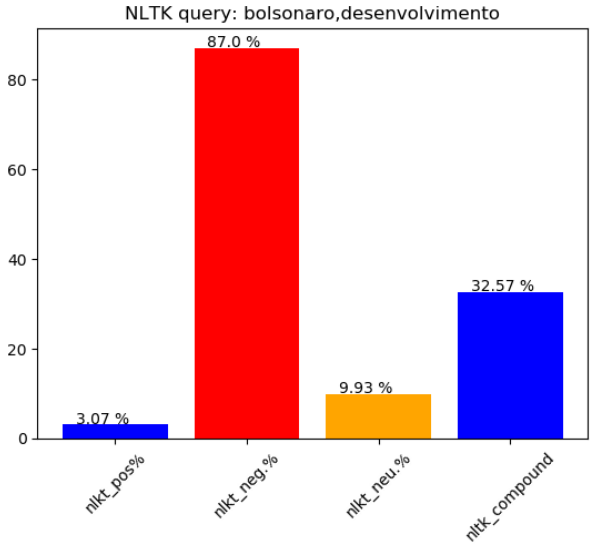




Ambas ferramentas indicam analise de sentimento como negativo. 1982 registros.

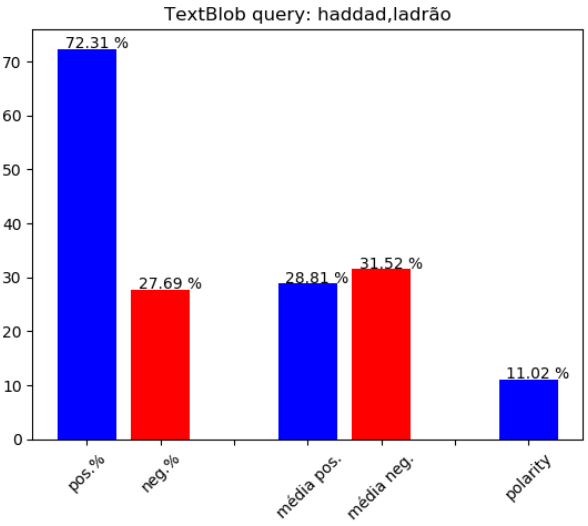
Query: "bolsonaro,desenvolvimento"

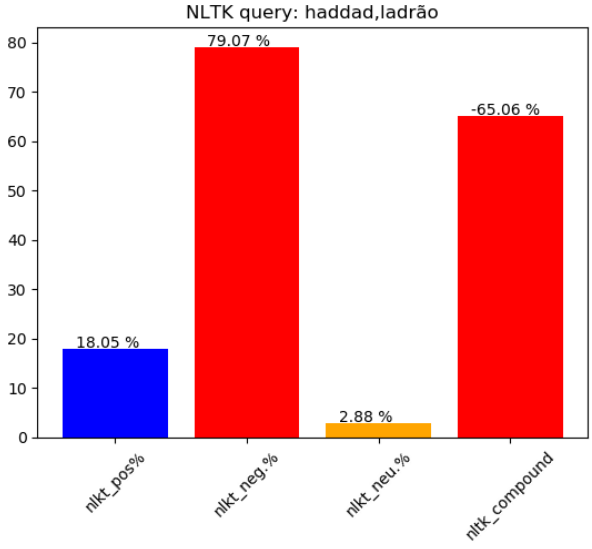




Ambas ferramentas apontam para sentimento negativo. Foram analisado 199 registros, na data especificada não encontramos mais tweets com essa combinação.

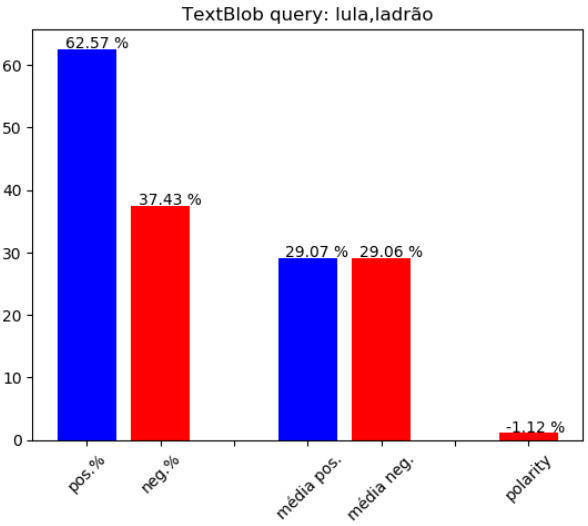
Query: "haddad,ladrão"

