**Case Study**

**Twitter API**

**YouTube video link:**

<https://youtu.be/DGGljYx5qrA>

**What did I try to do?**

As a student in Applied Computer Science, I often hear people ask: “Which language should I learn?”. Often times each person has their own bias, but if there were a way to decide, I think that one should learn a language that is most commonly used as that would be more beneficial for the person. Therefore I decided to put the Twitter API into practice with PowerShell and see which language is most trending. I tried to collect the number of tweets mentioning each of 7 programming languages I specified (C, Python, Java, JavaScript, Go, Swift, PHP) over the span of 1 week. Then I would analyse that data and publish ad tweet stating the result on my twitter account.

**How did I solve it , or how did I configure it?**

To go about making this idea come to life, I did the following:

1. **I requested my Twitter API keys:**
   1. First I needed to make a developers account to login to the Developers Portal.
   2. Then I made a project called project 1 and an app in it called AI News 0.1

**Graphical user interface, application

Description automatically generated**

* 1. I initially got the Essentials Access from Twitter immediately but apparently it turned out that it wasn’t what I needed and that I must apply for Elevated Access. I eventually got it later on. The twitter API version I’m using is v2.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

* 1. So now I have my API Key and Secret, Bearer Token, Access Token and Secret (with Read and Write permissions), and a Twitter development ID.

1. **Using PowerShell to retrieve the data from Twitter:**

I looked up online for modules that I can use to retrieve data from twitter with PowerShell. I found and tried a couple such as mckellerman’s PSTwitterAPI (<https://github.com/mkellerman/PSTwitterAPI#pstwitterapi> ), and thedavecarroll’s BluebirdPS (<https://github.com/thedavecarroll/BluebirdPS> ), both on GitHub.

The first managed to help me be able to retrieve tweets and post them, but I decided to use the second one with BluebirdPS since it had a script that can get the count of tweets mentioning a keyword. This module is on the PowerShell Gallery as well.



I had to use PowerShell 7 either way since PowerShell 5 was not suitable. BluebirdPS is a Twitter automation client for PowerShell 7.

Now, First I had to Authenticate my twitter using the credentials I received. For this I used the script: Set-TwitterAuthentication from the module. Then I would be prompted to enter the API Key, API Secret, Access Token, and Access Token Secret in the command line in PowerShell.

Text

Description automatically generated

Next, I started working on my script that I wrote. Which is as follows:

Install-Module -Name BluebirdPS -Scope CurrentUser

Import-Module BluebirdPS

Get-TweetCount -SearchString '#C' -Granularity "Day" | Export-Csv c.csv

Get-TweetCount -SearchString '#Python' -Granularity "Day" | Export-Csv python.csv

Get-TweetCount -SearchString '#Java' -Granularity "Day" | Export-Csv java.csv

Get-TweetCount -SearchString '#JavaScript' -Granularity "Day" | Export-Csv javascript.csv

Get-TweetCount -SearchString '#Go' -Granularity "Day" | Export-Csv go.csv

Get-TweetCount -SearchString '#Swift' -Granularity "Day" | Export-Csv swift.csv

Get-TweetCount -SearchString '#PHP' -Granularity "Day" | Export-Csv php.csv

$c = (import-csv c.csv) | Select-Object -ExpandProperty Count

$python = (import-csv python.csv) | Select-Object -ExpandProperty Count

$java = (import-csv java.csv) | Select-Object -ExpandProperty Count

$javascript = (import-csv javascript.csv) | Select-Object -ExpandProperty Count

$go = (import-csv go.csv) | Select-Object -ExpandProperty Count

$swift = (import-csv swift.csv) | Select-Object -ExpandProperty Count

$php = (import-csv php.csv) | Select-Object -ExpandProperty Count

"C " + $c + " Python " + $python + " Java " + $java + " JavaScript " + $javascript + " Go " + $go + " Swift " + $swift + " PHP " + $php | Out-File tweet.csv

.\pythonScript.ps1

$trending = Get-Content .\result.txt

# $trending.GetType()

# "$trending".GetType()

Publish-Tweet -TweetText "$trending" -Path .\trendingImage.jpg -Category TweetImage

**Script Explanation:**

1. I installed and imported the module BluebirdPS.

Install-Module -Name BluebirdPS -Scope CurrentUser

Import-Module BluebirdPS

1. Then I used the provided script Get-TweetCount and specified the parameters:

Get-TweetCount -SearchString '#C' -Granularity "Day" | Export-Csv c.csv

-SearchString (mandatory) 🡪 to tell the PowerShell what string we are looking for, in this case the hashtags of each programming language.

