Working with streaming data: Mining Twitter Data with Python and MongoDB

Scott Herford email: jherford@mail.smu.edu
Nuoya Rezsonya email: nrezsonya@mail.smu.edu
Lu Cheng email: lucheng@mail.smu.edu

***Abstract*— Twitter is the live internet. It is a social network like Facebook but more about news, current social events, and comments. Data from Twitter reflects what people are seeing, thinking and saying. Extraction of the data using methods such as filtering, aggregation, and various analysis techniques can allow a way to harness the full value of the data helping in the generation of valuable insights. The objective of this project is to investigate stream processing technology operating on social media streams. The project report will entail building a sample service that accepts data from a Twitter stream and stores it locally in a database. The project report will also discuss the different design considerations such as real-time vs. batch processing, performance, and scalability.**

***Index Terms*—Twitter REST API, Streaming filter API, MongoDB, Python (packages: pymongo, json, twitter, pprint)**

I. INTRODUCTION

Twitter can generate a social graph based on the data from its users. It reveals what's happening, what people are seeing, and what kind of comments people are making. Due to the 280-character limitation, it reveals the real thoughts of how people think about current events. The way people read their news has changed from main stream media to social media platforms like Twitter. Twitter generates data continuously by its users. Around 6,000 tweets are tweeted on Twitter every second on average. The data can be in various formats such as, text, image, video etc. These are called streaming data.

Streaming data needs to be processed sequentially and incrementally on a record-by-record basis or over sliding time windows. It is used for a wide variety of analytics including correlations, aggregations, filtering, and sampling. Filtering, aggregating, analyzing such data can allow a way to harness the full value of the data, extracting valuable information.

There are two major methods of streaming data from Twitter: Streaming API and REST API. Streaming API is real-time, and it will give records from the point of query until we stop, and the later one is retrospective which will give us the records from the past to as far as the search index.

In this project report, we will do the following:

- ➢ Building a sample service.

  The sample service can accept data from a Twitter stream and stores locally in database. This part includes:

1. Streaming tweets using Twitter REST API
   - Establish the connection with Twitter.
   - Establish the connection with MongoDB database.
   - Define searching parameters in REST API.
   - Extract twitter data with the defined searching parameters and fetching data from twitter into MongoDB.
2. Streaming tweets using Twitter Streaming API
   - Establish the connection with Twitter.

- Establish the connection with MongoDB database.
- Set up the query definition in the Streaming API.
- Streaming process and store the collected data into local MongoDB.

➢ Comparing the design considerations: Real-time processing vs. Batch processing.
Real-time vs. batch processing in terms of processing and performance and scalability.

II. Building A sample service: Streaming API and REST API

There are two methods of collecting data using Twitter API: Streaming API and REST API. We will provide both of them in order to illustrate the process of collecting data using different approaches. The goal of this sample service is to accept data from a Twitter stream and stores locally in database. This sample service includes streaming tweets using Twitter REST API and Streaming API, filtering out irrelevant data by using keywords and store to local database. There is a list of parameters we can use to filter and collect specific data (Table 1 Searching Parameters).

Parameter	Definition
q	A UTF-8, URL-encoded search query of 500 characters maximum, including operators. Queries may additionally be limited by complexity.
Count	The number of tweets to return per page, up to a maximum of 100. Defaults to 15.
Geocode/locations	Returns tweets by users located within a given radius of the given latitude/longitude.
delimited	This parameter may be used on all streaming endpoints, unless explicitly noted. Setting this to the string length indicates that statuses should be delimited in the stream.
stall_warnings	Setting this parameter to the string true will cause periodic messages to be delivered if the client is in danger of being disconnected.
language	Setting this parameter to a comma-separated list of BCP 47 language identifiers corresponding to any of the languages listed on Twitter’s advanced search page will only return Tweets that have been detected as being written in the specified languages.
follow	A comma-separated list of user IDs, indicating the users whose Tweets should be delivered on the stream.
track	A comma-separated list of phrases which will be used to determine what Tweets will be delivered on the stream.
with (deprecated)	The with parameter controls the types of messages delivered to User and Site Streams clients. • The default for Site Streams is with=user, which only streams messages from the user associated with the stream. • The default for User Streams is with=followings which adds messages from accounts the user follows, equivalent to the user’s home timeline.
replies (deprecated)	By default, @replies are only sent if the current user follows both the sender and receiver of the reply.
stringify_friend_ids (deprecated)	By default, user and site streams send the Friends List preamble as an array of integers (equivalent to stringify_friend_ids=false).

Table 1 Searching Parameters

In order to get access to data on Twitter, we first have to grant API key, API secret and also access token, access token secret. Streaming data from Twitter is based on the keys, tokens and secrets generated from Twitter since OAuth is the only way to access Twitter API now. Then in the followings, we will use those tokens and secrets in Python to stream data and later use MongoDB to do a simple query on them.

