Covid Update Discord Bot

> Using Twitter API v2 and Discord API

💡 Idea & Plan

In Australia, the number of COVID-19 cases have been spiking up a lot recently. I wish to know the number of new cases every day without having to go on Twitter and being distracted by everything else. What I visualised in my head was: I want to create a discord bot that keeps track of the tweets from a certain list of users (e.g. NSWHealth, SAHealth, ACTHealth, etc.), and if the tweet is the daily update, it will post the tweet URL onto the discord channel.

And so the project was split into 2 parts:

1. Access to Twitter API to get the real-time update on tweets
2. Create a Discord bot that send tweet updates and host it

🐦 Twitter API

I spent the very last week of 2021 to play around with different APIs, from Spotify, to Youtube, and finally Twitter. And the first thing I need for any kind of API access is a set of tokens and Keys.

Tokens and Keys

First of all, I created a Twitter Developer account, obtained my Consumer Keys and Authentication Tokens, and copied it into my Python files. There are a number of ways to do this.

> If it's a quick retrieval and I want to run the script on my computer, I would save the tokens and keys in a separate file called config.py and import it into my main file.
>
> However, with the Discord bot hosted on the server (Part 4), all of my tokens and keys are saved as environment variables using the OS module, which can then be accessed as follows:

```
import os

BEARER_TOKEN = os.environ['BEARER_TOKEN']
CONSUMER_KEY = os.environ['CONSUMER_KEY']
CONSUMER_SECRET = os.environ['CONSUMER_SECRET']
ACCESS_TOKEN = os.environ['ACCESS_TOKEN']
ACCESS_TOKEN_SECRET = os.environ['ACCESS_TOKEN_SECRET']
```

Twitter API v2

My Twitter Developer account only has access to Twitter API v2, which is very much enough for my current use. I chose to use the Tweepy Python library as it allows me to take advantage of the pre-defined functions to access tweets information. I then create a client with my keys and tokens as follows

```
import tweepy

tweet_client = tweepy.Client(bearer_token=BEARER_TOKEN,
                             consumer_key=CONSUMER_KEY,
                             consumer_secret=CONSUMER_SECRET,
                             access_token=ACCESS_TOKEN,
                             access_token_secret=ACCESS_TOKEN_SECRET)
```

With this `tweet_client`, I can already do a number of things such as

- liking a tweet: Copy the tweet ID into your script and run the following code

  ```
  # tweet URL: https://twitter.com/NSWHealth/status/1477097282453610501
  tweet_id = 1477097282453610501
  tweet_client.like(tweet_id)
  ```

- retweet:

  ```
  tweet_client.retweet(tweet_id)
  ```

- get tweets within a specified time: For example, the following code will give me all the tweets that have been posted within the past 8 hours by @NSWHealth. According to Tweepy documentations, the `start_time` parameter is the earliest UTC timestamp from which the Tweets will be provided. Note that the timestamp:
  - Has to be in UTC
  - Follows the ISO 8601/RFC 3339 format (e.g. YYYY-MM-DDTHH:mm:ssZ) , and
  - Does not support a millisecond value.

  I can specify the max_results to be minimum 5 and maximum 100. Using Pagination, the most I can get is the 3200 most recent tweets. This method returns data, includes, errors, and meta. If the user I'm interested in has posted something within the last 8 hours, the Tweet objects will be returned within `data`. If not, data will be of type None.

  ```
  now = datetime.datetime.utcnow()
  start_time = now - datetime.timedelta(hours=24)
  start_time = start_time.isoformat(timespec='seconds') + 'Z'

  user_id = 40778270 # twitter ID of @NSWHealth
  response = tweet_client.get_users_tweets(id=user_id, exclude=['retweets'],
                                           max_results=100, start_time=start_time)
  data, includes, errors, meta = response
  ```

```
(DataEngineering) Thaos-MacBook-Pro:twitter_api3 thaopham$ python3 check.py
Response(data=[<Tweet id=1477519966354087936 text=@Uy_Alroy You can only leave self-isolation after 7 days if you do not have a sore throat, ru
nny nose, cough or shortness of breath. If you have symptoms in the 24 hours before your 7 days is finished, please call the NSW Care at Home S
upport Line on 1800 960 933 for further advice.>, <Tweet id=1477489839822016514 text=As we ring in the new year, consider making some healthy c
hanges to give your mind and body a boost.

Small steps to be more active, eat more vegetables and adopt healthy sleep habits will go a long way to improve your health and wellbeing.

More: https://t.co/1vTBHuyLGr https://t.co/D26tK3F1h2>, <Tweet id=1477459640954888204 text=Going away over the Summer break?

Pack medication, scripts and your medical history.
If you become unwell, get tested and self-isolate until you get a negative result
Check testing requirements for where you are travelling to https://t.co/5Gr2n53Lok>], includes={}, errors=[], meta={'oldest_id': '1477459640954
888204', 'newest_id': '1477519966354087936', 'result_count': 3})
```