-Granularity (optional, by default it is “Hour”) 🡪 to tell PowerShell that we are getting the data in hours or days. In my case I chose days, and it retrieved data for every day in the week for each programming language.

| Export-Csv *filename.csv* 🡪 I used the pipeline to put the collected data into a csv file so I can analyze it later on in Python. I tried using Out-File at first but it messed up the formatting and wasn’t suitable, so I exported instead to get a proper cleaner and neater formatting when I open it in Excel like the following example:

Graphical user interface, application, table, Excel

Description automatically generated

Note that it has 8 entries as the data is collected from, let’s say, Monday to the following Monday, both included. Therefore we have 8 days of data which is around a week. We now have the csv files for each language in the folder I’m working on.

Graphical user interface, application

Description automatically generated

1. Next I made variables for each language to store the entries in the count column of the csv file. First I import the csv file using the command import-csv *filename.csv* then I use a pipeline ‘|’ to put the csv file’s content as the input for the Select-Object command. I use the parameter -ExpandProperty and specify the column’s name (in this case “Count”) which I want its data to be collected and stored in the variable.

$c = (import-csv c.csv) | Select-Object -ExpandProperty Count

1. I needed a way to somehow label each group of counts for each language with the name of that programming language. I realized this when I was writing my Python Script because I needed to know which language was the owner of the highest calculated percentage among the 7 languages, and without a label for each percentage then it would be unknown and thus I lost the aim of the my idea for which language is most trending. So to solve this issue I wrote down the labels in advance and associated them with their corresponding count numbers which were stored in the variables in the previous step. And then I used the pipeline ‘|’ and the command Out-File to put them into a new csv file which I called *tweet.csv*.

"C " + $c + " Python " + $python + " Java " + $java + " JavaScript " + $javascript + " Go " + $go + " Swift " + $swift + " PHP " + $php | Out-File tweet.csv

1. Now it was time to put the python code that I wrote into action. I made a script that contains the command to run the python file in a script and then ran that script in my main script. I tried to just put the command for running the python file in my main script directly but I ran into problems so I had to use a roundabout way.

.\pythonScript.ps1

I will explain the python code I wrote later in this documentation.

1. Now for the final steps towards publishing my tweet. After the python code is run, I get 2 outputs: a file called result.txt, and an image named trendingImage.jpg. These are the 2 things which will be published as a tweet.

The result.txt contains the number 1 trending language and its percentage compared to the other 6 languages. This will be the text of the tweet. As for the image, it is the visualization I decided to go with for the information that was collected and analysed. It is a pie chart that was made using python’s matplotlib library.

$trending = Get-Content .\result.txt

# $trending.GetType()

# “$trending”.GetType()

Publish-Tweet -TweetText "$trending" -Path ..\trendingImage.jpg -Category TweetImage

I used the BluebirdPS module’s script cmdlet Publish-Tweet to tweet on my account. The parameters used were:

-TweetText 🡪 to specify the text of the tweet. In this case it had to be a string, and I couldn’t just put the .\resuly.txt file there. So the variable $tending was mad and I used the command Get-Content to get the content inside the result/txt file and store it in the variable. But even then it did not work, so on the side I checked the type of data that was now stored in the variable $trending using the command GetType() and apparently it ended up being an array. So I added quotations and checked the type again, it was string. Thus, I typed “$trending” for my -TweetText’s argument.

-Path 🡪 to add the path to my media file which I want to upload, in this case it is the trendingImage.jpg pie chart image.

