BDA Assignment 3

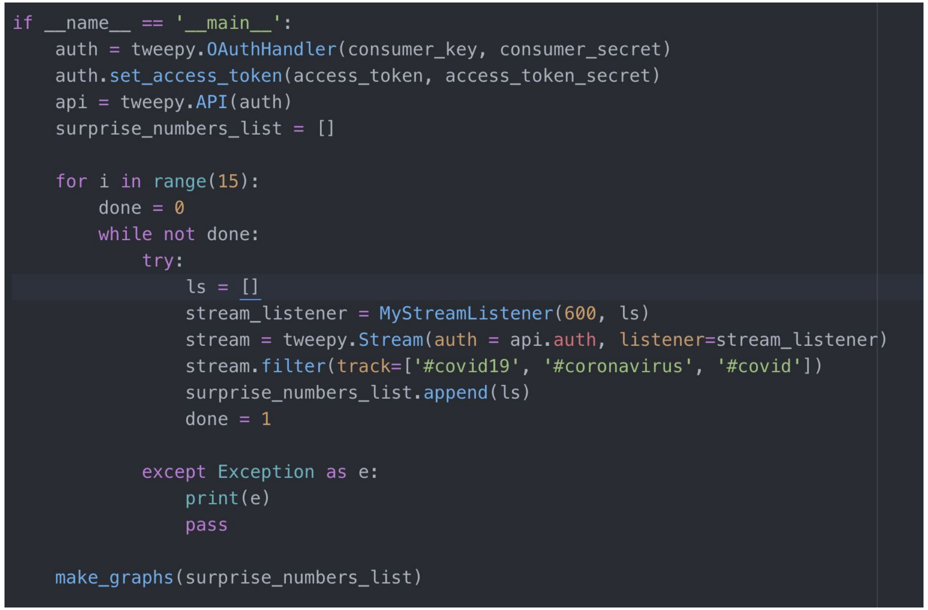
Ahilya Sinha (2016009)
Anushika Verma (2016015)

The process

- Our code used can be found in the attached streaming.py
- We used tweepy to stream data from the twitter API
- We crawled for data over 3 different hashtags - '#covid19', '#coronavirus' and '#covid'
- The AMS Algorithm had to be implemented from scratch as tweepy does not have that in it’s library.
- Over the next few slides, we go over the different functions used.
- In the end, we present our results.

Overview of the code (ie, main function)

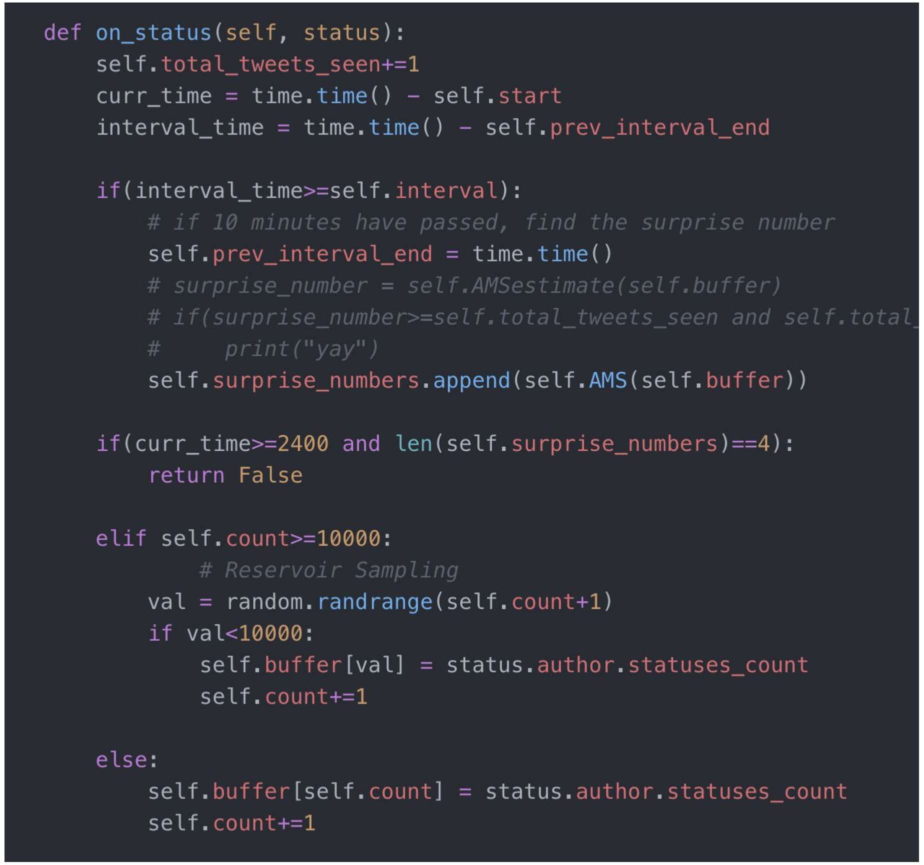
- Initially, we create a OAuthHandler to connect to the API and collect the data needed.
- A list of surprise numbers is created to be used for plotting the graphs later.
- Crawl tweets 15 times, each time for 40 minutes each.
- The stream listener checks at every 600s (ie, 10 minutes) and calculates the surprise number for the reservoir.



```python
if __name__ == '__main__':
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    api = tweepy.API(auth)
    surprise_numbers_list = []

    for i in range(15):
        done = 0
        while not done:
            try:
                ls = []
                stream_listener = MyStreamListener(600, ls)
                stream = tweepy.Stream(auth = api.auth, listener=stream_listener)
                stream.filter(track=['#covid19', '#coronavirus', '#covid'])
                surprise_numbers_list.append(ls)
                done = 1

            except Exception as e:
                print(e)
                pass

    make_graphs(surprise_numbers_list)
```

