Tesla Keyword Project

By: Rish Pednekar, Aatrey Sahay

Goal/Summary:

Within this project, we are using the New York Times API in order to answer the question of:

> ***What and how many different keyords were written about Tesla in the Technology section from January 1st, 2020 till current?***

In order to answer this question, I used several different techniques within Python and its modules as well the New York Times API. For the modules, I utilized the following:

```
- pathlib
- pickle
- requests
- time
- pandas
- NYT API
```

I also used Python syntax and concepts in the form of:

- variable declaration
- for-loops
- lists
- dataframes
- methods
- f-strings

Explanation of Project:

> **I decided to do this project because I wanted to see how the New York Times suggests articles according to our search query and what it**

> displays. The main goal is to see what other kinds of articles the New York Times shows according to my search query of Tesla. Furthermore, I wanted to see if there was any upspike in Twitter results since Elon Musk bought it recently and see how it compared to Tesla showing up as a keyword compared to Twitter, OpenAI, or SpaceX. Thus, using this program, I was able to see what New York Times suggests in terms of keywords for articles on the search query "Tesla" within the Technology section. Also, I decided to do July 31, 2023 as the end date to account for any unknown articles that might pop up within this month of July.

***Note*: Results will be displayed at the bottom of the document.**

Code Explanation

Introduction

```
import pathlib, pickle, requests, time, pandas as pd

# Q: What and how many different keyords were written about Tesla in the Technology section from January
1st, 2020 till current?


# Creating the API Key
API_KEY = "byYqRTtU1q9wpnK1Gd1bIkbTqWb8eaWm"

# New York Times Link
url = 'https://api.nytimes.com/svc/search/v2/articlesearch.json'

# Choosing the parameters for the call
parameters = {'q': 'Tesla', 'api-key': API_KEY, "fq": "document_type: (article) AND section_name:
(Technology)",
              "begin_date": "20200101",
              "end_date": "20230731", "page": 0}
```

Description:

These first few lines of code are extremely important because they set up the entirety of

the project. The first statement is importing all the neccessary modules to complete the task which I stated previously in the report. Next, I created a variable to acces the “New York Times” API and set it equal to the string value to ensure it is accessible within that variable named “API_Key”. Afterwards, I created a similar statement for the URL in order to access the New York Times and set it equal to another variable. Lastly, the parameters variable which is a dictionary is extremely important because it shows us the parameters of our search query. Thus, I am looking for the keyword “Tesla” marked by the key ‘q’, the API_Key variable which was previously defined and the “fq” key represents the filters of our query. Therefore, within that key I specified the document type which needed to be an article as well the section being Technology. Finally, I set the “begin_date” key as January 1st, 2020 and the end date of July 31, 2023 and the page to start at 0.

API Call Method

```
def api_pull(url_link, parameters1):
    """
     function to run a get request from the New York Times API to extract information about Tesla

     :argument:
        url (str): API endpoint
        parameters (dict): API params
    :return:
        dict: content
    """
    # Getting the requests
    response = requests.get(url_link, params=parameters1)

    # Storing results in a variable named content
    content = response.json()

    return content


# Getting the total results of the query and then calculating important values such as hit number and
page count
results = api_pull(url, parameters)
```

Description:

This method was vital to the success of the project because without it, there would have been no data. Using the requests module and the `.get()` method using the parameters of the url link which will be passed onto the method when it needs to be used later as well as

the `parameters1` argument which is the field that allows the `parameters` variable previously defined to be passed on in the variable `results` which is outside the method. The content variable within the method is there to ensure that a `.json` format is returned in the format of a Python dictionary. Thus, we have the results of the query stored in the variable `results`.