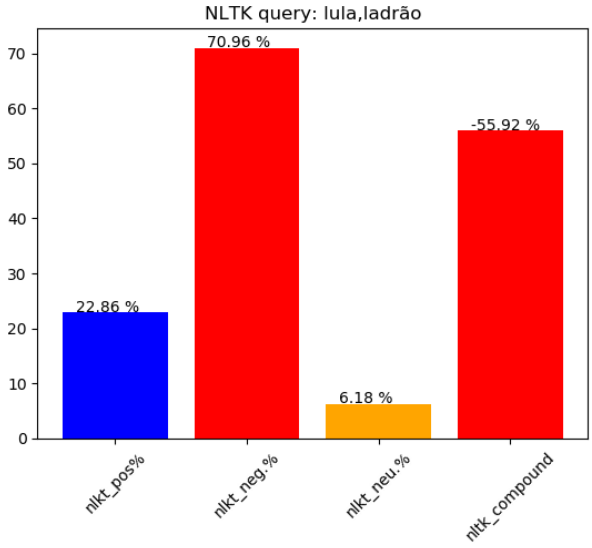




Textblob indica mais positivos do que negativos, e uma intensidade equilibrada de positivos e negativos. Já o NLTK aponta para sentimentos bem mais negativos do que o Textblob. No momento pesquisa com essa combinação encontramos 260 tweets.

Query: "lula,ladrão"

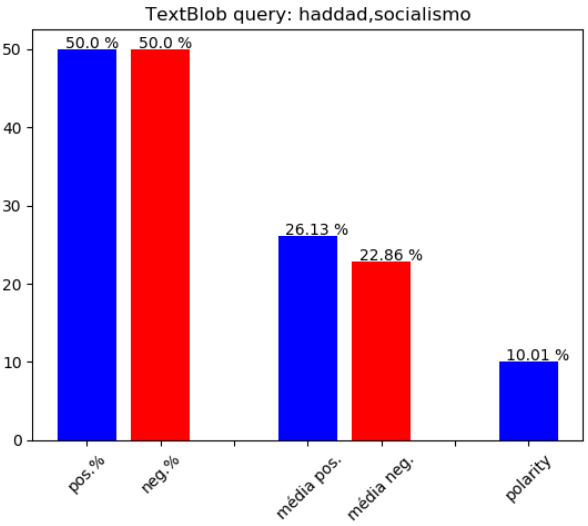


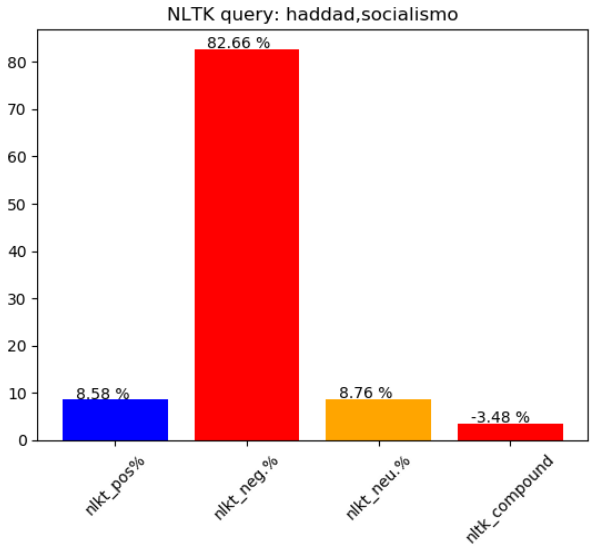


TextBlob classificou maior quantidade como positivos e intensidade equilibrada, possivelmente tweets discordando de que Lula é ladrão e fazendo elogios.

Já o NLTK classifica como mais para negativo. Foram analisados 724 registros de tweets.

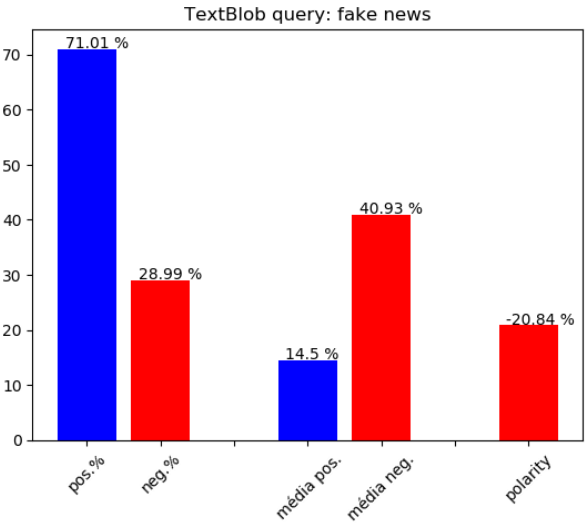
Query: "haddad,socialismo"

