Apêndice – Código fonte – script para extração de tweets.

Para execução do código fonte é necessário ter instalado:

Python 3.0, disponível em https://www.python.org/
Google Chrome
Selenium webdriver, disponível em https://www.selenium.dev/

Arquivos de pacotes requeridos do Python (requirements.txt)

```
beautifulsoup4==4.9.3
certifi==2020.12.5
chardet==3.0.4
colorama==0.4.4
configparser==5.0.1
crayons==0.4.0
et-xmlfile==1.0.1
idna==2.10
install==1.3.4
jdcal==1.4.1
MouseInfo==0.1.3
numpy==1.19.3
openpyxl==3.0.5
pandas==1.1.4
PyAutoGUI==0.9.52
PyGetWindow==0.0.9
PyMsgBox==1.0.9
pyperclip==1.8.1
PyRect==0.1.4
PyScreeze==0.1.26
python-dateutil==2.8.1
PyTweening==1.0.3
pytz==2020.4
requests==2.25.0
selenium==3.141.0
six==1.15.0
soupsieve==2.0.1
urllib3==1.26.2
webdriver-manager==3.2.2
```

Arquivo com código fonte (scrapy.py):

```
#! usr/bin/python3

from bs4 import BeautifulSoup
import time
from csv import DictWriter
import pprint
import datetime
import pickle
from datetime import datetime as DT
from datetime import date, timedelta
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException

from webdriver_manager.chrome import ChromeDriverManager

import re
import pandas as pd
import pyautogui
import NomArq
import os

# Validar se internet está presente
try:
    import httplib
```

```
except:
    import http.client as httplib

def checkInternetHttplib(url="www.google.com", timeout=10):
    conn = httplib.HTTPConnection(url, timeout=timeout)
    try:
        conn.request("HEAD", "/")
        conn.close()
        return True
    except Exception as e:
        print(e)
        return False


DIR_ATUAL = os.getcwd()
URL_INTERNET_OK = 'twitter.com'
SEGUNDOS_ATE_INTERNET_OK = 900 # 15 minutos
SEGUNDOS_RETESTE_INTERNET = 300 # 5 MINUTOS



def init_driver(driver_type):
    if driver_type == 1:
        driver = webdriver.Firefox()
    elif driver_type == 2:
        #driver =
webdriver.Chrome(ChromeDriverManager().install())
        try:
            driver = webdriver.Chrome()
        except:
            driver =
webdriver.Chrome(ChromeDriverManager().install())
    elif driver_type == 3:
        driver = webdriver.Ie()
    elif driver_type == 4:
        driver = webdriver.Opera()
    elif driver_type == 5:
        driver = webdriver.PhantomJS()
    driver.wait = WebDriverWait(driver, 5)
    return driver


def scroll(driver, start_date, end_date, words, lang, max_time,
caminhoArqIESHtml, IES):
    print('Entrou no Scroll')

    languages = { 1: 'en', 2: 'it', 3: 'es', 4: 'fr', 5: 'de', 6:
'ru', 7: 'zh', 8:'pt'}
    url = "https://twitter.com/search?q="
    for w in words[:-1]:
        url += "{}%20OR".format(w)
    url += "{}%20".format(words[-1])
    url += "since%3A{}%20until%3A{}&".format(start_date, end_date)
    if lang != 0:
        url += "l={}&".format(languages[lang])
    url += "src=typd"

    print(url)
    driver.get(url)

    start_time = time.time()  # remember when we started
    while (time.time() - start_time) < max_time:
```

```
        driver.execute_script("window.scrollTo(0,
document.body.scrollHeight);")
        #tempo = time.time() - start_time
         #print(f'---> Tempo Atual: {tempo}')

    for contatab in range(50):
        pyautogui.press('tab')

    return url


def scrape_tweets(url, driver, df):
    print('vai fazer scrape_tweets - fazendo backup')
    # Grava um backup do que já foi feito
    dfBk = df.copy(deep=True)

    try:
        tweet_divs = driver.page_source
        obj = BeautifulSoup(tweet_divs, "html.parser")
        #content = obj.find_all("div", class_="content")
        content = obj.find_all("article")
        print(f'--> Nro. Articles {len(content)}')

        for i in content:

            try:
                article = i