I can now access each Tweet object in the data list using a for loop fairly easily.

```
for item in data:
    print(f'{item.id}')
    print(f'{item.text}')
    print()
```

```
(DataEngineering) Thaos-MacBook-Pro:twitter_api3 thaopham$ python3 check.py
1477519966354087936
@Uy_Alroy You can only leave self-isolation after 7 days if you do not have a sore throat, runny nose, cough or shortness of breath. If you have symptoms in the 24 hours befor
e your 7 days is finished, please call the NSW Care at Home Support Line on 1800 960 933 for further advice.

1477489839822016514
As we ring in the new year, consider making some healthy changes to give your mind and body a boost.

Small steps to be more active, eat more vegetables and adopt healthy sleep habits will go a long way to improve your health and wellbeing.

More: https://t.co/1vTBHuyLGr https://t.co/D26tK3F1h2

1477459640954888204
Going away over the Summer break?

Pack medication, scripts and your medical history.
If you become unwell, get tested and self-isolate until you get a negative result
Check testing requirements for where you are travelling to https://t.co/5Gr2n53Lok
```

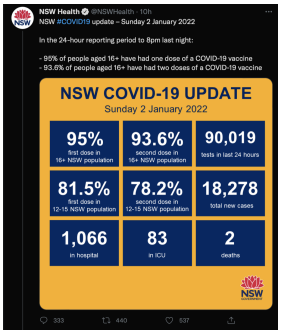
Filter tweets

Now to filter out only the tweets that I want, I decided on the simplest way possible. For example, NSWHealth always have the keywords "COVID-19 update" or VicGovDH always have the hashtag "#COVID19VicData" in their daily update. A simple check can help me filter out the tweets that I need.

```
keywords_for_nsw = id_dict['NSWHealth'][1]

for tweet in data:
    text = tweet.text.lower()

    contain_keywords = True
    for word in keywords:
        if word not in text:
            contain_keywords = False # if one of the keywords is not found, immediately break
            break
    if contain_keyword:
        print(f'{tweet.id}')
        print(f'{tweet.text}')
```

This is the result tweet that I retrieved, which is the correct tweet that I was hoping for.

```
(DataEngineering) Thaos-MacBook-Pro:twitter_api3 thaopham$ python3 check.py
1477399412380868611
NSW #COVID19 update - Sunday 2 January 2022

In the 24-hour reporting period to 8pm last night:

- 95% of people aged 16+ have had one dose of a COVID-19 vaccine
- 93.6% of people aged 16+ have had two doses of a COVID-19 vaccine https://t.co/Rdrsv5lGq1
```



Putting everything into functions

```
def process_tweet(tweet, username, keywords):
    text = tweet.text.lower()
    contain_keyword = True
    for word in keywords:
        if word not in text:
            contain_keyword = False
            break
    if contain_keyword:
        tweet_url = f'http://twitter.com/{username}/status/{tweet.id}'  # create url
        return tweet_url
    return


def get_tweets_from_user(username, user_id, start_time, keywords):
    data, includes, errors, meta = tweet_client.get_users_tweets(id=user_id, exclude=['retweets'],
                                                                 max_results=100, start_time=start_time)
    if data is None:
        return []
    results = []
    for tweet in data:
        tweet_url = process_tweet(tweet, username, keywords)
        if tweet_url:
            results.append(tweet_url)
    return results
```

To get tweets from multiple users:

```
def get_tweets(data, minute_delta):
    now = datetime.datetime.utcnow()
    start_time = now - datetime.timedelta(minutes=minute_delta)
    start_time = start_time.isoformat(timespec='seconds') + 'Z'

    combined_url_list = []

    username_list = [elem for elem in data if elem not in ['CHANNEL_ID', 'MINUTE_INTERVAL']]
    for username in username_list:
        user_id, keywords = data[username]
        url_list = get_tweets_from_user(username, user_id, start_time, keywords)
        combined_url_list = combined_url_list + url_list

    return combined_url_list
```

🤖 Discord Bot

Create a Bot user

To create my very first Discord bot, I went to Discord Developer Portal, create a new Application, and call it my Baby Bot. I then invited the bot to my server and generated its token. Similar to above, I saved the BOT_TOKEN into my config file or os environment. Then I import the commands and tasks frameworks from the discord.ext module to leverage some of the predefined methods. I set up my simple Baby Bot as follows:

```
from discord.ext import commands, tasks

bot = commands.Bot(command_prefix='!')

@bot.event
async def on_ready():
    print(f'We have logged in as {bot.user}')

bot.run(BOT_TOKEN)
```

Loop a task

Finally, I can create a loop to repeat a task every specified interval. My bot will check for recent tweets every 5 minutes, if any of the new tweets contains the keywords in them, the tweet URL will be posted into the discord channel. Note that the channel ID must be specified.