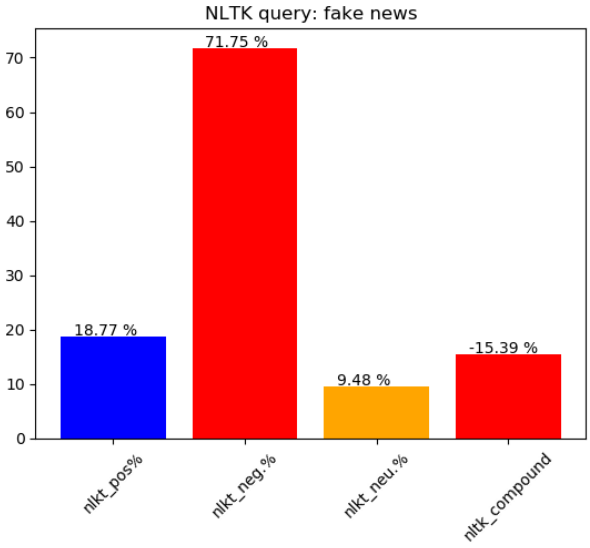




Foram analisados 182 tweets, Textblob classificou metade positiva e metade negativa. Já o NLTK classificou como muto negativo.

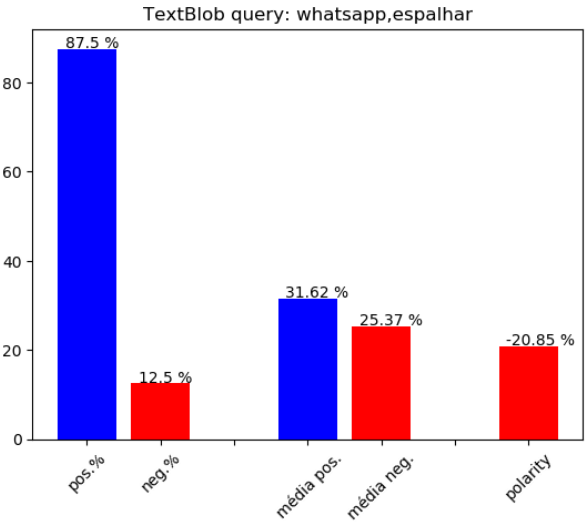
Query: "fake news"

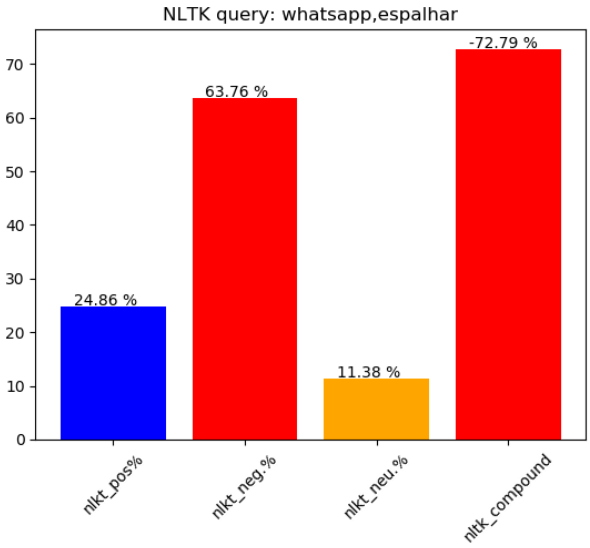




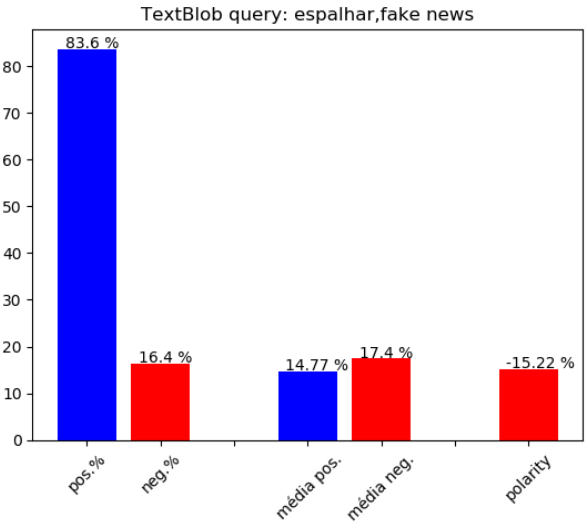
Analisados 1994 registros, TextBlob classificou mais positivos do que negativos, porém os negativos tem maior intensidade. NLTK identificou mais sentimentos negativos.

