**Wrangling ‘your Twitter’ data with Python & MongoDB**

Project:

Sateesh Babu P

July 21,2019

Version 1.0

# Document Revisions

|  |  |  |
| --- | --- | --- |
| Date | Version Number | Document Changes |
| 17.07.2019 | 0.1 | Initial Draft |
| 21.07.2019 | 1.0 | Final Draft |
|  |  |  |

# Approvals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Role** | **Name** | **Title** | **Signature** | **Date** |
| Instructor Name | Gaurav | Instructor |  |  |

Contents

[1 Document Revisions 1](#_Toc14594956)

[2 Approvals 1](#_Toc14594957)

[3 Introduction 3](#_Toc14594958)

[3.1 Problem statement 3](#_Toc14594959)

[3.1.1 Requirement Decomposition 3](#_Toc14594960)

[3.2 Project Scope 3](#_Toc14594961)

[3.2.1 In Scope Functionality 3](#_Toc14594962)

[3.2.2 Out of Scope Functionality 3](#_Toc14594963)

[3.2.3 Assumptions 4](#_Toc14594964)

[3.2.4 Constraints 4](#_Toc14594965)

[3.2.5 Pre-requisites 4](#_Toc14594966)

[4 Solution Overview 5](#_Toc14594967)

[5 Data Wrangling 5](#_Toc14594968)

[5.1 Extract & Load to MongoDB 6](#_Toc14594969)

[5.1.1 Full Load 6](#_Toc14594970)

[5.1.2 Delta Load 6](#_Toc14594971)

[5.2 Parsing, Transformation and Data Analysis 7](#_Toc14594972)

[5.2.1 Parsing to Data structures 7](#_Toc14594973)

[5.2.2 Transformations 7](#_Toc14594974)

[5.2.3 Analysis & Visualization 8](#_Toc14594975)

[5.3 Python code for execution 8](#_Toc14594976)

[6 Summary 8](#_Toc14594977)

[7 Appendices 9](#_Toc14594978)

[7.1 List of Acronyms 9](#_Toc14594979)

[7.2 Glossary of Terms 9](#_Toc14594980)

[7.3 Related Documents 9](#_Toc14594981)

# Introduction

## Problem statement

Twitter, a social micro blogging website interface, don’t have a provision to classify or to categorize and analyze user’s current as well as past tweets and retweets on subjects like - Machine Learning, AI and Big Data systems.

With high volumes, velocity and missing hashtags, we require a tool to store and process the current as well as archived tweets/retweet (replies) for further analysis or reference.

### Requirement Decomposition

* Extract and store user’s tweets (as well as retweets) on a periodic basis.
* Historical tweets shouldn’t be more a year.
* Data should be stored in its original form (i.e. Json) as shared by Twitter API.
* It can be any size and structure.
* Data to be parsed and tokenized before getting classified as per below priority:

|  |  |
| --- | --- |
| **Category label** | **Key words** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Project Scope

### In Scope Functionality

* Only user’s tweets and retweets are in scope for extraction.
* Tweepy (Twitter API wrapper)’s GET methods are used for data extractions:
  + Full Load - for initial load
  + Delta Load - for periodic load
* PyMongo, a MongoDB’s python driver, is used to connect MongoDB.
* sPacy, nlp package, is used to tokenize the text messages.
* Pandas are used for defining the data structures, data analysis and visualization.
* API calls should have timeouts so the program can’t block permanently.
* Provide API connection audit trail.

### Out of Scope Functionality

* Any other twitter actions like – liking a post, creating and deleting a tweet etc.
* Applying Machine learning models/techniques for twitter text classification.
* Real-time API extractions from Twitter.
* Enhancing or amending Python packages – Tweepy, Pandas, sPacy and PyMongo

### Assumptions

* Twitter text messages will have identified keywords for respective categories.
* Twitter user’s developer keys and access tokens will be active forever.