-Category 🡪 the category parameter is to specify the media type, it can only be TweetImage, TweetVideo, or TweetGif. In this case it is TweetImage.

The tweet is now published onto my Twitter account (my username: [@192\_yona](https://twitter.com/192_yona)):

Graphical user interface, chart, application, pie chart

Description automatically generated

**Now Time to explain my Python code:**

import operator

import matplotlib.pyplot as plt

total = 0

languagesList = []

numbersList = []

with open("./tweet.csv") as file:

    line = file.readline()

    line = line.strip()

    while line:

        record = line.split(' ')

        # Adding to the languages list

        languagesList.append(record[0])

        languagesList.append(record[9])

        languagesList.append(record[18])

        languagesList.append(record[27])

        languagesList.append(record[36])

        languagesList.append(record[45])

        languagesList.append(record[54])

        # Adding to the numbers list

        # For C

        numbersList.append(record[1])

        numbersList.append(record[2])

        numbersList.append(record[3])

        numbersList.append(record[4])

        numbersList.append(record[5])

        numbersList.append(record[6])

        numbersList.append(record[7])

        numbersList.append(record[8])

        # For Python

        numbersList.append(record[10])

        numbersList.append(record[11])

        numbersList.append(record[12])

        numbersList.append(record[13])

        numbersList.append(record[14])

        numbersList.append(record[15])

        numbersList.append(record[16])

        numbersList.append(record[17])

        # For Java

        numbersList.append(record[19])

        numbersList.append(record[20])

        numbersList.append(record[21])

        numbersList.append(record[22])

        numbersList.append(record[23])

        numbersList.append(record[24])

        numbersList.append(record[25])

        numbersList.append(record[26])

        # For JavaScript

        numbersList.append(record[28])

        numbersList.append(record[29])

        numbersList.append(record[30])

        numbersList.append(record[31])

        numbersList.append(record[32])

        numbersList.append(record[33])

        numbersList.append(record[34])

        numbersList.append(record[35])

        # For Go

        numbersList.append(record[37])

        numbersList.append(record[38])

        numbersList.append(record[39])

        numbersList.append(record[40])

        numbersList.append(record[41])

        numbersList.append(record[42])

        numbersList.append(record[43])

        numbersList.append(record[44])

        # For Swift

        numbersList.append(record[46])

        numbersList.append(record[47])

        numbersList.append(record[48])

        numbersList.append(record[49])

        numbersList.append(record[50])

        numbersList.append(record[51])

        numbersList.append(record[52])

        numbersList.append(record[53])

        # For PHP

        numbersList.append(record[55])

        numbersList.append(record[56])

        numbersList.append(record[57])

        numbersList.append(record[58])

        numbersList.append(record[59])

        numbersList.append(record[60])

        numbersList.append(record[61])

        numbersList.append(record[62])

        line = file.readline()

for number in numbersList:

    total += int(number)

cTotal = 0

pythonTotal = 0

javaTotal = 0

javaScriptTotal = 0

goTotal = 0

swiftTotal = 0

phpTotal = 0

for c in numbersList[0:8]:

    cTotal += int(c)

for py in numbersList[9:17]:

    pythonTotal += int(py)

for j in numbersList[18:26]:

    javaTotal += int(j)

for js in numbersList[27:35]:

    javaScriptTotal += int(js)

for g in numbersList[36:44]:

    goTotal += int(g)

for s in numbersList[45:53]:

    swiftTotal += int(s)

for ph in numbersList[54:]:

    phpTotal += int(ph)

cPercentage = round((cTotal/total)\*100, 2)

pythonPercentage = round((pythonTotal/total)\*100, 2)

javaPercentage = round((javaTotal/total)\*100, 2)

javaScriptPercentage = round((javaScriptTotal/total)\*100, 2)

goPercentage = round((goTotal/total)\*100, 2)

swiftPercentage = round((swiftTotal/total)\*100, 2)

phpPercentage = round((phpTotal/total)\*100, 2)

percentagesList = [cPercentage, pythonPercentage, javaPercentage, javaScriptPercentage, goPercentage,

                   swiftPercentage, phpPercentage]

languageDictionary = {}

for language in languagesList:

    for percentage in percentagesList:

        languageDictionary[language] = percentage

        percentagesList.remove(percentage)

        break

sorted\_dictionary = sorted(languageDictionary.items(), key=operator.itemgetter(1), reverse=True)

with open('./result.txt', 'w') as file2:

    file2.write("The number 1 trending programming language is: "