            except:
                article = ''

            try:
                tweet_bruto = i.find_all("div", {"lang" :
"pt"})[0]
            except:
                tweet_bruto = ''

            try:
                emoticons = i.find_all("img")
                emoticon = ''
                for e in emoticons:
                    emoticon = emoticon + "|" + e['alt']
                print(f'emoticon: {emoticon} - fim emoticon')
            except:
                print('Erro nos emoticons')
                emoticon = ''

            try:
                date_descr =
(i.find_all('time')[0].string).strip()
                date = i.find_all('time')[0]['datetime']
                print(f'-->> data: {date} - {date_descr}')

                try:
                    url_tweet =
i.find_all('time')[0].find_parent('a')['href']
                    #print(f'url_tweet: {url_tweet}')
                except:
                    #print(f'--> Erro pegar urt tweet em: {i}')
                    url_tweet='erro ao pegar url tweet'
            except:
                #print(f'--> Erro pegar data em: {i}')
                date_descr = ''
```

```
                url_tweet = ''
                date = ''

            try:
                #name = (i.find_all("strong",
class_="fullname")[0].string).strip()
                name =
(i.find_all(string=re.compile("^@"))[0].string).strip()
                print(f'-->> name: {name} - Fim name')
            except AttributeError:
                name = ''

            try:
                resposta = i.find_all(attrs={"data-testid":
"reply"})[0]['aria-label']
            except:
                #print(f'--> Erro pegar resposta em: {i}')
                resposta = ''

            try:
                retweet = i.find_all(attrs={"data-testid":
"retweet"})[0]['aria-label']
            except:
                #print(f'--> Erro pegar retweet em: {i}')
                retweet = ''

            try:
                curtida = i.find_all(attrs={"data-testid":
"like"})[0]['aria-label']
            except:
                #print(f'--> Erro pegar retweet em: {i}')
                curtida = ''

            try:
                tweets = i.find("div").strings
                tweet_text = "".join(tweets)
                #print(f'---> {tweet_text}')
            except:
                print('Deu pau no tweet_text')
                tweet_text = ''

            try:
                #print('vai pegar hashtag')
                hashtags = i.find_all("a", href =
re.compile(r'\/hashtag\/'))
                #print(f'Achou {len(hashtags)} hashtags')
                hashtag_text = ''
                for h in hashtags:
                    #print(f'hashtag: {h.string}')
                    hashtag_text = hashtag_text + ' ' + h.string
            except:
                hashtag_text = ''

            NovaLinha = {
                'url_dia':[url],
                'url_tweet':[url_tweet],
                'data':[date],
                'data_descr':[date_descr],
                'nome':[name],
                'tweet_text':[tweet_text],
                'hashtags':[hashtag_text],
                'resposta':[resposta],
```

```
                'retweet':[retweet],
                'curtida':[curtida],
                'article':[article],
                'tweet_bruto':[tweet_bruto],
                'emoticon':[emoticon]
            }

            dfNovaLinha = pd.DataFrame(data=NovaLinha)

            #print(NovaLinha)
            antesAppend = len(df.index)
            df = pd.concat([df, dfNovaLinha])
            depoisAppend = len(df.index)
            print(f'DF antes {antesAppend}; Depois:
{depoisAppend}')

        # Retorna o DF + Verdadeiro Teste se InernetOk