### Constraints

* Access to historical tweets is extremely limited. You can retrieve the last 3,200 tweets from a user timeline and search the last 7-9 days of tweets.
* Tweepy’s API.user\_timeline method limits 900 requests per 15 mins.
* If you are sharing datasets of tweets, you can only publicly share the ids of the tweets, not the tweets themselves.
* Any tweets which have been deleted or become protected will not be available.
* Sharing Twitter user tokens and passwords are restricted.

### Pre-requisites

#### Twitter API Access

* Create your account in Twitter.
* Visit the Application Management page at https://apps.twitter.com/, and sign in with your Twitter account
* Click on the "Create New App" button, fill in the details and agree the Terms of Service
* Navigate to "Keys and Access Tokens" section and take a note of your Consumer Key and Secret
* In the same section click on "Create my access token" button
* Take note of your Access Token and Access Token Secret.

Note: All of these parameters should be treated as passwords and should not be included in your code in plain text.

#### MongoDB Installation

Refer the link for instructions. <https://www.mongodb.com/download-center/community>

#### Python Packages.

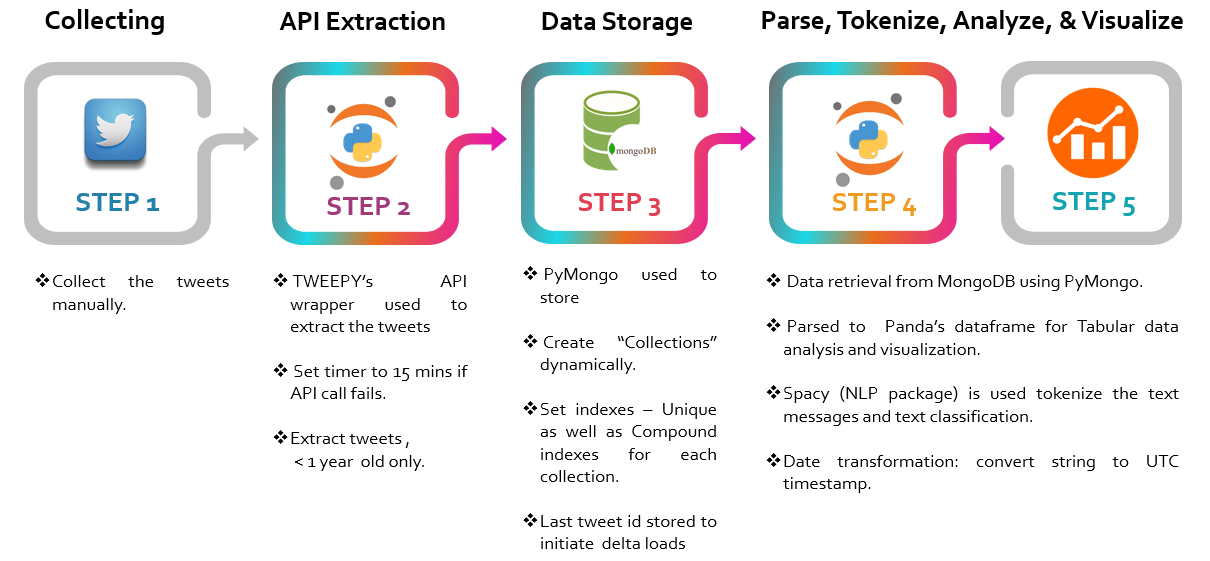
* pip install pymongo
* pip install tweepy
* pip install sPacy
* pip install pandas