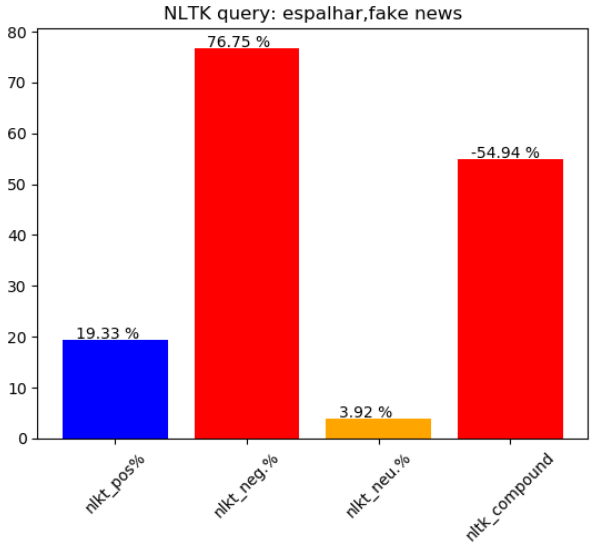
Query: "whatsapp,espalhar"





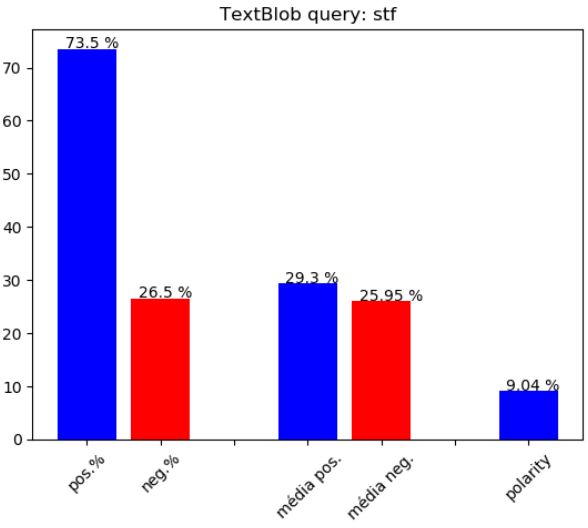
Query: "espalhar,fake news"

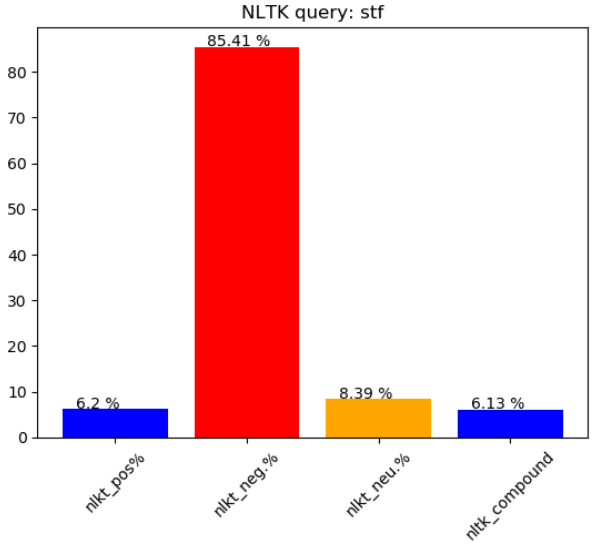




Mais uma vez o TextBlob mostra resultados divergentes do NLTK aparentemente o NLTK tem resultados mais plausiveis e aceitaveis. Foram analisados 573 registros

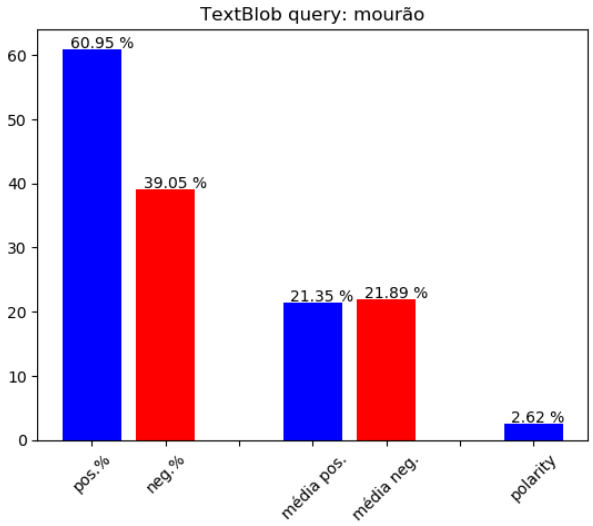
Query: "stf"

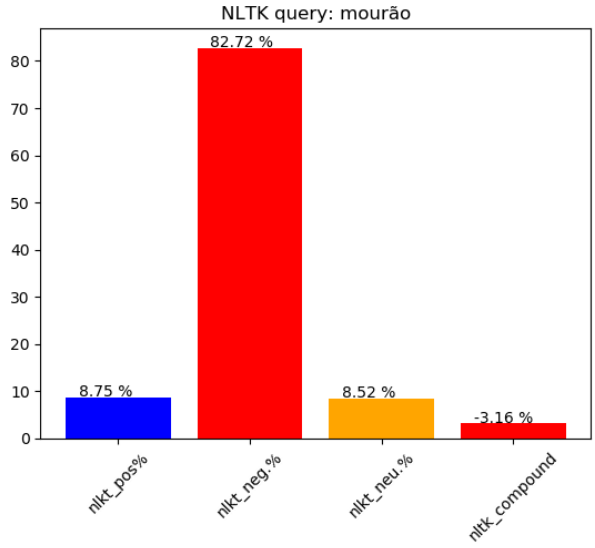




Na primeira busca por essa palavra não foram salvos registros no banco de dados. Feita uma busca especifica com 2000 registros. TextBlob apresenta sentimento ligeiramente positivo, e NLTK apresenta muito negativo.