Reservoir and AMS (ie, Handling each new tweet)

- Keep track of time since stream was started, and time since the last surprise number calculation.
- If 10 minutes have passed - update previous interval time and add the new surprise number to the stream's list.
- If 40 minutes have passed - end the stream
- If there are greater than 10,000 elements in the reservoir, add new one based on reservoir sampling
- If there is still space in the reservoir, then add the element.
- The element is the number of tweets made by that user in total.

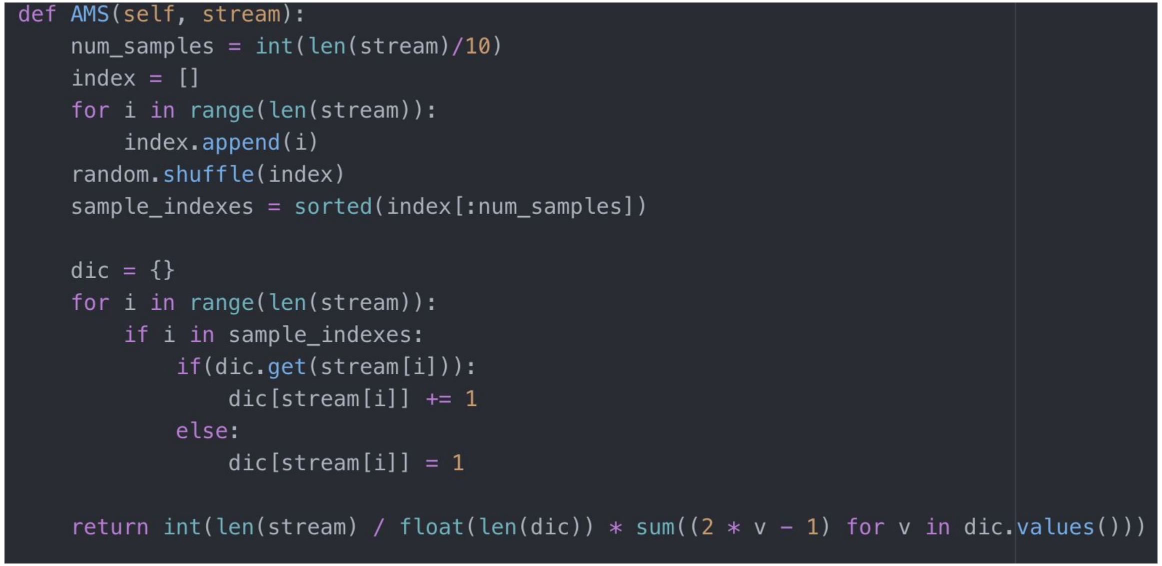


```python
def on_status(self, status):
    self.total_tweets_seen+=1
    curr_time = time.time() - self.start
    interval_time = time.time() - self.prev_interval_end

    if(interval_time>=self.interval):
        # if 10 minutes have passed, find the surprise number
        self.prev_interval_end = time.time()
        # surprise_number = self.AMSestimate(self.buffer)
        # if(surprise_number>=self.total_tweets_seen and self.total_
        #     print("yay")
        self.surprise_numbers.append(self.AMS(self.buffer))

    if(curr_time>=2400 and len(self.surprise_numbers)==4):
        return False

    elif self.count>=10000:
            # Reservoir Sampling
        val = random.randrange(self.count+1)
        if val<10000:
            self.buffer[val] = status.author.statuses_count
            self.count+=1

    else:
        self.buffer[self.count] = status.author.statuses_count
        self.count+=1
```

Surprise Number Calculation

- We take the number of samples to be length of the stream/10 (ie, 1000)
- We then pick 1000 random indices.
- Then, the count for the numbers at the indices is stored.