To get the Channel ID: Go to the Discord App → User Settings → Advanced → Turn on Developer Mode. After that, go to the server that your bot is currently in, right click on the channel that you want → Copy ID at the bottom.

```
MINUTE_INTERVAL = 5

@tasks.loop(minutes=MINUTE_INTERVAL)  # repeat after every defined interval
async def send_update():
    await bot.wait_until_ready()
    channel = bot.get_channel(CHANNEL_ID)  # replace with channel ID that you want to send to
    most_recent_tweets = get_tweets(data, MINUTE_INTERVAL)

    if len(most_recent_tweets) > 0:
        for each_msg in most_recent_tweets:
            await channel.send(each_msg)

send_update.start()
bot.run(BOT_TOKEN)
```

Host my Bot on a server

I followed the instructions provided here to host my bot for free in the cloud by migrating all of my codes onto Replit and set up a monitor on UptimeRobot to ping my Replit workspace every 5 minutes. This way I can keep my Discord bot running 24/7.

🧪 Testing

To test my bot, I included my own Twitter ID as one of the accounts to follow, with keywords to be 'new' and 'cases'. I set the loop interval to be 1 minute.

https://youtu.be/EjF4kaP34y8

The code has worked as I wanted!

🚥 Workflow

Set up

I decided on a fairly simple way to set up things in terms of saving usernames, user_ids and keywords. I created a `setup.py` script where I defined a dictionary of the parameters including:

1. the channel ID on Discord where I want my bot to post
2. the minute interval for the loop, and
3. the accounts that I want my bot to update me on with the keywords.

You can change these parameters to that of your own.

```
input_dict = {"CHANNEL_ID": 926733136883236937, # change this to your channel id
              "MINUTE_INTERVAL": 2,
              "NSWHealth": ["covid", "update"],
              "VicGovDH": ["#covid19data"],
              "SAHealth": ["covid", "update"],
              "ACTHealth": ["new", "cases"],
              "qldhealth": ["new", "cases"]}
```

While I can use this `input_dict` directly, I decided to add another step. That is, for every username provided, I call the `tweet_client.get_user()` function and add the user_id into a new dictionary. I then save this new dictionary into a JSON file. This is because the `tweet_client.get_user_tweets()` method requires user_id as a unique identifier and I do not want to get user_id again and again every time my bot runs the task as it will add up to my monthly quota very quickly. The JSON file looks like this:

```
{"CHANNEL_ID": 926733136883236937,
"MINUTE_INTERVAL": 2,
"NSWHealth": [40778270, ["covid", "update"]],
"VicGovDH": [43064490, ["#covid19data"]],
"SAHealth": [369808606, ["covid", "update"]],
"ACTHealth": [346251068, ["new", "cases"]],
"qldhealth": [35997368, ["new", "cases"]]}
```

After that, I can run the `main.py` script and get my bot up and running.

Overall, my code works as follows:

1. Retrieve user_id(s) of the provided username(s) and save into a JSON file.
2. Create a discord bot that runs a background task loop.
3. Retrieve the most recent tweets that have been posted by username(s) within the last specified minute interval.
4. Filter the tweets so that only the ones containing the keywords will be kept.
5. Post the tweet url onto the discord channel.

🎬 Conclusion

Waking up to see my Baby Bot working as I wanted to was a very surreal feeling. A very fun project to end my 2021 and start my 2022 - lots of exploration.

- I learned how to use Twitter API.
- I learned a lot throughout the process of reading through the documentations on Twitter Developer Portal and Tweepy. I learned how to read the documentations, understand what is expected to come in, how to format the parameters to get exactly what I want, and what is expected to come out, how to read the results.
- I learned about os environment.
- I familiarised myself more with datetime objects.
- I learned how to work with json objects/files.
- I learned how to create a Discord Bot, and how to host it using Replit and UptimeRobot. During the process, I learned a little bit about AWS EC2 and how to host the bot on a virtual machine but I have not successfully made it work.
- I learned how to organise my code with the concept of orthogonality in mind.
- I learned about using `if __name__ == "__main__"` , `def main()` and fstring.

👣 Potential next steps

- An easier way for users to set up: Create a command on Discord so that users can set username and keywords and that will be added into the database.
    1. Set a new account for the bot to follow. If username already in database, this will overwrite.

       `$set username keyword1 keyword2 keyword3`
    2. Delete an account complete from the list. If username not in database, this will fail:

       `$del username`

3. Add a new keyword on top of what's already in the database. If username is not in database, this will fail.

   `$add_kw username new_keyword`

4. Remove a keyword from what's already in the database. If username is not in database, this will fail.

   `$remove_kw username keyword_to_remove`

- Create a database that has different states as columns and rows as the case numbers of a certain day. Then I can add commands for lookup. For example, if a user is interested in case number in Victoria of a certain date, they can input something like `$inquire vic 24/10/2021`.

🔗 Resources

Twitter API

Twitter API V2 Programming with Python and Tweepy (Skolo Online Learning)

Discord Bot

Code a Discord Bot with Python - Host for Free in the Cloud (freeCodeCamp.org)

How to Make Discord Bot Commands in Python (Eric Chi)

How to Loop a Task in Discord.py (StackOverflow)

Hosting a Python Discord Bot using AWS & Redis (Evan Deubner)