We need to set up connection to database as well. Database system we are using for this project is MongoDB. We use functions from python package pymongo to create MongoDB client connected to service. We also create and maintain database and collections for data storage and analysis.

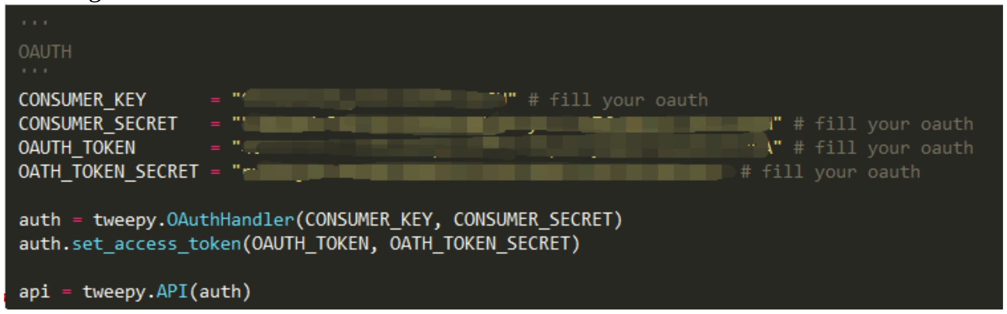
The python packages we have engaged in this step including: pymongo, json, twitter, tweepy and pprint.

➢ Streaming tweets using Twitter Streaming API.

In this sample service, we are using tweepy package in Python to use Streaming API. The Twitter streaming API is used to download twitter messages in real time. It is useful for obtaining a high volume of tweets. The request/response is the main difference between Streaming API and REST API. In Streaming API, the server continues sending response to client whenever an update is available.

- Establish the connection with Twitter.

We are using a package – tweepy in python to establish the connection. Tweepy supports accessing Twitter via OAuth which is the current only way to access Twitter API. This package provides a Python interface with Twitter API. See Figure 1 Code of Connection with Twitter.

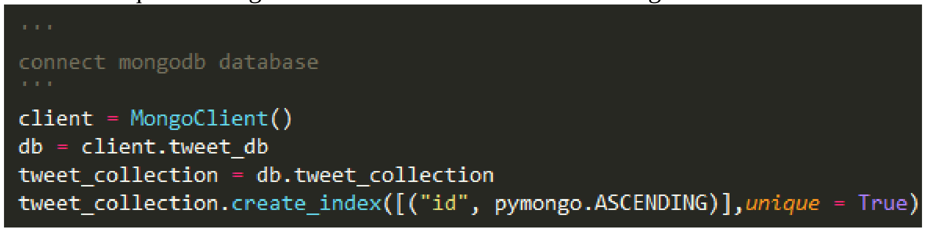


```
'''
OAUTH
'''
CONSUMER_KEY      = "[redacted]" # fill your oauth
CONSUMER_SECRET   = "[redacted]" # fill your oauth
OAUTH_TOKEN       = "[redacted]" # fill your oauth
OATH_TOKEN_SECRET = "[redacted]" # fill your oauth

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(OAUTH_TOKEN, OATH_TOKEN_SECRET)

api = tweepy.API(auth)
```

Figure 1 Code of Connection with Twitter

- Establish the connection with MongoDB database.

We are using MongoClient from pymongo library to set up the connection with the MongoDB database so that we can store collected tweets in MongoDB for future use. The last line of this chunk code is to make sure that the collected tweets are unique. See Figure 2 Code of Connection with MongoDB.

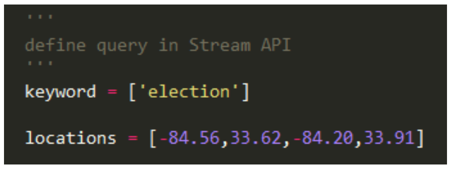


```
'''
connect mongodb database
'''
client = MongoClient()
db = client.tweet_db
tweet_collection = db.tweet_collection
tweet_collection.create_index([("id", pymongo.ASCENDING)],unique = True)
```

Figure 2 Code of Connection with MongoDB

- Set up the query definition in the Streaming API.

This step will set up the query definition for StreamListener process later. In this sample service, we used two arguments: keywords and locations. The word, election, has been used as the keyword in the streaming process in order to monitor all tweets that have this keyword. And all other words can be used as keyword to subset tweets contain the keyword. The locations factor has been provided here and the factor has longitude, latitude pairs specifying a set of bounding boxes to filter tweets. The location pairs can be any locations and any tweets falling within the requested bounding boxes will be collected. Query definition can help to keep the only related data. Figure 3 Code of Query Definition.