1. Finally, we calculate the
   a. X.value for each random
   b. Index and then
   c. Return the average of
   d. The values.



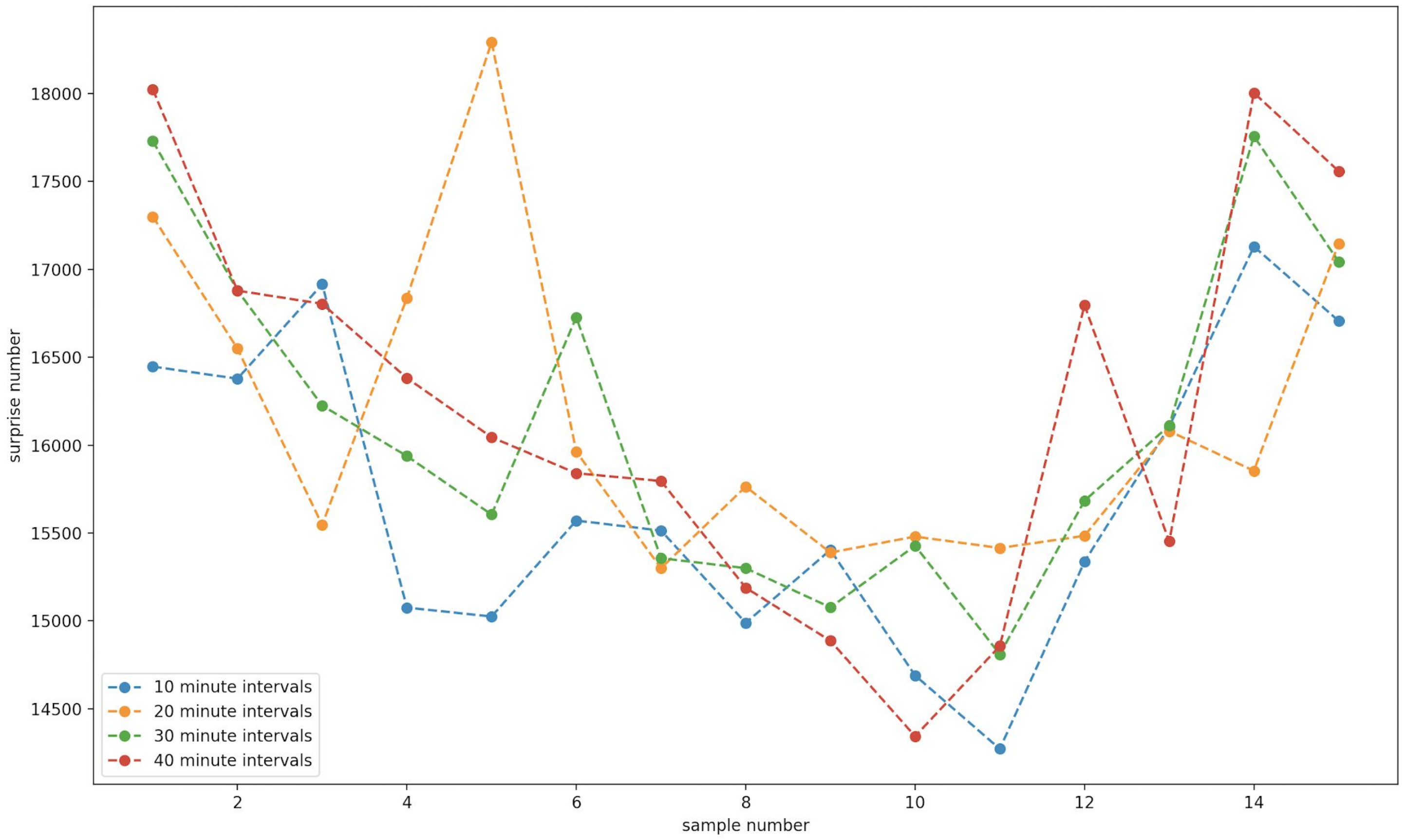
```
def AMS(self, stream):
    num_samples = int(len(stream)/10)
    index = []
    for i in range(len(stream)):
        index.append(i)
    random.shuffle(index)
    sample_indexes = sorted(index[:num_samples])

    dic = {}
    for i in range(len(stream)):
        if i in sample_indexes:
            if(dic.get(stream[i])):
                dic[stream[i]] += 1
            else:
                dic[stream[i]] = 1

    return int(len(stream) / float(len(dic)) * sum((2 * v - 1) for v in dic.values()))
```

