Study Method

Data Collection

Social Media platform

Twitter is the main social media platform that I have used here because the twitter tweets are some kind of more accurate and reliable than other social media platforms according to the current findings on the Internet. Twitter is mainly used for share valuable information with different parties and it has become the quickest way to share news nowadays.

Therefore, here I will be using twitter tweets as the main data collection.

Dataset Collecting Procedure

Due to COVID-19, Twitter API developers permission application review times might take longer than usual therefore the best available option had was to use selenium to collect tweets out of twitter regarding covid-19 lockdown.

Therefore, the current tweets collecting selenium algorithm is mentioned below:

```
import selenium
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
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,
StaleElementReferenceException
from bs4 import BeautifulSoup as bs
import time


def init_driver():
    # initiate the driver:
    driver = webdriver.Chrome()

    # set a default wait time for the browser [5 seconds here]:
    driver.wait = WebDriverWait(driver, 5)

    return driver


def close_driver(driver):
    driver.close()

    return


def login_twitter(driver, username, password):
    # open the web page in the browser:
    driver.get("https://twitter.com/login")

    # find the boxes for username and password
```

```python
    username_field = driver.find_element_by_class_name("js-username-field")
    password_field = driver.find_element_by_class_name("js-password-field")

    # enter your username:
    username_field.send_keys(username)
    driver.implicitly_wait(1)

    # enter your password:
    password_field.send_keys(password)
    driver.implicitly_wait(1)

    # click the "Log In" button:
    driver.find_element_by_class_name("EdgeButtom--medium").click()

    return


class wait_for_more_than_n_elements_to_be_present(object):
    def __init__(self, locator, count):
        self.locator = locator
        self.count = count

    def __call__(self, driver):
        try:
            elements = EC._find_elements(driver, self.locator)
            return len(elements) > self.count
        except StaleElementReferenceException:
            return False


def search_twitter(driver, query):
    # wait until the search box has loaded:
    box = driver.wait.until(EC.presence_of_element_located((By.NAME, "q")))

    # find the search box in the html:
    driver.find_element_by_name("q").clear()

    # enter your search string in the search box:
    box.send_keys(query)

    # submit the query (like hitting return):
    box.submit()

    # initial wait for the search results to load
    wait = WebDriverWait(driver, 10)

    try:
        # wait until the first search result is found. Search results will be tweets,
which are html list items and have the class='data-item-id':
        wait.until(EC.visibility_of_element_located((By.CSS_SELECTOR, "li[data-item-
id]")))

        # scroll down to the last tweet until there are no more tweets:
        while True:

            # extract all the tweets:
            tweets = driver.find_elements_by_css_selector("li[data-item-id]")

            # find number of visible tweets:
```

```
            number_of_tweets = len(tweets)

            # keep scrolling:
            driver.execute_script("arguments[0].scrollIntoView();", tweets[-1])

            try:
                # wait for more tweets to be visible:
                wait.until(wait_for_more_than_n_elements_to_be_present(
                    (By.CSS_SELECTOR, "li[data-item-id]"), number_of_tweets))

            except TimeoutException:
                # if no more are visible the "wait.until" call will timeout. Catch the
exception and exit the while loop:
                break

        # extract the html for the whole lot:
        page_source = driver.page_source

    except TimeoutException:

        # if there are no search results then the "wait.until" call in the first "try"
statement will never happen and it will time out. So we catch that exception and
return no html.
        page_source = None

    return page_source


def extract_tweets(page_source):
    soup = bs(page_source, 'lxml')

    tweets = []
    for li in soup.find_all("li", class_='js-stream-item'):

        # If our li doesn't have a tweet-id, we skip it as it's not going to be a
tweet.
        if 'data-item-id' not in li.attrs:
            continue

        else:
            tweet = {
                'tweet_id': li['data-item-id'],
                'text': None,
                'user_id': None,
                'user_screen_name': None,
                'user_name': None,
                'created_at': None,
                'retweets': 0,
                'likes': 0,
                'replies': 0
            }

            # Tweet Text
            text_p = li.find("p", class_="tweet-text")
            if text_p is not None:
                tweet['text'] = text_p.get_text()

            # Tweet User ID, User Screen Name, User Name
            user_details_div = li.find("div", class_="tweet")
            if user_details_div is not None:
                tweet['user_id'] = user_details_div['data-user-id']
                tweet['user_screen_name'] = user_details_div['data-screen-name']
                tweet['user_name'] = user_details_div['data-name']

            # Tweet date
            date_span = li.find("span", class_="_timestamp")
            if date_span is not None:
                tweet['created_at'] = float(date_span['data-time-ms'])

            # Tweet Retweets
            retweet_span = li.select("span.ProfileTweet-action--retweet > 
span.ProfileTweet-actionCount")
            if retweet_span is not None and len(retweet_span) > 0:
                tweet['retweets'] = int(retweet_span[0]['data-tweet-stat-count'])

            # Tweet Likes
            like_span = li.select("span.ProfileTweet-action--favorite > 
span.ProfileTweet-actionCount")
            if like_span is not None and len(like_span) > 0:
                tweet['likes'] = int(like_span[0]['data-tweet-stat-count'])

            # Tweet Replies
            reply_span = li.select("span.ProfileTweet-action--reply > 
span.ProfileTweet-actionCount")
            if reply_span is not None and len(reply_span) > 0:
                tweet['replies'] = int(reply_span[0]['data-tweet-stat-count'])

            tweets.append(tweet)

    return tweets


driver = init_driver()

# log in to twitter (replace username/password with your own):
username = "rukshan123"
password = "abcdefg"
login_twitter(driver, username, password)

# search twitter:
query = "covid19 lockdown"
page_source = search_twitter(driver, query)

# extract info from the search results:
tweets = extract_tweets(page_source)

# close the driver:
close_driver(driver)
```