# maxTweetedWord

## How the bot works

The bot scans through its followers and for every follower, it scans the user's tweets and tweet their most tweeted word in the format: @username's most tweeted word is [word] (number of times word is used)

To find their most tweeted word, a Twitter user will have to follow this bot's account.

As of now, a user following the bot will not act as a trigger. I will manually be running the bot from my machine a few times a day to work it.

### Logic

Getting a user from followers list:

- first, I open a blank .txt file
- next, using the api.followers(user) function, I pulled the bot's followers
  scan through the followers and filter out the protected accounts -- add the selected user IDs to a followers_list
- cycle through the followers_list formed in the above step with a for loop
  IF "follower" in the for loop iteration is not in the .txt file, append users_list with "follower" and also append .txt file
- the above step makes sure the bot does not run for the same account more than once
  choose only 5 users at a time.

Finding max tweeted word:

- This is for chosen user among 5 users in users_list
- Pull all tweets of the user. Twitter API limits maximum tweets to be pulled at 200. To bypass this and pull all (up to ~3,500) tweets, cycle through tweets by filtering by using tweet_id which is timestamped
- store all tweets in a list
- initialize a words_list and a words_count_list to store the words tweeted by the user and corresponding count
- in a for loop, for every tweet, pull tweet.text to extract the text content mentioned in each element of the tweets list from the step above
- Once tweet.text is extracted, comb through each word with a for loop. For word selected in for loop, convert it to small-case to avoid multiple counting. For example, YES, Yes and yes are all considered to be different words if it is not case insensitive. Check if word selected in for loop iteration exists in words_list. If YES, then find index of

word in words_list and increment the element at index in words_count. If NO, check if chosen word in the loop is NOT in the words_to_filter.txt file. If word is not in words_to_filter.txt, append word in words_list and then do index matching as mentioned in the YES step

- After this, sort the words_count_list in reverse order -- element at pos [0] is the count of the max tweeted word. Find index of element at [0] in sorted list in words_count_list and the word at this index is the most tweeted word

Tweeting:

- using api.update_status, tweet in the format @username's most tweeted word is [word] (number of times word is used)

## Story

It was April 2021, and we had gone under a second lockdown because of the new COVID-19 surge, and as I was surfing through Twitter, I came across a bunch of bot accounts that retweeted tweets that had a specific hashtag or specific words (like beds, oxygen, covid, etc) which helped people in need of the stuff get their requirements by expanding their reach.
I figured after my 12th grade I would want to do similar stuff.

So one day while taking a shower I thought of building a bot that finds someone's most tweeted word, like Spotify Wrapped. Seemed like a pretty cool idea.

After my 12th I started learning Python. I watched videos to get a better understanding of the syntaxes involved because the concepts (at the beginning) are the same as any other programming language -- data types, loops, iterables, etc.
After getting comfortable with the syntaxes, I contacted one of my college seniors and asked him what he would suggest I do, with regards to how I could expand my knowledge and expertise in python. He told me that a bunch of his friends did challenges from a website called HackerRank.
So that afternoon, I sat at my laptop and opened the website. The way HackerRank works is the website has a fixed set of challenges (a LOT of challenges), and they have a ranking system. You solve challenges → you get points → you move up the ranks.
I started out at rank ~1.5 million and kept solving challenges and moved up to 49k.
This helped me because -

- It made me more comfortable with syntaxes I already knew
- It taught me new syntaxes and functions which I didn't know before
- It taught me how to think about solving a problem or a particular type of problem and how to approach problems

Moving from rank 1,512,354 to rank 49,674 took me about 3-4 weeks if I remember correctly.

So after doing this, I felt I was equipped with the knowledge and expertise to work with Twitter and its API.
I registered on Twitter's Developer Portal and got my API Keys and Tokens, only to find out that I had no idea what to do with them.

So I did some research (or at least, I tried to) and got nowhere.
A few days later, I came across a course on Coursera, called Using Python to Access Web Data, offered by the University of Michigan. I thought this was interesting so I registered for it and completed the course.
The course *really* helped me because it actually taught me how a program gets data and information from the web server, how the whole request-response cycle works, what APIs are, how they work, and a lot more.

After this, I decided I will take one more shot at trying to use Twitter API. I sat down at my laptop at 8 pm, and within 30 minutes I tweeted my first tweet from my Python console on my laptop. It was pretty easy and quick. I just needed to understand how it all worked.

That night, I sat down with my laptop at 10:30 pm, and in 30-45 minutes, I had written the basic algorithm for the bot (which I have uploaded on GitHub).

Then I played around with the code and tried to find out the most tweeted words of celebrities and friends for an hour.

That was fun, but it actually helped me realize that I was going to need to do some *filtering* to remove very commonly used words, words that are essential for English language grammar, and just plain boring words like "to".
By the time I realized this it was 1 am, so I decided to work on it the next day.

The next day, I sat down and made a .txt file consisting of words I felt should be filtered or excluded. They can be found on twitter.com/maxTweetedWord.

After a few test runs, I realized that all this time when I was testing the code, I was going through only the 20 most recent tweets of a certain user. So I needed to figure out how to pull more than that.
After a bit of thinking and snooping on StackExchange and geeksforgeeks, I figured out how to go through more tweets. I ran my code, and as expected, it took longer to execute, however it was going through ~3,500 of the user's most recent tweets.

After this, the only thing I needed to do was change the way the bot was choosing a user. When I was testing it, I was manually typing in the username of the target user. My goal was to choose a user from the list of accounts that follow the bot.
So I started figuring out how to do that. A day or two later, I figured it out and integrated it into the final code.

And so, in 7 days, I built my bot.

Then I just had to get API credentials for @maxTweetedWord on Twitter.

A day after finishing the bot, on October 31, 2021, I launched the bot on Twitter.
On that day it got 150 followers within 5 hours, which felt like a lot
But then the next day, within 30 hours of launching the bot, the account got 800+ followers.
The next day, 1,500 followers. And it kept growing, and I could not keep up because I had to run this bot on my personal laptop, and it took ~1 minute per user.
At its peak, the bot had 2,789 followers.

It was really overwhelming, and while it felt good that I made something like this, it was also slightly stressful because I knew I could not keep up.
So 10 days later, on November 10, I suspended the bot indefinitely, and I am trying to figure out how to run it more efficiently in the background or on the cloud somehow.

## What I learned

This bot was a *huge* learning experience for me because it was my first “project” of sorts.
First, I learned how to work with APIs, and what APIs are.
Second, I learned how to work with certain data and how to salvage data from real people.
Third, I learned that I need to learn infrastructure management and learn how not to be dependent on my personal device for executing my code.
Fourth, after launching the bot, I realized that more words and characters needed to be filtered, which I added to the words_to_filter file.
Fifth, I learned that working on projects like these is the best way to grow and get better at something – no amount of problem-solving will be a substitute for this.

## Scope for improvement

- The data analysis can be done better and faster – more optimized.
- One dictionary can be used instead of two index-matched lists.

## Moving forward

I have a few plans moving forward,

- Convert this project into a web application hosted on the cloud. This will help me learn cloud computing and infrastructure and also help me learn web development and connecting my frontend to my backend