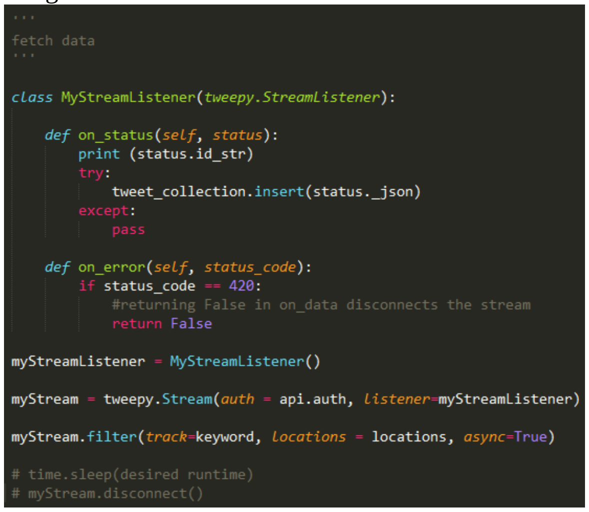


```
'''
define query in Stream API
'''
keyword = ['election']

locations = [-84.56,33.62,-84.20,33.91]
```

Figure 3 Code of Query Definition

- Streaming process and store the collected data into local MongoDB.

In the tweepy package, tweepy.Stream is the instance to establish a streaming session and routes messages to StreamListener. A stream listener inserts messages under the on_status method. If there is an error message reporting the rate limit has been reached, the on_error method will stop the streaming. Then later on, we use the query definition to filter out election related data to store in the database. In this section, it contains three steps to use the Streaming API, there are three steps: Creating a class from StreamListener, Using the class to create a Stream object and Connect to the Twitter API using the stream. There is also another way to stop streaming after a certain time. One can use time.sleep(desired runtime) and myStream.disconnect() together to halt the control and stop the process. See Figure 4 Code of Streaming Process.

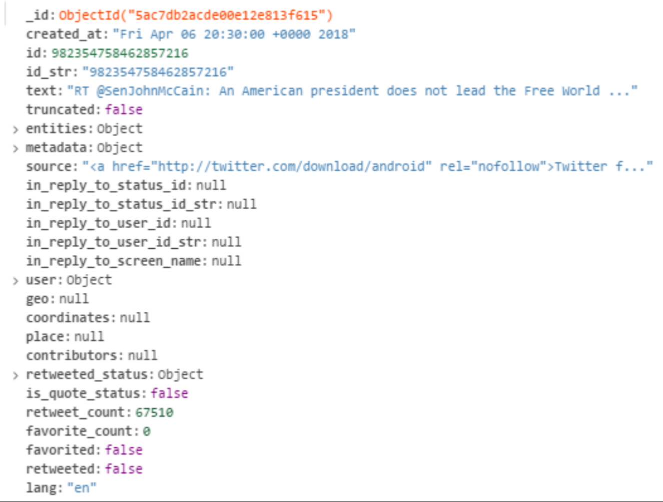


```python
'''
fetch data
'''

class MyStreamListener(tweepy.StreamListener):

    def on_status(self, status):
        print (status.id_str)
        try:
            tweet_collection.insert(status._json)
        except:
            pass

    def on_error(self, status_code):
        if status_code == 420:
            #returning False in on_data disconnects the stream
            return False

myStreamListener = MyStreamListener()

myStream = tweepy.Stream(auth = api.auth, listener=myStreamListener)

myStream.filter(track=keyword, locations = locations, async=True)

# time.sleep(desired runtime)
# myStream.disconnect()
```

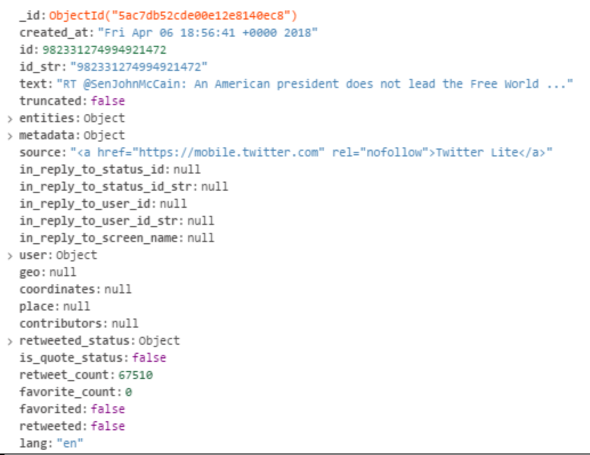
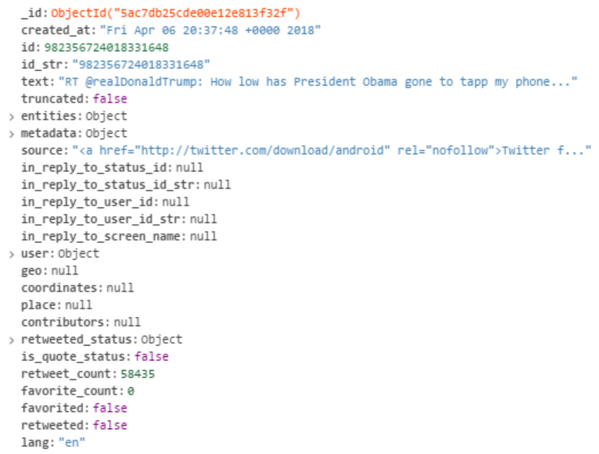
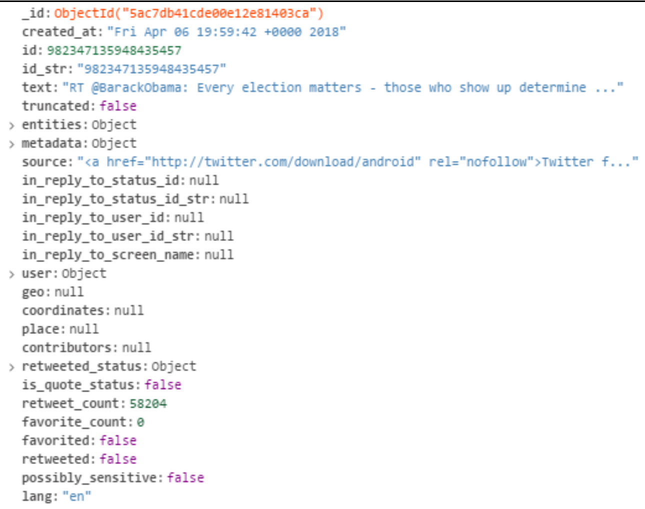
Figure 4 Code of Streaming Process