Query: "mourão"





4 CONCLUSÕES

Após realizar todas buscas e analisar os resultados obtidos, podemos concluir que o NLTK utilisando *sentiment.vader* apresenta resultados melhores que o TextBlob. Mesmo tendo utilizado duas formas de análise com o TextBlob, seus resultados nem sempre representam o real sentimento expressado no texto. Possivelmente se utilizasse uma base de dados com mais palavras e também implementar a classificação 'neu' para o TextBlob, teríamos resultados melhores. Mesmo as analises com texto em inglês, o NLTK mostra resultados mais plausíveis e confiáveis.

5 CÓDIGO

O projeto completo e todas as bases de dados utilizadas está disponível em :
`https://github.com/walkerfelipe/twitter_analise_sentimentos_pt`

5.1 Script de busca (buscatweets.py)

```
# -*- encoding: utf-8 -*-
import tweepy
import sqlite3
'''
Esse script busca os tweets e salva em banco de dados local
utilizando sqlite3
'''
#Palavras buscadas
```

```
query
↪ =['ditadura','facismo','Brasil,medo','eleições,medo','facism
↪ o,brasil','ditadura,medo','medo,bolsonaro','eleições,bolsona
↪ ro','voto,eleições','voto,bolsonaro','bolsonaro,desenvolvime
↪ nto','haddad,ladrão','lula,ladrão','haddad,socialismo','fake
↪ news','whatsapp,espalhar','espalhar,fake
↪ news','governo,povo','mourão,militar','manuela
↪ d\'ávila','stf','mourão']
# numero maximo de tweets para cada palavra( para o trabalho
↪ buscamos 2000)
max_tweets = 100

#Conexão com banco de dados
conn = sqlite3.connect('tweets.db')
c = conn.cursor()
#criando tabelas se não existirem
c.execute('''Create TABLE if not exists
↪ tweets("display_name","name","text","data","query",id
↪ PRIMARY KEY)''')

#função de inserir no banco
def data_entry():
    for i in range(len(tweets_encontrados)):
        try:
            if 'retweeted_status' in dir(tweets_encontrados[i]):
                c.execute("INSERT INTO tweets('name','display_n
                ↪ ame','text','data','query','id')
                ↪ VALUES('"+tweets_encontrados[i].user.screen_
                ↪ name.replace("'","")+"','"+tweets_encontrado
                ↪ s[i].user.name.replace("'","")+"','"+tweets_
                ↪ encontrados[i].retweeted_status.full_text.re
                ↪ place("'","")+"','"+str(tweets_encontrados[i
                ↪ ].created_at)+"','"+query[n]+"','"+str(tweet
                ↪ s_encontrados[i].id)+"')")
            else:
                c.execute("INSERT INTO tweets('name','display_n
                ↪ ame','text','data','query','id')
                ↪ VALUES('"+tweets_encontrados[i].user.screen_
                ↪ name.replace("'","")+"','"+tweets_encontrado
                ↪ s[i].user.name.replace("'","")+"','"+tweets_
                ↪ encontrados[i].full_text.replace("'","")+"',
                ↪ '"+str(tweets_encontrados[i].created_at)+"',
                ↪ '"+query[n]+"','"+str(tweets_encontrados[i].
                ↪ id)+"')")
            conn.commit()
        except:
            pass

#Autenticação do Twitter
```

```
auth = tweepy.OAuthHandler("gfU74Yf0e52dR7QTjhqlmHiEO",
↪ "yRSpI5dTZVs0AMocMwUwO5cw0vhNsazS7YtQzTk8Wob8t3RB4b")
auth.set_access_token("2170605474-XyPs0zpvCrl27d11TWGDhbEAklotE
↪ lJc48Y3bhP",
↪ "5pHPIkKF2KreBB2ZyBpfs2WMBgCVHMCsv2fDtQXPu9CZ9")
api = tweepy.API(auth)
#parte de busca dos tweets
for n in range(len(query)):
    print("Buscando por "+query[n]+" -- "+str(n+1)+" de
    ↪ "+str(len(query)))
    count=0
    tweets_encontrados = []
    last_id = -1
    while len(tweets_encontrados) < max_tweets:

        count = max_tweets - len(tweets_encontrados)
        try: #until='2018-11-02' # parametro, buscar antes
        ↪ dessa data ( max ~ 1 semana)
            new_tweets = api.search(q=query[n],lang = 'pt'
            ↪ ,count=count,tweet_mode='extended',max_id=str(la
            ↪ st_id -
            ↪ 1))
            if not new_tweets:
                break
            tweets_encontrados.extend(new_tweets)
            last_id = new_tweets[-1].id
        except tweepy.TweepError as e:
            # Desistir caso haja algum erro
            break
    data_entry()
    del tweets_encontrados
conn.close() # close conexão com banco de dados
```