#### Jupyter Notebook

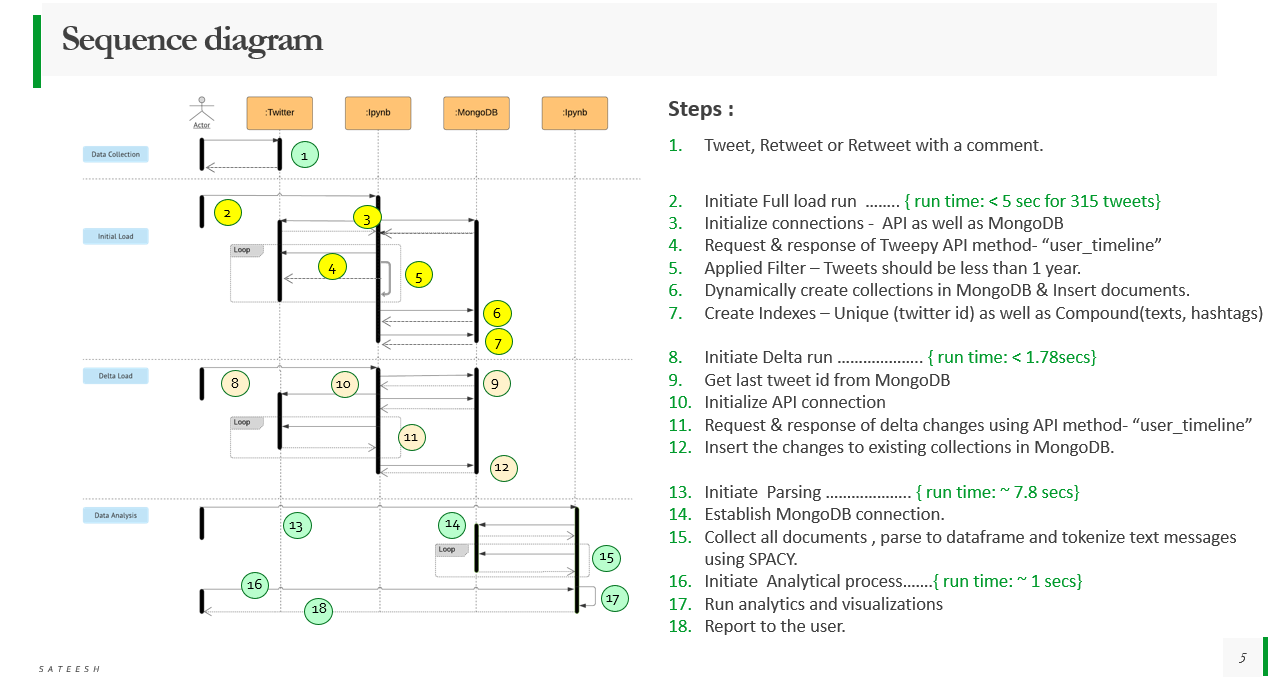
Refer the link for instructions. <https://jupyter.org/install>

# Solution Overview

Solution is mainly focused on API extraction, data storage, parsing and text tokenization.



This data wrangling solution is designed in view of performance & scalability strategies. Entire code is based on 12 functional components/programs. Each component does a specific task. They are tested and improvised for better performance, scalability, readability, and maintainability. Design also supports parallel processing of API extractions, like – initial run or delta run of user1 is independent to initial run or delta run of user2. Runs should be sequenced within standard twitter limits manually.



Please refer section 5.3 for code.

# Data Wrangling

In this project, the following are two major steps in data wrangling:

1. Extract and store the twitter(source) feed in raw data form to MongoDB.
2. Parse, tokenize and transform the raw feed into Panda’s data frame for further analytics.

The objective is to retain the originality of the source feed which can reused by any another project.

## Extract & Load to MongoDB

Python packages (Tweepy and PyMongo) are used in extracting and storing the twitter data in MongoDB. Full (or Initial) load is when the MongoDB collections are created dynamically, and twitter data is loaded for the first time in these new collections. A delta data load means that either loading changes to already loaded data or add new transactions in the existing collections.

### Full Load

The following are steps of Initial Load:

1. Initiate API
2. Run function “get\_all\_tweets” (screen\_name,api) for given user screen name.

The function does the following:

1. Call Tweepy API method – “user\_timeline” which returns 200 messages per request call.
2. Rerun the API method with max twitter id of the previous batch.
3. All the messages are appended to empty list
4. The above loop breaks if one of the two conditions are satisfied
5. No more tweets to extract from user’s account
6. Or if the last tweet in the batch has created time stamp (UTC) is one or above years old.

age = dt.datetime.utcnow() - tweet\_created\_date

tweet\_age\_days = age.days

break the loop….if tweet\_age\_days < 365

1. Create a new collection per user and use “insert\_many” to store tweets.
2. Finally, create unique as well as compound indexes on the newly created collection.

