**METHODOLOGY**

Twitter basically has two APIs for developers who have accounts with them to retrieve data, the Representative State Transfer (REST) API and the Streaming API. The REST API allows you to go back in time to retrieve tweets, often this includes going 7 days back, as it always comes with a limit, unless you want to subscribe to premium access, where you pay a certain amount of money.

When we speak of a REST system, we are referring to any interface between systems that use HTTP to obtain their and generate their operations on those data in all possible formats. The streaming API as the name implies looks into the future, and capture tweets as they arrive in real time.

Twitter API’s has some implementation which were not in favor of the direction of the project. For instance the fact that you can’t get tweets older than 7 days, also, the maximum number of tweets you could request for with the API was just 100. and lastly we need historical tweet data, of about 10 years worth, so the Twitter API was not just the way.

It was at this stage we decided to adopt the web scraping technique, and that involved querying the twitter search engine, using tools such as PyQuery, Lxml, and Beautifulsoup4. We used and modified GetOldTweets3 to suit our purpose, which was an improvement fork of the GetOldTweet-Python web scraping script Originally written written by Jefferson Henrique. We specified the search keyword of interest, the geolocation (Longitude and Latitude), Duration of search (in form of start data and end date), the tweets language to be returned in the output, and the search radius from the geolocation details (Longitude and Latitude).

How the algorithm works is that it scrapes twitter search very fast, lookng for the keyword you specified in your code, and other paarameters until the end of the search duration you specified. And if you didn’t specify any, it would scrape till there is no more tweet to scrape anymore.

Putting this web scraping technique and script to use, in order to gather our data, we set the search location to the geographic coordinates of the polytechnic university Hong Kong which is () in Longitude and Latitude, and set a 5km radius around the university. Also, we set the start and end date of the search to 2008-01-01, and 2019-06-30. Finally we set our search keywords, we had a number of them ranging to about 21 in total, these include the following; Air pollution, Burning, climate change, community, disorder, events, family, flooding, funeral parlour, housing, illegal, noise, parking, rents, resilience, resilient, safety, shopping, students, university, waste. We started with using students as the search keyword, and iterated the process for all of the remaining 20 keywords. So all tweets that has any of the keyword specified would be selected and saved.

We also noticed that a large majority of the tweets were in chinese Language, and not very useful for our analysis, so we filtered then out which took a sizable chunk out of our big data.

At the end of the mining, we had 40,853 tweets saved in a csv file. It is worthy to note that these tweets were accompanied with some underlying metadata, such as replies, username, permalink, favorites, and retweets. Attached to this section is the web scraping scipt and a link to our github, to get the full description of the code in the readme.md section, in case the data needs to be reproduced.

**Preprocessing the tweets**

One major unique characteristic of textual data is that they are unstructured, and need to be pre-processed in order to get the right insights from it. After gathering our dataset of over 40,000 tweets, we pre-processed it using Pandas- a convenient python library for data analysis and manipulation. The preprocessing operation we used for the dataset was majorly cleaning the tweet text, getting rid of special characters & symbols such as “@, £, $, #. :, /z,\ etc”. In a bid to making the tweet text more readable. We just wrote a function in Python that we could also call to do that for us, and as usual, the code used to do this is attached at the end of this section. # attach code

After preprocessing, the next step in our pipeline was to generate the sentiments for each of the tweet we gathered, and this was achieved with the help of Python’s TextBlob and VADER Libraries, which are widely known for their efficiency in handling Natural Language Processing tasks such as sentiments analysis, part of speech tagging, noun phrase extraction, classification, translation and more.

It is worthy to note that the sentiment/emotion mining analysis was carried out based on each of the search keywords which were further broadly classified into Environmental, Social and Economic issues.

After rating each tweet by giving its respective sentiment score using TextBlob and Vader Libraries, we were able to generate some further analysis and plots to derive some more insights in our data. One of which was the line graph, indexed over time ‘t’ from 2009 till 2019, showing the distribution of Negative sentiments over the years. With this plot we could actually tell what months or seasons of the year experienced the most negative sentiments, as well as the least negative tweets. We were also able to see and identify patterns that have always occurred over the years. For instance in the Noise category, under the environmental issues, we identified that more negative sentiments about Noise were recorded around march-April and then in October-December which strongly correlates with the fact that school session runs from September till April every year.

Another time line analysis we did was to show the monthly volume of tweets on a line plot over the 10 year period we gathered the data from. With this we were able to also get more insight and identify trends in the monthly volume of tweets, we also noticed that some months had little to no tweets, and some had very high volume of tweets.

Basically we were just manipulating the data with the help of python’s pandas library, to try new stuffs, get more insights, generate new analysis etc. At the end of this section is the link to the github account where you can find all the code used for all these analysis discussed here in.

**TOPIC MODELING USING LATENT DIRICHLET ALLOCATION (LDA) MODEL**

Topic modeling is a form of text mining, employing unsupervised and supervised statistical machine learning techniques to identify patterns in a corpus or large amount of unstructured text. It can take your huge collection of documents and group the words into clusters of words, identify topics, by a using process of similarity.

Latent Dirichlet Allocation on the other hand is an unsupervised learning algorithm that attempts to describe a set of observations as a mixture of different categories, the categories are themselves a probability distribution over the feature. Tasks which LDA can be useful for:

1. Clustering customers based on product measures

2. Automatic Harmonic Analysis in music

3. Topic model in Text Corpus.

For Topic modeling, more preprocessing has to been done in order to ensure that the final output is a correct representation of the total document, which in our case is the list of all the tweets with negative sentiments. This preprocessing operations in no partcular order includes:

1. Removal of stop words

2. Creation of Bigrams and Trigrams

3.Word Tokenization

4. Lemmatization

All these preprocessing if done properly would go a really long way in a improving the coherence of the final output.