Keyword File Creation

```
number_of_hits = results["response"]["meta"]["hits"]
print(f"Number of Hits: {number_of_hits}")
page_count = number_of_hits // 10
print(f"Page Count: {page_count}\n")

# Creating the keyword files
for i in range(page_count):
    try:
        NYT_tesla = pathlib.Path.cwd() / "NYT_Tesla_keywords"
        NYT_tesla.mkdir(exist_ok=True)
        pg_num = i
        file_name = f"kews{pg_num}"
        parameters["page"] = i
        results = api_pull(url, parameters)
        for documents in results["response"]["docs"]:
            kews = []
            for keys in documents["keywords"]:
                kews.append(keys["value"])
                with open(f"NYT_Tesla_keywords/{file_name}", 'wb') as p_file:
                    pickle.dump(kews, p_file)
            time.sleep(7)
        print(f"Page Number ({pg_num}) completed.")
    except:
        print("Something unknown occurred.")
```

Description:

Here, I created the files from the page numbers of the API and storing the information the API gives us into these pickle binary files. First, I calculated the number of hits that the `results` variable had given us accessing its hit count and then printing out that statement.

Afterwards, I needed to find the page count so I floor divided by 10 to get rid of any remainders.
After finding that the hits matched the number the New York Times gave and figuring out the page count necessary, I needed to actually create the files.

To iterate through the files, I used a for-loop because I knew how many pages there were. To account for any errors during the process, I used a `try-except` statement printing "Something unknown occurred" if there was any issues. The first statement in the `try` is ensuring that my path is in the "NYT_Tesla_keywords" directory and the second statement is making that directory with `exist_ok=True` checking that if the directory already exists to not throw an error. After that, I created files by iterating thrugh the page_num variable and calling it "kews" and then which ever page number it is with "kews" meaning keywords. I also needed to iterate through the parameter page number to ensure that it kept changing and was not stuck at 0. In the nested-for loop, I created an empty list of keywords called `kews` to house all the values. Afterwards, using the pickle module and the `with` statement, I used the `dump` method to make files using the `file_name` variable creating different files with different names.

The last print statement is to show that the process is being completed as the file is running.

Appending File Content to a List

```
# Appending the words in the file into a large list to then utilize in a DataFrame
for i in nyt_tesla_dir.iterdir():
    with open(i, 'rb') as keyword_file:
        df = pd.read_pickle(keyword_file) # temporary variable to store the list value
        for _ in df:
            list_of_all_words.append(_)

print(list_of_all_words)
```

The `nyt_tesla_dir` variable was defined previously in the whole `final.py` as a directory indicator. Thus, iterating through the directory with the `with` and `open`, I used the pandas method of `read_pickle` to retrieve information from the keyword_file as a list. Thus, `df` was a list that contained all the keywords which I could now iterate through using a second for-loop. The `list_of_all_words` was also defined previously in the whole `final.py` and using the repeater character "_" in the for loop, I appended each word in df to the list of all words variable.

List to DataFrame to CSV Results

```
# Converting list into a Pandas Dataframe to make it easier to use certain calculation methods
# Write to a CSV using Pandas method
calc_dataframe = pd.DataFrame(list_of_all_words)

calc_dataframe = calc_dataframe.rename(columns={0: 'Keyword'})
print(calc_dataframe)

final_results = pd.DataFrame(calc_dataframe["Keyword"].value_counts())

print(final_results)


# This converts into a CSV file which is available on the folder I created previously with the other
binary files or pickle files
final_results.to_csv("/Users/rishpednekar/PycharmProjects/allprojects/2023_summer/NYT_Tesla_keywords/fin
al_results_tesla.csv")
```

Description:

The first line of code here is is creating a pandas DataFrame object out of all the items in the list. Afterwards, I renamed the column "Keyword" and then printed out the dataframe. The real great thing about pandas is the `value_counts()` method which I used for the `final_results` variable which counts the number of instances each word showed up within the list.

Afterwards, I used the pandas `.to_csv()` method to convert the dataframe that had the count information to the folder that had the binary files.

Below, is the visualization of the CSV in Markdown format.

The **Keyword** column shows how many times that Keyword has appeared.