Problems Faced

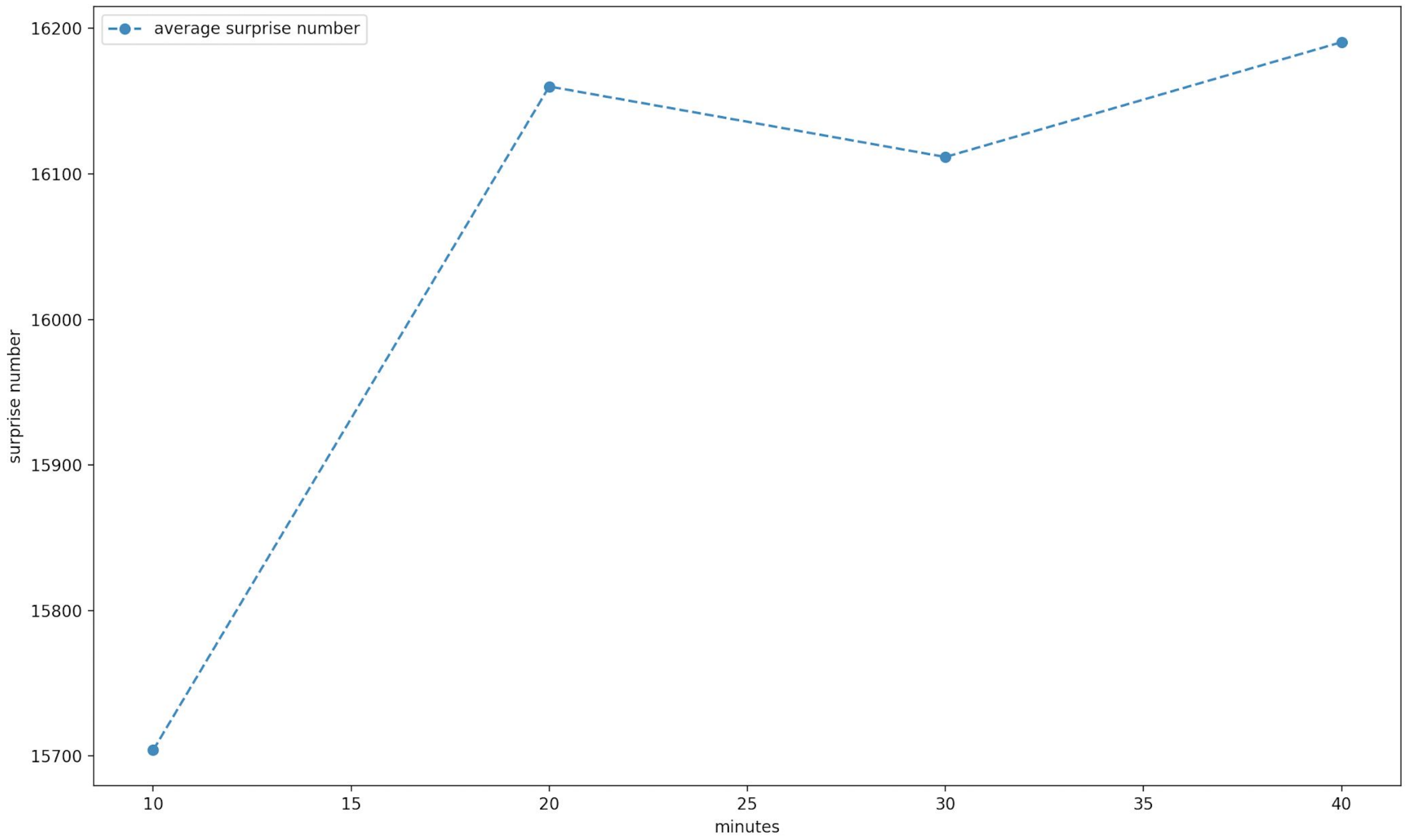
- We initially tried to use Apache Storm but were unable to install it and get it working correctly, so used Tweepy as an alternative.
- Twitter rate limits the number of streams you can run at the same time. Initially, we tried running 4 different streams at the same time - one each for the different time intervals. This was problematic
- Due to this we shifted over to using one single stream and checking and recalculating the surprise number every 10 minutes.
- The time taken to collect 15 samples for each time interval was very, very long. The stream would often disconnect and everything would have to be started over. Due to this, we added in the try catch block around the stream to retry so the code could continue running in the background

Results

- The entire assignment took 5 days to complete. 1 day to try and install Apache Storm, 2 days to write the code in Tweepy, and 2 days to collect all the data (after frequent disconnections).
- The following slides contain our graphs



Surprise number for each time interval for each sample



Average of the surprise numbers for each time interval

Resources

- Tweepy documentation
- Mmds for better understanding of the algorithms
- Python Documentation