- The top five retweets

Now we can use MongoDB to find the top 5 retweets in this collected data. In order to be intuitive, we will attach figures from MongoDB Compass.

Tweets	Counts	Details
Tweets from John McCain: An American president does not lead the Free World by congratulating dictators on winning sham elections. And by doing so with Vladimir Putin, President Trump insulted every Russian citizen who was denied the right to vote in a free and fair election.	67510	

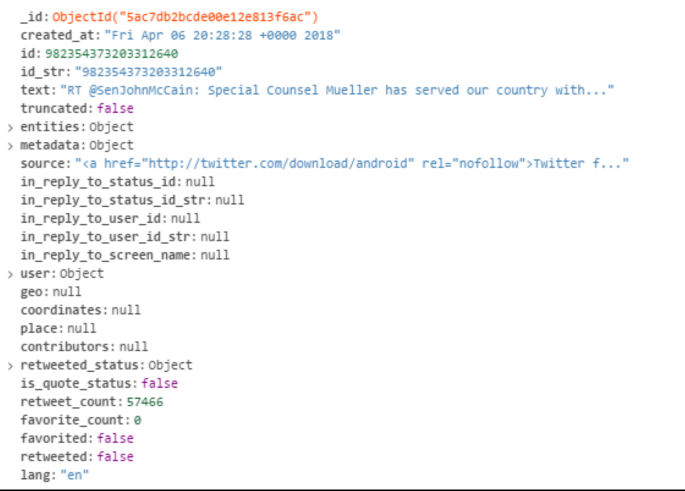
```
_id: ObjectId("5ac7db2acde00e12e813f615")
created_at: "Fri Apr 06 20:30:00 +0000 2018"
id: 982354758462857216
id_str: "982354758462857216"
text: "RT @SenJohnMcCain: An American president does not lead the Free World ..."
truncated: false
› entities: Object
› metadata: Object
source: "<a href="http://twitter.com/download/android" rel="nofollow">Twitter f..."
in_reply_to_status_id: null
in_reply_to_status_id_str: null
in_reply_to_user_id: null
in_reply_to_user_id_str: null
in_reply_to_screen_name: null
› user: Object
geo: null
coordinates: null
place: null
contributors: null
› retweeted_status: Object
is_quote_status: false
retweet_count: 67510
favorite_count: 0
favorited: false
retweeted: false
lang: "en"
```