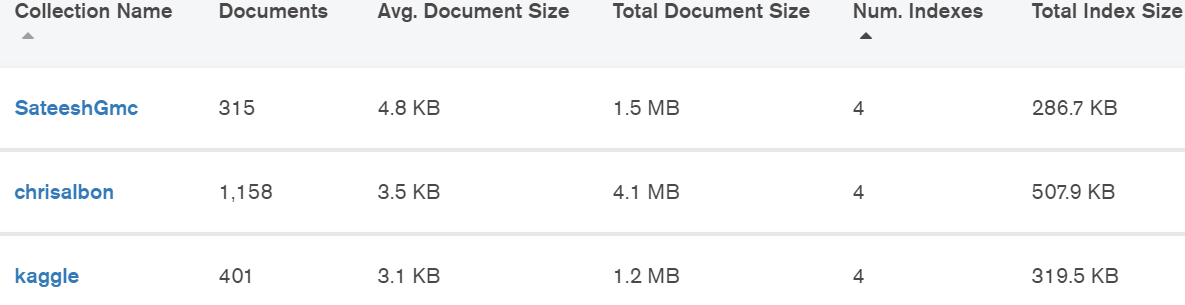


Noticed the approx. run time is less than 5 seconds in getting all 315 tweets.

### Delta Load

Delta run is designed for a periodic run either user can run every night or once in week/months

1. Initiate API
2. Get user’s tweet id from MongoDB collection
3. Run function “delta\_load(last\_tweet\_id)”. This function will call method “api.user\_timeline” and saves the delta changes (since last extract’s last tweet id) to existing collection in MongoDB.



Please refer section 5.3 for code. Have noticed the approx. delta run time is less than 1.78 seconds.

## Parsing, Transformation and Data Analysis

The following are the steps in involved in parsing, tokenizing and analyzing the twitter data:

### Parsing to Data structures

Before initiating the parsing, run a quick check on the MongoDB database for number tweets per collections using “count\_documents” and also to view the indexes of the collection with help of “index\_information().keys()”.

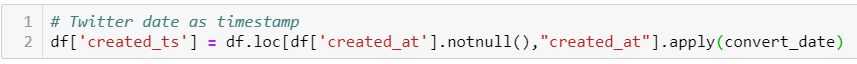
The documents (tweets) are sorted by twitter unique id (‘id’) and retrieved from each collection with help of PyMongo function – “fulldb[collection].find().sort('id', DESCENDING)”. There exists three different json structures – tweet, retweet and retweet with comment. Hence, the following condition is used in creating a common data structure while retaining their statuses. Using a function loop, the completed structure(dictionary) of each tweet is appended to data frame for further tabular data analysis. To my surprise, this function run time is only 7.8 seconds. Please refer section 5.3 for the complete code.



### Transformations

#### Time stamp creation

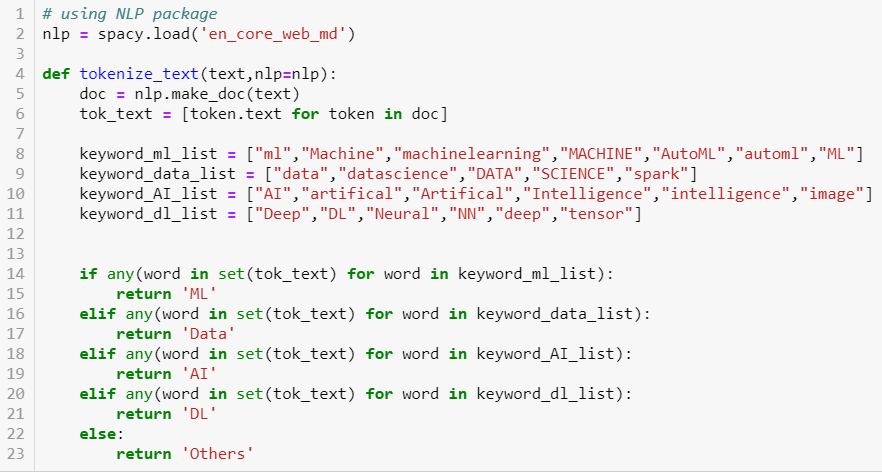
Used a function “convert\_date(string)” to convert “create timestamp” of tweet into a UTC timestamp.