5.2 Script de análise (classification.py)

```
#como setar variavel do google translator ( arquivo json):
#$env:GOOGLE_APPLICATION_CREDENTIALS="C:\Users\username\Downloa
↪ ds\[FILE_NAME].json"
#necessário criar variavel antes de iniciar o python(
↪ recomendado utilizar powershell)
import sqlite3
from textblob import TextBlob
from textblob.classifiers import NaiveBayesClassifier
import os
import re
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon') # necessário apenas na primeira
↪ vez
```

```
sid = SentimentIntensityAnalyzer()

# Importando biblioteca de tradução do Google Cloud
from google.cloud import translate

# Instanciando o cliente, essa parte só vai funcionar
# se o arquivo json tiver sido setado na variavel
↪ GOOGLE_APPLICATION_CREDENTIALS ( no S.O ).
try:
    translate_client = translate.Client()
except:
    pass
# necessário ser a base gerada pelo 'buscatweets.py'
conn = sqlite3.connect('tweets.db')
c = conn.cursor()

# Adicionando colunas para salvar o resultado das analises de
↪ sentimento
try:
    c.execute('''ALTER TABLE tweets ADD COLUMN polarity
    ↪ DOUBLE''')
    conn.commit()
except:
    pass

try:
    c.execute('''ALTER TABLE tweets ADD COLUMN subjectivity
    ↪ DOUBLE''')
    conn.commit()
except:
    pass

try:
    c.execute('''ALTER TABLE tweets ADD COLUMN pos_neg TEXT''')
    conn.commit()
except:
    pass

try:
    c.execute('''ALTER TABLE tweets ADD COLUMN text_en TEXT''')
    conn.commit()
except:
    pass
try:
    c.execute('''ALTER TABLE tweets ADD COLUMN NLTK_neg
    ↪ DOUBLE''')
    conn.commit()
except:
    pass
```

```
try:
    c.execute('''ALTER TABLE tweets ADD COLUMN NLTK_neu
    ↪ DOUBLE''')
    conn.commit()
except:
    pass

try:
    c.execute('''ALTER TABLE tweets ADD COLUMN NLTK_pos
    ↪ DOUBLE''')
    conn.commit()
except:
    pass

try:
    c.execute('''ALTER TABLE tweets ADD COLUMN NLTK_compound
    ↪ DOUBLE''')
    conn.commit()
except:
    pass

#carregando a base de dados
c.execute("SELECT * FROM tweets")
rows = c.fetchall()

'''
 Treinando o TextBlob para analisar textos em portugues.
 Essa forma, retorna apenas 'pos'(positiv) e 'neg'(negativo)
↪ para os textos analisados.
 Os arquivos foram extraidos de  ReLi (REsenha de LIvros)
↪ https://www.linguateca.pt/Repositorio/ReLi/

 '''
base_path = 'ReLi-Lex'
train = []
files = [os.path.join(base_path, f) for f in
↪ os.listdir(base_path)]

for file in files:
    t = 'pos' if '_Positivos' in file else 'neg'
    with open(file, 'r') as content_file:
        content = content_file.read()
        all = re.findall('\[.*?\]',content)
        for w in all:
            train.append((w[1:-1], t))

cl = NaiveBayesClassifier(train)
```

```
#Analise com TextBlob em pt ( utilizando treinamento )
print("Processando Polarity 'pos' ou 'neg':")
for i in range(len(rows)):
    print(str(i)+" de "+str(len(rows)) )
    blob=TextBlob(rows[i][2],classifier=cl)
    resp=blob.classify()
    c.execute("UPDATE tweets SET pos_neg='"+resp+"' WHERE
    ↪ id='"+rows[i][5]+"' ")
    conn.commit()

    # traduzindo para 'en' e aplicando a analise do textblob(
    ↪ nativa em ingles)
print("Processando subjectivity e polarity:")
for i in range(len(rows)):
    print(str(i)+" de "+str(len(rows)) )
    # Forma antiga, onde usa tradutor do proprio blob ( é
    ↪ limitado ... )

    try:
        texto_en=translate_client.translate(rows[i][2],target_l
        ↪ anguage='en')
        blob=TextBlob(texto_en['translatedText'])
    except:
        blob=TextBlob(rows[i][2])
        blob=blob.translate(to="en")

    # caso utilize a API do google cloud, comente as 2 linhas
    ↪ acima, e descomente as 2 abaixo

    ####
    c.execute("UPDATE tweets SET
    ↪ subjectivity='"+str(blob.sentiment.subjectivity)+"',pola
    ↪ rity='"+str(blob.sentiment.polarity)+"' WHERE
    ↪ id='"+rows[i][5]+"' ")
    c.execute("UPDATE tweets SET text_en
    ↪ ='"+str(blob).replace("'","")+"' WHERE
    ↪ id='"+rows[i][5]+"' ")
    conn.commit()

c.execute("SELECT * FROM tweets")
rows = c.fetchall()

# Analisando Com NLTK, tambem em ingles. Textos traduzidos
↪ anteriormente já foram salvos
print("Processing NLTK:")
for i in range(len(rows)):
    print(str(i)+" de "+str(len(rows)) )
    text_en=str(rows[i][9])
    ss=sid.polarity_scores(text_en)
```

```
    c.execute("UPDATE tweets SET NLTK_neg='"+str(ss['neg'])+"',
    ↪ NLTK_neu='"+str(ss['neu'])+"',
    ↪ NLTK_pos='"+str(ss['pos'])+"',NLTK_compound='"+str(ss['c
    ↪ ompound'])+"' WHERE id='"+rows[i][5]+"'
    ↪ ")
    conn.commit()

conn.close()
```