Tweets from Sen. John McCain which is the same as above: An American president does not lead the Free World by congratulating dictators on winning sham elections. And by doing so with Vladimir Putin, President Trump insulted every Russian citizen who was denied the right to vote in a free and fair election.	67510	
Tweets from President Trump: How low has President Obama gone to tapp my phones during the very sacred election process. This is Nixon/Watergate. Bad (or sick) guy!	58435	
Tweets from Barack Obama: Every election matters - those who show up determine our future. Go vote tomorrow!	58204	

```
  _id: ObjectId("5ac7db52cde00e12e8140ec8")
  created_at: "Fri Apr 06 18:56:41 +0000 2018"
  id: 982331274994921472
  id_str: "982331274994921472"
  text: "RT @SenJohnMcCain: An American president does not lead the Free World ..."
  truncated: false
> entities: Object
> metadata: Object
  source: "<a href="https://mobile.twitter.com" rel="nofollow">Twitter Lite</a>"
  in_reply_to_status_id: null
  in_reply_to_status_id_str: null
  in_reply_to_user_id: null
  in_reply_to_user_id_str: null
  in_reply_to_screen_name: null
> user: Object
  geo: null
  coordinates: null
  place: null
  contributors: null
> retweeted_status: Object
  is_quote_status: false
  retweet_count: 67510
  favorite_count: 0
  favorited: false
  retweeted: false
  lang: "en"
```

```
  _id: ObjectId("5ac7db25cde00e12e813f32f")
  created_at: "Fri Apr 06 20:37:48 +0000 2018"
  id: 982356724018331648
  id_str: "982356724018331648"
  text: "RT @realDonaldTrump: How low has President Obama gone to tapp my phone..."
  truncated: false
> entities: Object
> metadata: Object
  source: "<a href="http://twitter.com/download/android" rel="nofollow">Twitter f..."
  in_reply_to_status_id: null
  in_reply_to_status_id_str: null
  in_reply_to_user_id: null
  in_reply_to_user_id_str: null
  in_reply_to_screen_name: null
> user: Object
  geo: null
  coordinates: null
  place: null
  contributors: null
> retweeted_status: Object
  is_quote_status: false
  retweet_count: 58435
  favorite_count: 0
  favorited: false
  retweeted: false
  lang: "en"
```

```
  _id: ObjectId("5ac7db41cde00e12e81403ca")
  created_at: "Fri Apr 06 19:59:42 +0000 2018"
  id: 982347135948435457
  id_str: "982347135948435457"
  text: "RT @BarackObama: Every election matters - those who show up determine ..."
  truncated: false
> entities: Object
> metadata: Object
  source: "<a href="http://twitter.com/download/android" rel="nofollow">Twitter f..."
  in_reply_to_status_id: null
  in_reply_to_status_id_str: null
  in_reply_to_user_id: null
  in_reply_to_user_id_str: null
  in_reply_to_screen_name: null
> user: Object
  geo: null
  coordinates: null
  place: null
  contributors: null
> retweeted_status: Object
  is_quote_status: false
  retweet_count: 58204
  favorite_count: 0
  favorited: false
  retweeted: false
  possibly_sensitive: false
  lang: "en"
```

Tweets from Sen. John McCain: Special Counsel Mueller has served our country with honesty and integrity. It's critical he be allowed to complete a thorough investigation into Russia's interference in the 2016 election — unimpeded.	57466	

```
  _id: ObjectId("5ac7db2bcde00e12e813f6ac")
  created_at: "Fri Apr 06 20:28:28 +0000 2018"
  id: 982354373203312640
  id_str: "982354373203312640"
  text: "RT @SenJohnMcCain: Special Counsel Mueller has served our country with..."
  truncated: false
> entities: Object
> metadata: Object
  source: "<a href="http://twitter.com/download/android" rel="nofollow">Twitter f..."
  in_reply_to_status_id: null
  in_reply_to_status_id_str: null
  in_reply_to_user_id: null
  in_reply_to_user_id_str: null
  in_reply_to_screen_name: null
> user: Object
  geo: null
  coordinates: null
  place: null
  contributors: null
> retweeted_status: Object
  is_quote_status: false
  retweet_count: 57466
  favorite_count: 0
  favorited: false
  retweeted: false
  lang: "en"
```

➢ Streaming tweets with the REST API

Python Twitter is a library provides a pure Python interface for the Twitter API <https://dev.twitter.com/>. It works with Python versions from 2.7+ and Python 3. The library provides a Python wrapper around the Twitter API and the Twitter data model. It utilizes models to represent various data structures returned by Twitter.

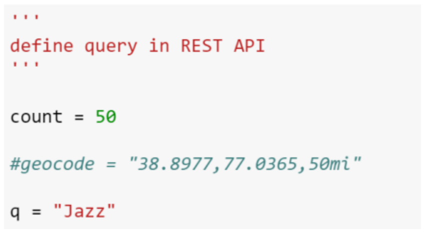
- Establish the connection with Twitter. (same procedure as streaming process)

Build up connection by define object as Twitter class. The python-twitter requires the use of OAuth keys for nearly all operations.

- Establish the connection with MongoDB database. (same procedure as streaming process)

Build up connection by defined object as mongo client class. The client object is thread-safe and has connection-pooling built in.

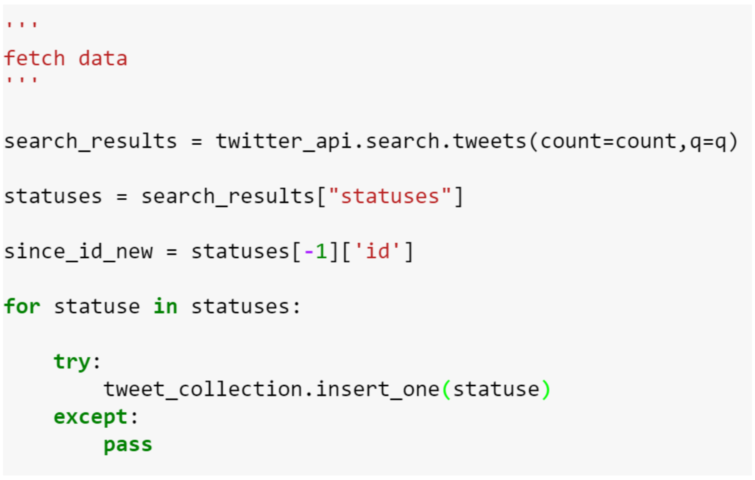
- Define searching parameters in REST API. See Figure 5 Query Definition of REST API.



```
'''
define query in REST API
'''

count = 50

#geocode = "38.8977,77.0365,50mi"

q = "Jazz"
```