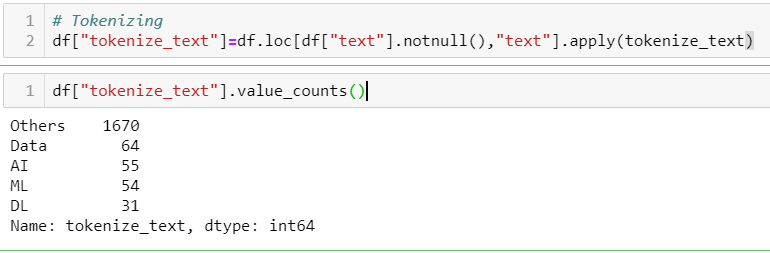
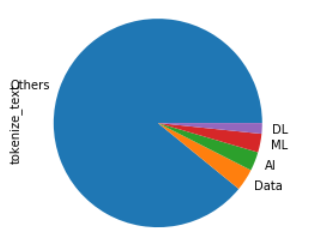
This helps for time-series analysis.

#### Tokenization using SPACY

Used “SPACY” natural language processing package to tokenize the tweet’s text messages. This is easiest and accurate way to create list of words from sentences, In fact, it’s better than using a regular expression. Even “SPACY” provides a rule-based matching. Due to lack of time, I haven’t explored the options. I used the following custom function to classify the tweets, which took less a 1 second.

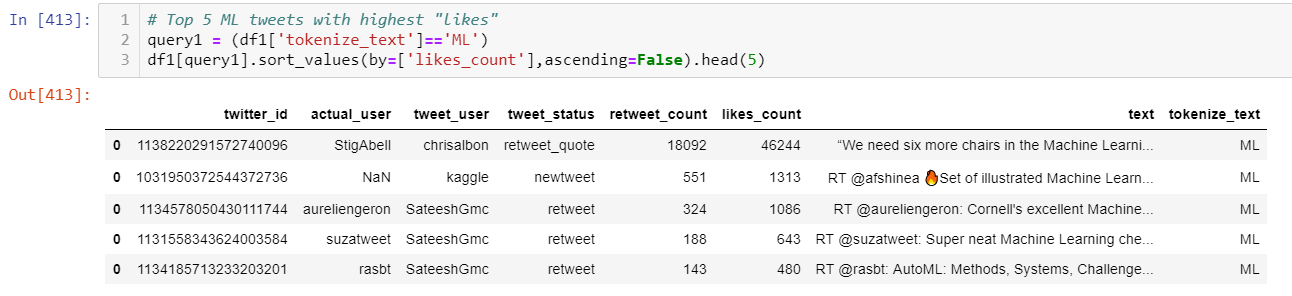
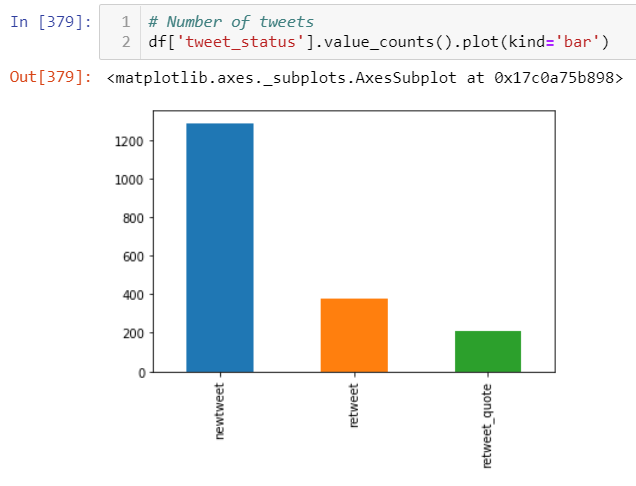


