

# HOW TO TRACK TWEETS BY GEOGRAPHIC LOCATION



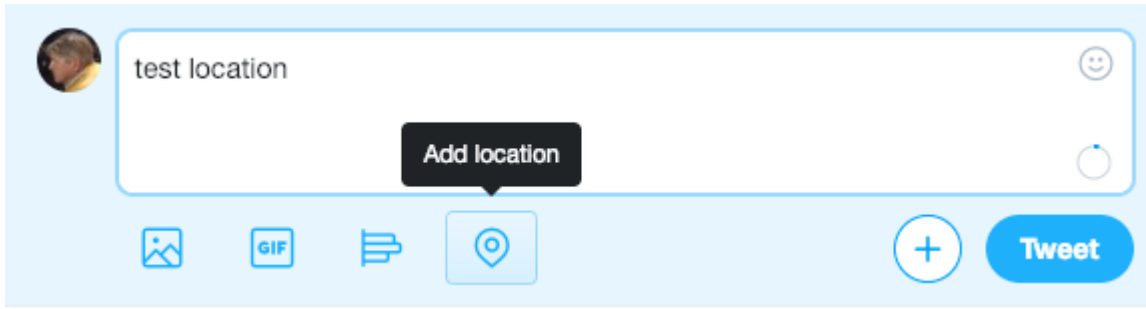
Here we explain how to track users and tweets by their location. As we will see, there are some limits. (For more background on GEOJson, i.e., the universal standard to designate geographic location, you can read what we wrote about using locations in MongoDB [here](#).)

## Twitter Users Must Opt into Location Tracking

You can track Tweets by location. But there is a limit to that, as Twitter users must opt into location tracking or manually add the location to tweets.

By one purely anecdotal estimate, somewhere between 1% of Twitter users have set up precise location tracking and 10% have set up tracking to a wider area, such as a city. This means it might not be really useful for marketing, customer support, or other kinds of research. But if you want to pursue this further, you could study [this academic paper](#) and write an ML algorithm to improve user tracking accuracy to make your application be better at doing that.

To get started, here is what a Tweet looks like from a browser. You can click the **Add Location** button and it will insert what Twitter calls a **Place**, which in GEOJson is called a **Polygon**. So that will be a general area, like a city, unless the user has also enable Precise Location Tracking.



In the Twitter iPhone

and Android app you can enable a Precise Location Tracking like this:

## ← Precise location

### Precise location

If enabled, Twitter will collect, store, and use your device's precise location, such as GPS information. This lets Twitter improve your experience — for example, showing you more local content, ads, and recommendations.

Location permission for Twitter is disabled. Enable Location permission for Twitter in [App Info](#).

This lets Twitter collect, store, and use your device's precise location, such as GPS information, in order to improve your experience — for example, showing you more local content, ads, and recommendations.

NOT NOW      OK

This is what a Point (Precise Location) looks

like in a Tweet.

```
'geo': {
  'type': 'Point',
  'coordinates':
}
```

And here is a general location, like a city, called a **Place**, defined by a rectangle, which in GEOJson is called a **Polygon**, which, of course, does not match the mathematical description of that. The Twitter Polygon are 4 line segments, which make a square.



it's not long. If you use those two arguments you can turn on your stream and it can run for as long as you want without being shut down by Twitter.

```
stream = Stream(auth=auth,  
listener=ListenerChild(api=None,producer=producer),wait_on_rate_limit=True,wa  
it_on_rate_limit_notify=True)
```

Here are the rest of the code excerpts all together. We put `l == 1` in `Listener` because we need some instruction there or Python will complain about an indentation error.

```
class Listener(StreamListener):
```

```
    l == 1;
```

```
class ListenerChild(Listener):
```

```
    def __init__(self,api,producer):  
        self.producer=producer  
        super().__init__(api)
```

```
    def on_data(self, data):
```

```
        j = json.loads(data)
```

```
        try:
```

```
            if j is not None:
```

```
                tt = parseTweet(j)
```

```
                logging.info(tt)
```

```
                logging.info (tt)
```

```
                self.producer.send('tweets', bytearray(tt,'utf-8'))
```

```
        except KeyError:
```

```
            logging.info ("rate limited" + date.today().strftime('%Y-%m-%d  
%H:%M:%S'))
```

```
producer = KafkaProducer(bootstrap_servers='localhost:9092')
```

```
auth = OAuthHandler(consumer_key, consumer_secret)
```

```
auth.set_access_token(access_token, access_token_secret)
```

```
stream = Stream(auth=auth,
```

```
listener=ListenerChild(api=None,producer=producer),wait_on_rate_limit=True,wa  
it_on_rate_limit_notify=True)
```

```
stream.filter(locations=region,languages=languages,track=track)
```