Figure 5 Query Definition of REST API

- Extract twitter data with the defined searching parameters and fetching data from twitter, returns results as json and storage data into MongoDB. See Figure 6 Streaming Process of REST API.



```
'''
fetch data
'''

search_results = twitter_api.search.tweets(count=count,q=q)

statuses = search_results["statuses"]

since_id_new = statuses[-1]['id']

for statuse in statuses:

    try:
        tweet_collection.insert_one(statuse)
    except:
        pass
```

Figure 6 Streaming Process of REST API

Twitter_api.search.tweets(Twitter): Twitter Standard search API is one of the public set of API, it searches against a sampling of recent Tweets published in the past 7 days.

Statuses(Twitter): Twitter GET statuses lookup returns fully-hydrated Tweet objects for up to 100 Tweets per request, as specified by comma-separated values passed to the id parameter.

Tweet_collection.insert_one(pymongo): insert a document into a collection.

III. Comparing the design considerations: Real-time Processing and Batch Processing

Upon determining the necessary requirements, such as we as users want the functionality to stream Twitter data, so that we can analyze tweets using specific parameters or keywords; we enter the design phase where consideration of certain factors for accomplishing this goal becomes critical. Considerations must be made with regard to the appropriate tech stack or architecture as well as coming to a consensus around how we intend extract the Twitter data and what methods we intend to implement. As it relates to the following project, we considered whether or not to design a real-time data processing or a batch data processing.

In comparing the two design considerations, we'll concentrate on batch data processing first. Today, most organizations utilize batch data processes which include collection, input, processing, and finally output of the data. During the collection process, groups or chunks of large volumes of data are collected where the data is contained within a specific period of time. In conducting batch data processing, separate programs are needed for input, data processing, and output. The advantages of this approach given an organization possesses the appropriate platforms to conduct this particular method are that management/analytics of big data is possible and efficient.

If we choose to consider designing/developing a streaming API for streaming Twitter data, we can leverage real-time data processing which involves continuous input, processing, and output of resulting data. Using real-time data processing, Twitter data processing is completed in small time periods or (near real time) periods. We can leverage this method of streaming Twitter data in order to take immediate steps/actions where time in the form of minutes and/or seconds are critical for decision-making. Not leveraging this method can prove fatal in some instances, for example, if a large retailer's POS systems which process data in real-time fail during a major holiday period, this situation could have a significant impact on revenue since the company is not able to derive actionable insights and take the appropriate steps needed to help drive customer behavior in a different direction.

Some use cases for real-time data processing are the following:

- Pricing and Analytics
- Ecommerce Transactions/Website Traffic
- Network Monitoring

- Intelligence & Surveillance Operations
- Transaction Cost Analysis
- Customer Services
- Bank ATMs

With both methods comes obvious benefits and challenges. One major benefit that stems from batch data processing is the simplicity where removing batches of data can add value to ROIs. Overall, the ability to conduct analytics from streaming methods is valuable with regard to scalability and flexibility regardless of whether or not the data is collected in real-time or batch (historical). One major challenge and a common problem associated with both methods for streaming is management of such a large volume of data. Management of big data becomes costly when resources are not available or legacy in nature to help with storage and overall performance or speed. The requirements needed for processing real-time data are different than batch data such that the demand for high data throughput is extremely large.

The following are just a few solutions to solve when attempting to implement stream processing methods:

- Processing massive amounts of streaming events (filter, aggregate, rule, automate, predict, act, monitor, alert)
- Real-time responsiveness to changing market conditions
- Performance and scalability as data volumes increase in size and complexity
- Rapid integration with existing infrastructure and data sources: input (e.g. market data, user inputs, files, historical data from a data warehouse) and output (e.g. trades, email alerts, dashboards, automated reactions)
- Fast time-to-market for application development due to quickly changing landscape and requirements
- Developer productivity throughout all stages of the application development lifecycle by offering good tool support and agile development
- Analytics: live data discovery and monitoring, continuous query processing, automated alerts and reactions
- Community

IV. Conclusion

Batch processing as a typical data analysis workflow includes retrieving stored data, loading into analysis tool and exploring the data. Batch processing is used in a variety of scenarios, from simple data transformations to a more complete ETL (extract-transform-load) pipeline. In a big data context, batch processing may operate over very large data sets, where the computation takes significant time. (For example, see Lambda architecture.) A very simple example is transforming a large set of flat, semi-structured CSV or JSON files into a schematized and structured format that is ready for further querying. In most organizations, batch is still a popular way of handling large amount of data.