**Removal of Stop words:**

Stopwords are the English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. For example, the words like, is, the, he, have, there, a etc. if we have a sentence like, “There is a tree near the river” and we run it through a stop words pipeline, we get the following output - “There, tree, near, river.” which can still be denoted to mean the former.

Creation of Bigrams and Trigrams: A bigram is a word pair. The bigrams within a sentence are all possible word pairs formed from neighboring words in the sentence. It is easier to look at an example. The bigrams in the sentence. **I really love Dancing** are **I really**, **really love,** and **love Danicing.**

That is a total of 3 word pairs, or bigrams. The same goes for trigrams, or triplets. This sentence would contain only two trigrams, which are **I really love**, and **really love Dancing**.

**Word Tokenization**

Tokenization is the process of tokenizing or splitting a string, text into a list of tokens. One can think of token as parts like a word is a token in a sentence, and a sentence is a token in a paragraph.

**Lemmatization**

In simpler words, I would explain lemmatization as returning different forms of a single word to its root form, for instance the Lemmatized version of “Generated” would be “Generate”, similarly “Went” is “go”, “Bough”t is “Buy” etc. This would be iterated for all the tweet text in the dataset. Also the Lemmatized data should be your final-preprocessed text that could be at this stage can be fed into an LDA model.

It should be of note that In topic modeling, the bulk of the work is in the data preprocessing stages. The LDA model is just a single line of code.

And from the output, we could see the distribution of topics and words across the document which is just essentially the list of all the negative tweets. So with that result and output, we could summarize the list of negative tweets to confirm if they were really Negative, and to also see its associative topical distribution comparing it with the pilot survey that was carried out on the residents, non residents, shop owners, and estate agents. It is worthy to note that the LDA Topics generated were somewhat correlated with the outcome of the pilot survey.

Link to the Github where you can find the codes and visualisations generated.

<https://github.com/marquisvictor/Modified-GetOldTweets3>

**REFERENCE**

* <https://textblob.readthedocs.io/en/dev/>
* <https://provalisresearch.com/blog/topic-modeling/>
* <https://www.quora.com/What-is-a-bigram-and-a-trigram-layman-explanation-please>
* <https://matplotlib.org/3.1.0/gallery/color/named_colors.html>
* <https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8>
* <https://github.com/Jefferson-Henrique/GetOldTweets-python>
* <https://twitter.com/login>
* <https://developer.twitter.com/en/apply-for-access.html>

# Modified-GetOldTweets3

GetOldTweets-Python is a project written in Python to mine old and backdated tweets, It bypasses some limitations/restrictions of the Twitter API. This Repo houses an improvement fork of the original GetOldTweets Library by Jefferson Herique (https://github.com/Jefferson-Henrique/GetOldTweets-python). The improvement makes running this package on Windows OS seamless with Python 3.x.

## Details

Before coming here to get backdated timeline tweets, you might have gone to Twitter to create a Developer account, before the sad reality dawns on you that you cannot mine tweets older than 7 days, using twitter's rest API, unless you pay and even when you do,you are restricted to a limit of 100 tweets daily. Just how slow would that be if you intend to mine say 10 years worth of textual big data for twitter for analuysis. You might have also tried Googling some other methods to mine old tweets, you might have seen different sites, libraries or tools you didnt know how to use, and all sorts of confusing information on the web. This modified getoldtweets3 library helps you download backdated-timeline twitter data easily and without hassles from command line, either on windows OS, Ubuntu Linux OS, or Mac OS. You only need to have python installed on your machine, yeah. you also need to set the environment variable or install ppython in such that you can call it from the command line.

## Prerequisites

This packages assumes you are running puthon version 3.x on your local machine, and that you have already set the python environm ent variable path, so you interactively fire up python from command prompt or terminal without getting any error. If you haven't kindly follow this [stackoverflow answer](https://stackoverflow.com/questions/3701646/how-to-add-to-the-pythonpath-in-windows-so-it-finds-my-modules-packages) for guidance. and [this too](https://stackoverflow.com/posts/54934172/edit). After doing all those, the next major packages you need to install are pyQuery, and Lxml for handling requests and xml/html documents. easy stuffs just use `pip install pyquery` and `pip install lxml`

## Components

- When you run this package from command line, it typically returns the following as columns in an output.csv file. It should be noted before hand that the geo attribute returns an empty column. You use the geographical coordinate as well as the search radius to get a boundary within which you want to retrieve your tweets data.

- id (str)

- permalink (str)

- username (str)

- text (str)

- date (date)

- retweets (int)

- favorites (int)

- mentions (str)

- hashtags (str)

- geo (str)

## Components

- username (str): An optional specific username from a twitter account. Without "@".

- since (str. "yyyy-mm-dd"): A lower bound date to restrict search.

- until (str. "yyyy-mm-dd"): An upper bound date to restrist search.

- querysearch (str): A query text to be matched.

- toptweets (bool): If True only the Top Tweets will be retrieved.

- near(str): A reference location area from where tweets were generated.

- within (str): A distance radius from "near" location (e.g. 15mi).

- maxtweets (int): The maximum number of tweets to be retrieved. If this number is unsetted or lower than 1 all possible tweets will be retrieved.