                + str(sorted\_dictionary[0][0]) + " (" + str(sorted\_dictionary[0][1]) + "%)")

languages = [str(sorted\_dictionary[0][0]), str(sorted\_dictionary[1][0]), str(sorted\_dictionary[2][0]),

             str(sorted\_dictionary[3][0]), str(sorted\_dictionary[4][0]), str(sorted\_dictionary[5][0]),

             str(sorted\_dictionary[6][0])]

values = [sorted\_dictionary[0][1], sorted\_dictionary[1][1], sorted\_dictionary[2][1], sorted\_dictionary[3][1],

          sorted\_dictionary[4][1], sorted\_dictionary[5][1], sorted\_dictionary[6][1]]

font = {'size': 8}

plt.rc('font', \*\*font)

plt.figure(figsize=(10, 3))  # width = 10, height = 3

plt.pie(values, labels=languages)

plt.suptitle("Programming Languages' Popularity this week")

plt.savefig("trendingImage.jpg")

**Explanation:**

* Imported the necessary libraries to work with dictionaries and to make graphs.
* Created a variable called total to store the sum of all the tweets collected for the calculation process.
* Created 2 lists, one for the languages’ names, and one for the count numbers.
* Opened the file tweet.csv which stores all the collected labelled data. The file is read and the lists are filled up in order. First the languagesList is filled then the numbersList.
* The variable total is also changing as the numbers add up via a for loop.
* Since I have data for the number of tweets each language is in for every day on the span of 8 days, I needed to add them up to calculate and find the most trending in the week. So I created variables for each language’s total.
* The variables are changed as we add up to the using for loops and slicing for the lists to get the right number for each language.
* The percentages for each language are then calculated and put into a list in order with the languagesList.
* The problem I faced now was that I needed to sort the percentagesList without losing track which percentage is for which language, so I made a dictionary called languageDictionary.
* The languages (keys) and the percentages (values) were added to the dictionary.
* Then I made a new sorted dictionary from the first one using the sorted function and operator library.
* The first entry in the newly sorted dictionary is the most trending programming language as the sortation was done descending. It is written to the file result.txt.
* Then the pie chart is plotted with the languages and values from the dictionary, and then saved as the trendingImage.jpg image.

**The PowerShell scripts used from the module are Get-TweetCount and Publish-Tweet:**

**Get-TweeCount.ps1:**

Text

Description automatically generated

Text

Description automatically generated

**Publish-Tweet.ps1:**

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**Sources Used:**

1. (MAIN ONE): The GitHub repository for the BluebirdPS: <https://github.com/thedavecarroll/BluebirdPS>
2. The GitHub repository for the PSTwitterAPI: <https://github.com/mkellerman/PSTwitterAPI#pstwitterapi>
3. The Tools and Libraries mention on the Developer Platform of Twitter: <https://developer.twitter.com/en/docs/twitter-api/tools-and-libraries/v2>

**(Extra) For context, this is my folder structure:**

Table

Description automatically generated

**In Brief to make things clear:**

* + **How to retrieve information via an API?**

Information was retrieved from the Twitter API via PowerShell.

* + **Document the code, cmdlets or tools that you use**

Mentioned above.

* + **What information do you filter?**

The number of times each of 7 programming languages occurred in tweets in the form of hashtags in a week. I filtered it into percentages.

* + **How do you visualise the information and what code do you need for this?**

I visualized the information in a pie chart image, and a published tweet containing text and that image. I used Python code to make the pie chart, and PowerShell to tweet.