5.3 Script calcula médias e salva gráficos (calcmedias.py)

```
# -*- encoding: utf-8 -*-
'''
Esse script tira as médias e plota graficos dos resultados
↪ anteriores
salvos no banco indicado
'''
import sqlite3
import re
import os

try:
    os.stat("images")
except:
    os.mkdir("images")

def column(matrix, i):
    return [row[i] for row in matrix]

conn = sqlite3.connect('tweets.db')

c = conn.cursor()

c.execute("SELECT query FROM tweets")
rows2= c.fetchall()
rows2=column(rows2,0)
c.execute("SELECT distinct(query) FROM tweets")
rows = c.fetchall()
query=column(rows,0)
c.execute("SELECT * FROM tweets")
rows = c.fetchall()

pos_=[0]*len(query)
neg_=[0]*len(query)
subjectivy=[0]*len(query)
polarity=[0]*len(query)
polarity_pos=[0]*len(query)
count_pos=[0]*len(query)
polarity_neg=[0]*len(query)
```

```
count_neg=[0]*len(query)
#11
nltk_neg=[0]*len(query)
#12
nltk_neu=[0]*len(query)
#10
nltk_pos=[0]*len(query)
#13
nltk_compound=[0]*len(query)
for i in range(len(query)):
    z=0
    for k in range(len(rows)):
        if rows[k][4]==query[i]:
            nltk_neg[i]+=rows[k][11]
            nltk_neu[i]+=rows[k][12]
            nltk_pos[i]+=rows[k][10]
            nltk_compound[i]+=rows[k][13]
            if rows[k][6]>0:
                polarity_pos[i]+=rows[k][6]
                count_pos[i]+=1
            elif rows[k][6]<0:
                polarity_neg[i]+=rows[k][6]
                count_neg[i]+=1
            polarity[i]+=rows[k][6]
            subjectivy[i]+=rows[k][7]
            if rows[k][8]=='pos':
                pos_[i]+=1
            else:
                neg_[i]+=1
for i in range(len(query)):
    try:
        quant=rows2.count(query[i])
        if count_neg[i]!=0:
            polarity_neg[i]=polarity_neg[i]/count_neg[i]
        if count_pos[i]!=0:
            polarity_pos[i]=polarity_pos[i]/count_pos[i]
        polarity[i]=polarity[i]/quant
        subjectivy[i]=subjectivy[i]/quant
        nltk_neg[i]=nltk_neg[i]/quant
        nltk_neu[i]=nltk_neu[i]/quant
        nltk_pos[i]=nltk_pos[i]/quant
        nltk_compound[i]=nltk_compound[i]/quant
        temp=(pos_[i]*100)/(pos_[i]+neg_[i])
        neg_[i]=(neg_[i]*100)/(pos_[i]+neg_[i])
        pos_[i]=temp
    except:
        pass
arquivo = open('Medias.txt', 'w')
for i in range(len(query)):
```

```
    arquivo.write(str(query[i]))
    arquivo.write("  ==============")
    arquivo.write("\n")
    arquivo.write("nltk_neg = ")

    arquivo.write(str(nltk_neg[i]))
    arquivo.write("\n")
    arquivo.write("nltk_neu = ")

    arquivo.write(str(nltk_neu[i]))
    arquivo.write("\n")
    arquivo.write("nltk_pos = ")


    arquivo.write(str(nltk_compound[i]))
    arquivo.write("\n")
    arquivo.write("nltk_compound = ")

    arquivo.write(str(nltk_pos[i]))
    arquivo.write("\n")
    arquivo.write("subjectivy = ")


    arquivo.write(str(subjectivy[i]))
    arquivo.write("\n")
    arquivo.write("polarity = ")

    arquivo.write(str(polarity[i]))
    arquivo.write("\n")
    arquivo.write("polarity_neg = ")

    arquivo.write(str(polarity_neg[i]))
    arquivo.write("\n")
    arquivo.write("polarity_pos = ")