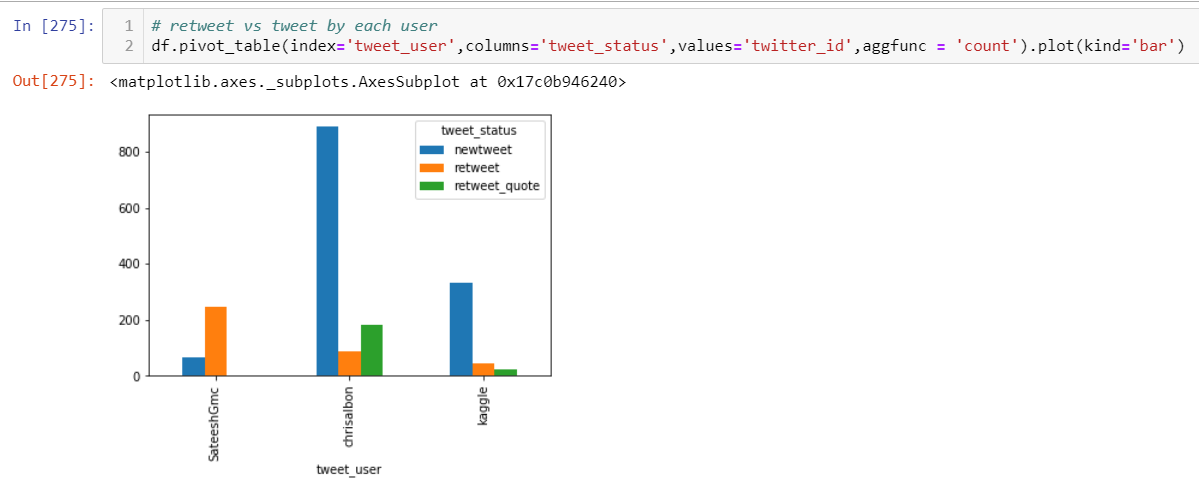
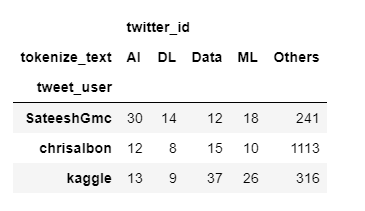
Tweets classified as below:

### Analysis & Visualization

The following are few analytics and visualizations:

## Python code for execution



# Summary

To summarize, I have analyzed Twitter API feeds and rate limits. The extracts (full & delta loads) are designed using Tweepy API wrapper in view of performance strategies, like - Set time filters and # tweets per call, code is componentized into 12 functions. Data is saved to MongoDB by creating collections dynamically (per user) …..to support future scalability. Leveraged MongoDB functionality to get inputs (last tweet id per user) for delta load. Also, using MongoDB find function, I have retrieved raw data. Parsed, applied transformations & did Tokenization using SPACY (NLP package). Finally, I did data analysis and visualization using Pandas.

Hopefully this guide has provided enough of a description of the landscape for Twitter data wrangling that you can move forward with further research on text analytics and NLP.

# Appendices

## List of Acronyms

|  |  |
| --- | --- |
| Acronyms | Meaning |
| ETL | Extract Transform and Load |
| DW | Data warehouse |
| DB | Database |
| API | Application Programming Interface |
| NLP | Natural Language Processing |

## Glossary of Terms

**Twitter data** is the information collected by either the user, the access point, what’s in the post and how

users view or use your post. While this might sound somewhat vague, it’s largely due to the massive

amount of data that can be collected from a single Tweet.

**Data wrangling (ELT)**, sometimes referred to as data munging, is the process of transforming and mapping

data from one "raw" data form into another format with the intent of making it more appropriate and

valuable for a variety of downstream purposes such as analytics. A data wrangler is a person who performs

these transformation operations.

**Extract-Transform-Load (ETL)** is different from data wrangling. Here, we extract source data and apply the

required transformations and then, load the refined data to target system.

## Related Documents

|  |  |
| --- | --- |
| Topic | Link(s) |
| Tweepy | <http://www.tweepy.org/> |
| PyMongo | <https://api.mongodb.com/python/current/> |
| Twitter API rate limit | <https://developer.twitter.com/en/docs/tweets/timelines/api-reference/get-statuses-user_timeline.html> |