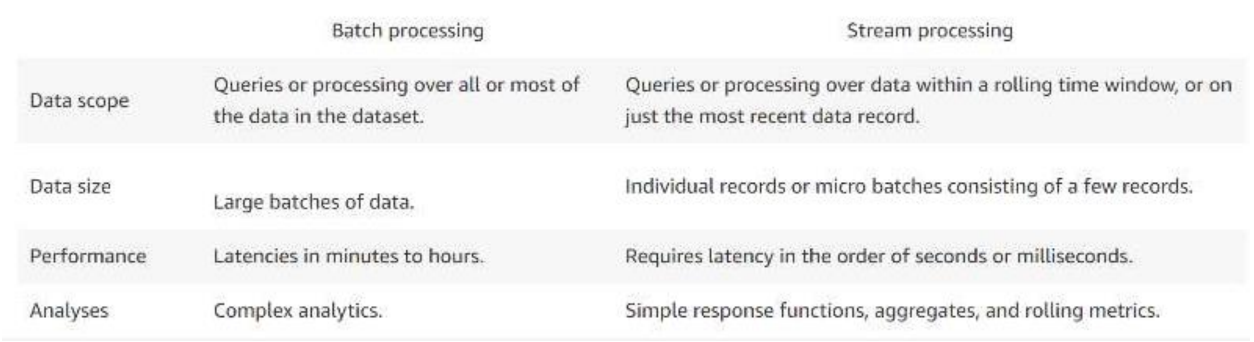
Real-time processing deals with streams of data that are collected in real time and then processed with a minimal latency to generate real-time reports to automated responses. It is often used when one wants analytical results in real time. For example, fraud detection. Another very simple example in daily life is real-time traffic light monitor systems which to make the traffic light systems to be coordinated in real time to deal with changing traffic patterns.

The nature of the data sources is playing an important role in defining whether the data suited for batch or real-time. To utilize which processing method should also follow the purpose of the study. One processing method is not necessarily better than the other one. E-commerce platforms usually utilize both of these processing methods. They need real-time processing during holiday season to monitor the purchase behavior while batch processing method during the regular times. In our case, if the purpose of the study is to know how people react to the election before and after. We can use both processing methods:

- Batch processing: After the election to see based on what kind of criterions people making their decisions.

➢ Real-time processing: To check people's reaction on comments or proposals made by any candidates

The summary of comparison of these two processing methods is as below:



	Batch processing	Stream processing
Data scope	Queries or processing over all or most of the data in the dataset.	Queries or processing over data within a rolling time window, or on just the most recent data record.
Data size	Large batches of data.	Individual records or micro batches consisting of a few records.
Performance	Latencies in minutes to hours.	Requires latency in the order of seconds or milliseconds.
Analyses	Complex analytics.	Simple response functions, aggregates, and rolling metrics.

Table 2 Comparison of two processing methods

V. Appendix

Tweet Data Dictionary

Attribute	Type	Description
created_at	String	UTC time when this Tweet was created
id	Int64	The integer representation of the unique identifier for this Tweet. This number is greater than 53 bits and some programming languages may have difficulty/silent defects in interpreting it. Using a signed 64 bit integer for storing this identifier is safe. Use id_str for fetching the identifier to stay on the safe side.
id_str	String	The string representation of the unique identifier for this Tweet. Implementations should use this rather than the large integer in id.
text	String	The actual UTF-8 text of the status update. See twitter-text for details on what characters are currently considered valid.
source	String	Utility used to post the Tweet, as an HTML-formatted string. Tweets from the Twitter website have a source value of web.
truncated	Boolean	Indicates whether the value of the text parameter was truncated, for example, as a result of a retweet exceeding the original Tweet text length limit of 140 characters. Truncated text will end in ellipsis, like this ... Since Twitter now rejects long Tweets vs truncating them, the large majority of Tweets will have this set to false . Note that while native retweets may have their toplevel text property shortened, the original text will be available under the retweeted_status object and the truncated parameter will be set to the value of the original status (in most cases, false).
in_reply_to_status_id	Int64	Nullable. If the represented Tweet is a reply, this field will contain the integer representation of the original Tweet's ID.
in_reply_to_status_id_str	String	Nullable. If the represented Tweet is a reply, this field will contain the string representation of the original Tweet's ID.