    arquivo.write(str(polarity_pos[i]))
    arquivo.write("\n")
    arquivo.write("pos_ = ")

    arquivo.write(str(pos_[i]))
    arquivo.write("\n")
    arquivo.write("neg_ = ")

    arquivo.write(str(neg_[i]))
    arquivo.write("\n")
    arquivo.write("\n")
arquivo.close()
############### matplot lib
```

```
import numpy as np
import matplotlib.pyplot as plt
for i in range(len(query)):
    fig = plt.figure(1)
    ax = fig.add_subplot(111)

    legendas=['pos.%','neg.%','',' média pos.','média
    ↪ neg.','','polarity']

    ax.bar(1,neg_[i],color="red")
    ax.text(1-0.3,neg_[i]+0.2,str(round(neg_[i],2))+" %")

    ax.bar(0,pos_[i],color="blue")
    ax.text(0-0.3,pos_[i]+0.2,str(round(pos_[i],2))+" %")


    ax.bar(3,polarity_pos[i]*100,color="blue")
    ax.text(3-0.3,polarity_pos[i]*100+0.2,str(round(100*polarit
    ↪ y_pos[i],2))+"
    ↪ %")

    ax.bar(4,-1*polarity_neg[i]*100,color="red")
    ax.text(4-0.3,-1*polarity_neg[i]*100+0.2,str(round(-100*pol
    ↪ arity_neg[i],2))+"
    ↪ %")

    if (polarity[i]>=0):
        ax.bar(6,abs(polarity[i])*100,color="blue")
        ax.text(6-0.3,abs(polarity[i])*100+0.2,str(round(100*po
        ↪ larity[i],2))+"
        ↪ %")
    else:
        ax.bar(6,abs(polarity[i])*100,color="red")
        ax.text(6-0.3,abs(polarity[i])*100+0.2,str(round(100*po
        ↪ larity[i],2))+"
        ↪ %")
    plt.title('TextBlob query: '+query[i])
    plt.xticks(range(len(legendas)),legendas,rotation=45)
    fig.savefig(str('./images/textblob_'+query[i]),
    ↪ bbox_inches='tight')
    plt.cla()   # Clear axis
    plt.clf()   # Clear figure
    plt.close() # Close a figure window
    #plt.show()
    ############### NLKT
    fig = plt.figure(1)
    ax = fig.add_subplot(111)
```

```
    legendas=['nlkt_pos%','nlkt_neg.%','nlkt_neu.%','nltk_compo
    ↪ und']

    ax.bar(0,100*nltk_pos[i],color="blue")
    ax.text(0-0.3,100*nltk_pos[i]+0.2,str(round(100*nltk_pos[i]
    ↪ ,2))+"
    ↪ %")

    ax.bar(1,100*nltk_neg[i],color="red")
    ax.text(1-0.3,100*nltk_neg[i]+0.2,str(round(100*nltk_neg[i]
    ↪ ,2))+"
    ↪ %")

    ax.bar(2,100*nltk_neu[i],color="orange")
    ax.text(2-0.3,100*nltk_neu[i]+0.2,str(round(100*nltk_neu[i]
    ↪ ,2))+"
    ↪ %")

    if nltk_compound[i]>=0:
        ax.bar(3,100*abs(nltk_compound[i]),color="blue")
        ax.text(3-0.3,abs(100*nltk_compound[i])+0.2,str(round(1
        ↪ 00*nltk_compound[i],2))+"
        ↪ %")
    else:
        ax.bar(3,100*abs(nltk_compound[i]),color="red")
        ax.text(3-0.3,abs(100*nltk_compound[i])+0.2,str(round(1
        ↪ 00*nltk_compound[i],2))+"
        ↪ %")

    plt.title('NLTK query: '+query[i])
    plt.xticks(range(len(legendas)),legendas,rotation=45)
    fig.savefig(str('./images/nltk_'+query[i]),
    ↪ bbox_inches='tight')
    plt.cla()   # Clear axis
    plt.clf()   # Clear figure
    plt.close() # Close a figure window
```

6 REFERÊNCIAS

- https://textblob.readthedocs.io/en/dev/#
- https://www.nltk.org/
- https://www.linguateca.pt/Repositorio/ReLi/
- http://t-redactyl.io/blog/2017/04/using-vader-to-handle-sentiment-analysis-with-social-media-text.html