Results & Findings:

	Keyword
Computers and the Internet	15
Twitter	5
Social Media	5
Musk, Elon	5
Google Inc	4
internal-sub-only-nl	4
Start-ups	4
Artificial Intelligence	3
Venture Capital	2
Apple Inc	2
Research	2
Tesla Motors Inc	2
Stocks and Bonds	2
Mergers, Acquisitions and Divestitures	2
Initial Public Offerings	1
United States	1
Innovation	1
Synopsys Inc	1

	Keyword
Shortages	1
Factories and Manufacturing	1
High Net Worth Individuals	1
Computer Chips	1
United States Politics and Government	1
Trump, Donald J	1
Patterson, David A (1947-)	1
Quarantine (Life and Culture)	1
Search Engines	1
Airbnb	1
Silicon Valley (Calif)	1
Automobiles	1
Space Exploration Technologies Corp	1
Alphabet Inc	1
Appointments and Executive Changes	1
Driverless and Semiautonomous Vehicles	1
Trucks and Trucking	1
Aurora Innovation Inc	1
Uber Technologies Inc	1
Waymo	1
Urmson, Chris P (1976-)	1

	Keyword
Traffic Accidents and Safety	1
Hennessy, John L	1
Biden, Joseph R Jr	1
Visas	1
Fringe Groups and Movements	1
Smartphones	1
Advertising and Marketing	1
Layoffs and Job Reductions	1
Science Fiction	1
OpenAI Labs	1
Microsoft Corp	1
Altman, Samuel H	1
Kurzweil, Ray	1
Amazon.com Inc	1
Facebook Inc	1
Boards of Directors	1
Suits and Litigation (Civil)	1
Lindell, Mike	1
Immigration and Emigration	1
West, Kanye	1
Rumors and Misinformation	1

	Keyword
Robots and Robotics	1
Surgery and Surgeons	1
your-feed-science	1
your-feed-health	1
University of California, Berkeley	1
Software	1
Presidential Election of 2020	1
Endorsements	1
Foreign Students (in US)	1
Turing Award	1
Global Warming	1

Describing & Analyzing the Results/Findings:

The most popular keywords were “Computers and the Internet”, "Twitter”, "Social Media”, “Elon Musk”, and "Google Inc”. These keywords emphasize the vast variety of topics and associations surrounding Tesla. They also show the connection between co-founder Elon Musk and Tesla as both Twitter and Elon Musk are tied for the second most popular keywords. The keywords OpenAI Labs and Space Exploration Technologies Corp were also important to our original curiosity about whether Elon Musk related companies and ideas had high counts in correlation to the Tesla search query. Twitter was tied for second most popular while SpaceX and OpenAI only had 1 count each. Only high profile companies like Twitter received mentions along with Tesla. This showed that although there is a relation in Musk related companies and Tesla, it is only a strong relationship if the company is also an industry titan. This shows that Elon Musk is still only a piece and not the entirety of what Tesla is.

After concluding that, high profile Elon Musk related companies and ideas have a generally high keyword count in articles written about Tesla, the next steps would be to find the keyword count on articles about Artificial Intelligence in the Technology section from January 1st, 2020 until current time. Since it was concluded that there is a high correlation between Musk and Tesla, the next defining technological idea which defines Tesla is Artificial Intelligence. Artificial Intelligence is a key aspect of their autonomous driving feature in their Tesla electric cars. It is unique to Tesla as they were the first to make it incredibly popular. This makes it interesting for a future idea to see if Tesla would be a popular keyword for the keyword search of Artificial Intelligence.

Additional Ideas:

In the future working with this dataset, we can experiment running with different queries as the main keyword chanes from Tesla, OpenAI, SpaceX and run it through the script using different methods. Furthermore, to build on this project, using some more Python libaries, we can build histograms to compare each of those queries and compare them. Within this program, we saw the keywords that were done when we ran with “Tesla” as the keyword. Building onto the program, we can make it so that it runs with multiple keywords using different methods and placing things at different times and using methods to create create different binary files for each of the keywords. After then extracting the data from each of the files and using pandas to compare them, we would be able to have multiple CSV files.

In the future as well, we can expand this to create a feature that has a list of keywords that we want to search as well as more of a data visualization aspect to the CSV. Thus, it gives a more robust data analysis aspect to this search query of Tesla and also asking more questions about what exactly is `internal-sub-only-nl` and what role does it play within the New York Times.

Written with StackEdit.