        return df, True

    except Exception as e:
        print(e)
        print("Whoops! Something went wrong! Retornando dfBk")
        driver.quit()
        # Retorna o DF + Teste se InernetOk
        return dfBk, checkInternetHttplib(URL_INTERNET_OK)


def get_all_dates(start_date, end_date):
    dates = []
    start_date = datetime.datetime.strptime(start_date, "%Y-%m-
%d")
    end_date = datetime.datetime.strptime(end_date, "%Y-%m-%d")
    step = timedelta(days=1)
    while start_date <= end_date:
        dates.append(str(start_date.date()))
        start_date += step

    return dates


def main():

    driver_type = 2
    IES = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
    wordsToSearch = IES.split(',')
    for w in wordsToSearch:
        w = w.strip()
    #start_date = "2013-01-01"
    #end_date = "2019-12-31"

    start_date = "2013-01-01"
    end_date = "2019-12-31"

    # Define os diretórios
    caminhoArq     = 'C:/TwitterAdvSearch'
    caminhoArqIES  = caminhoArq + '/' + IES
    caminhoArqIESHtml = caminhoArqIES + '/html'
    caminhoUltData = caminhoArqIES+'_ult_data.pkl'

    try:
        if os.path.isfile(caminhoUltData):
```

```
            start_date = pickle.load( open(caminhoUltData, "rb" )
)
            print(f'Pegou última data do arquivo {caminhoUltData}:
{start_date}')
    except:
        print(f'Erro ao pegar arquivo {caminhoUltData} da última,
fazendo de: {start_date}')




    # Cria diretórios
    if not os.path.exists(caminhoArq):
        os.mkdir(caminhoArq)
        os.mkdir(caminhoArqIES)
        os.mkdir(caminhoArqIESHtml)
    elif not os.path.exists(caminhoArqIES):
        os.mkdir(caminhoArqIES)
        os.mkdir(caminhoArqIESHtml)
    elif not os.path.exists(caminhoArqIESHtml):
        os.mkdir(caminhoArqIESHtml)

    lang = 0
    all_dates = get_all_dates(start_date, end_date)
    df =
pd.DataFrame(data={'url_dia':[],'url_tweet':[],'data':[],'data_des
cr':[],'nome':[],'tweet_text':[],
                           'hashtags':[],'resposta':[],'retweet':
[],'curtida':[],'article':[],'tweet_bruto':[],''
                           'emoticon':[]
                           })

    for i in range(len(all_dates) - 1):

        # Grava a última data processada no início, se uma data
der erro e travar isso vai fazer pular essa data com problema
        pickle.dump(all_dates[i + 1], open(caminhoUltData, "wb" )
)

        internetOk = True
        FezData = False
        while internetOk and not FezData:
            try:
                driver = init_driver(driver_type)
                url = scroll(driver, str(all_dates[i]),
str(all_dates[i + 1]), wordsToSearch, lang, 3, caminhoArqIESHtml,
IES)

                df, internetOk = scrape_tweets(url, driver, df)
                FezData = True
            except:
                internetOk = False

            while not internetOk: #and (time.time() - start_time)
<= SEGUNDOS_ATE_INTERNET_OK:
                dataHora = DT.now()
                print(f'Problema com internet - Aguardando
{dataHora.hour}h{dataHora.minute}')
                internetOk =
checkInternetHttplib(url=URL_INTERNET_OK)
                if not internetOk:
                    print(f'Sleep {SEGUNDOS_RETESTE_INTERNET}
segundos.')
```

```
                    time.sleep(SEGUNDOS_RETESTE_INTERNET)


        print("Concluído dia {} ".format(all_dates[i]))
        driver.quit()

        print('Vai Gerar Nome Arquivo')

        arqNome =
NomArq.RetArquivo(caminhoArqIES,True,IES,5,'xlsx',1)
        print(f'Vai gravar XLSx: {caminhoArqIES} - {arqNome}')
        #df.to_excel(DIR_ATUAL+'/arquivos/'+arqNome, index=False)
        df.to_excel(caminhoArqIES+'/'+arqNome, index=False)

if __name__ == "__main__":
    main()

```