in_reply_to_user_id	Int64	Nullable. If the represented Tweet is a reply, this field will contain the integer representation of the original Tweet's author ID. This will not necessarily always be the user directly mentioned in the Tweet.
in_reply_to_user_id_str	String	Nullable. If the represented Tweet is a reply, this field will contain the string representation of the original Tweet's author ID. This will not necessarily always be the user directly mentioned in the Tweet.
in_reply_to_screen_name	String	Nullable. If the represented Tweet is a reply, this field will contain the screen name of the original Tweet's author.
user	User object	
coordinates	Coordinates	
place	Places	Nullable When present, indicates that the tweet is associated (but not necessarily originating from) a Place .
quoted_status_id	Int64	This field only surfaces when the Tweet is a quote Tweet. This field contains the integer value Tweet ID of the quoted Tweet.
quoted_status_id_str	String	This field only surfaces when the Tweet is a quote Tweet. This is the string representation Tweet ID of the quoted Tweet.
is_quote_status	Boolean	Indicates whether this is a Quoted Tweet.
quoted_status	Tweet	This field only surfaces when the Tweet is a quote Tweet. This attribute contains the Tweet object of the original Tweet that was quoted.
retweeted_status	Tweet	Users can amplify the broadcast of Tweets authored by other users by retweeting . Retweets can be distinguished from typical Tweets by the existence of a retweeted_status attribute. This attribute contains a representation of the original Tweet that was retweeted. Note that retweets of retweets do not show representations of the intermediary retweet, but only the original Tweet. (Users can also unretweet a retweet they created by deleting their retweet.)
quote_count	Integer	Nullable. Indicates approximately how many times this Tweet has been quoted by Twitter users.
reply_count	Int	Number of times this Tweet has been replied to.
retweet_count	Int	Number of times this Tweet has been retweeted.
favorite_count	Integer	Nullable. Indicates approximately how many times this Tweet has been liked by Twitter users. Example: "favorite_count":1138
entities	Entities	Entities which have been parsed out of the text of the Tweet. Additionally see Entities in Twitter Objects .
extended_entities	Extended Entities	When between one and four native photos or one video or one animated GIF are in Tweet, contains an array 'media' metadata. Additionally see Entities in Twitter Objects .
favorited	Boolean	Nullable. Indicates whether this Tweet has been liked by the authenticating user.
retweeted	Boolean	Indicates whether this Tweet has been Retweeted by the authenticating user
possibly_sensitive	Boolean	Nullable. This field only surfaces when a Tweet contains a link. The meaning of the field doesn't pertain to the Tweet content itself, but instead it is an indicator that the URL contained in the Tweet may contain content or media identified as sensitive content.

filter_level	String	Indicates the maximum value of the filter_level parameter which may be used and still stream this Tweet. So a value of medium will be streamed on none, low, and medium streams.
lang	String	Nullable. When present, indicates a BCP 47 language identifier corresponding to the machine-detected language of the Tweet text, or und if no language could be detected. See more documentation HERE.
matching_rules	Array of Rule Objects	Present in filtered products such as Twitter Search and PowerTrack. Provides the id and tag associated with the rule that matched the Tweet. With PowerTrack, more than one rule can match a Tweet. See more documentation HERE.

1) Additional Tweet attributes

Twitter APIs that provide Tweets (e.g. the GET statuses/lookup endpoint) may include these additional Tweet attributes:

Attribute	Type	Description
current_user_retweet	Object	Perspectival Only surfaces on methods supporting the include_my_retweet parameter, when set to true. Details the Tweet ID of the user's own retweet (if existent) of this Tweet. Example: "current_user_retweet": { "id": 26815871309, "id_str": "26815871309" }
scopes	Object	A set of key-value pairs indicating the intended contextual delivery of the containing Tweet. Currently used by Twitter's Promoted Products. Example: "scopes":{"followers":false}
withheld_copyright	Boolean	When present and set to "true", it indicates that this piece of content has been withheld due to a DMCA complaint .
withheld_in_countries	Array of String	When present, indicates a list of uppercase two-letter country codes this content is withheld from. Twitter supports the following non-country values for this field: "XX" - Content is withheld in all countries "XY" - Content is withheld due to a DMCA request.
withheld_scope	String	When present, indicates whether the content being withheld is the "status" or a "user." Example: "withheld_scope": "status"

VI. References

[1] https://www.karambelkar.info/2015/01/how-to-use-twitters-search-rest-api-most-effectively./
[2] https://agsft.com/blog/challenges-real-time-data-vs-batch-data/
[3] https://www.datasciencecentral.com/profiles/blogs/batch-vs-real-time-data-processing
[4] https://www.linkedin.com/pulse/challenges-real-time-data-vs-batch-amit-saxena/
[5] https://www.infoq.com/articles/stream-processing-hadoop)
[6] https://www.youtube.com/watch?v=pUUxmvvl2FE
[7] http://docs.tweepy.org/en/v3.4.0/streaming_how_to.html
[8] http://www.shichaoji.com/2017/02/12/retrieve-streaming-posts-via-twitter-api-tweepy/
[9] https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object
[10] https://developer.twitter.com/en/docs/tweets/post-and-engage/api-reference

[11] https://stackoverflow.com/questions/41325743/how-to-stop-streaming-tweets-after-a-time-interval
[12] https://agsft.com/blog/challenges-real-time-data-vs-batch-data/
[13] https://www.datasciencecentral.com/profiles/blogs/batch-vs-real-time-data-processing
[14] https://www.linkedin.com/pulse/challenges-real-time-data-vs-batch-amit-saxena/
[15] https://www.infoq.com/articles/stream-processing-hadoop
[16] https://www.youtube.com/watch?v=pUUxmvvl2FE