Analysis of Twitter posts for Cape Town, South African Users

Brief Introduction

This is a simple program that analysis the Cape Town, South Africa tweets posts. We grouped the data based on location and the program then use the location to find the most frequency words users used on Twitter. Furthermore, we can used the analysis to find users ID and we predict the different emotions attached to the text. However, in this report we consider using the user id,location,text.

Experimental Overview

This program data was pre-processed using pandas, NLTK, gensim, and numpy libraries within the Python virtual environment. Plots were created using plotly.

Project Setup Environment Overview:

1. Python Virtual Environment; e.g. Anaconda
2. Used Python Libraries and Jupyter Notebook; egg pandas, NLTK, gensim, and numpy libraries within the Python
3. GitHub; versioning control

Implementation Overview: Scripts

Preprocessing steps of the Program

The following show the steps of the program. Data were cleaned and then we find by creating and grouping words of the most frequent used. The steps are as follows:

1.Data Cleansing

1. Remove URLs
2. Remove usernames
3. Remove tweets with *empty column*
4. Remove special characters
5. Remove numbers

2.Text processing

1. Tokenize
2. Transform to lowercase

3.Build word list for Words

Cleansing the tweets subset datasets

In this program, the subset_data_Cleanuper class was created. The class consists of methods or functions that allow task execution in the list. This is shown below. Most of those is done using regular expressions. The class exposes it's interface through iterate() method - it yields every cleanup method in proper order.

In [58]:
```
class subset_data_Initialize():
    # creating an arraries
    data = []
    processed_data = []
    wordlist = []
    # setting data_model and labels to none
    data_model = None
    data_labels = None
    is_test_data = False #

    def initialize(self, csv_file, is_testing_set=False, from_cached=None):
        if from_cached is not None:
            self.data_model = pd.read_csv(from_cached)
            return

        self.is_test_data = is_testing_set

        if not is_testing_set:
            self.data = pd.read_csv(csv_file, header=0, names=["id", "location", "text"])

        else:
            self.data = pd.read_csv(csv_file, header=0, names=["id", "text"],dtype={"id":"int64","text":"str"},nr
            not_null_text = 1 ^ pd.isnull(self.data["text"])
            not_null_id = 1 ^ pd.isnull(self.data["id"])
            self.data = self.data.loc[not_null_id & not_null_text, :]

        self.processed_data = self.data
        self.wordlist = []
        self.data_model = None
        self.data_labels = None
```

The class above load the data from the given tweets subset csv file for further processing, or we also preprocessed the file into cached and the subsequent execution read from the cache. Moreover, we ensure to avoid the empty entries, and yes we removed the empty entries. We added additional class properties such as data_model, wordlist etc for further use.

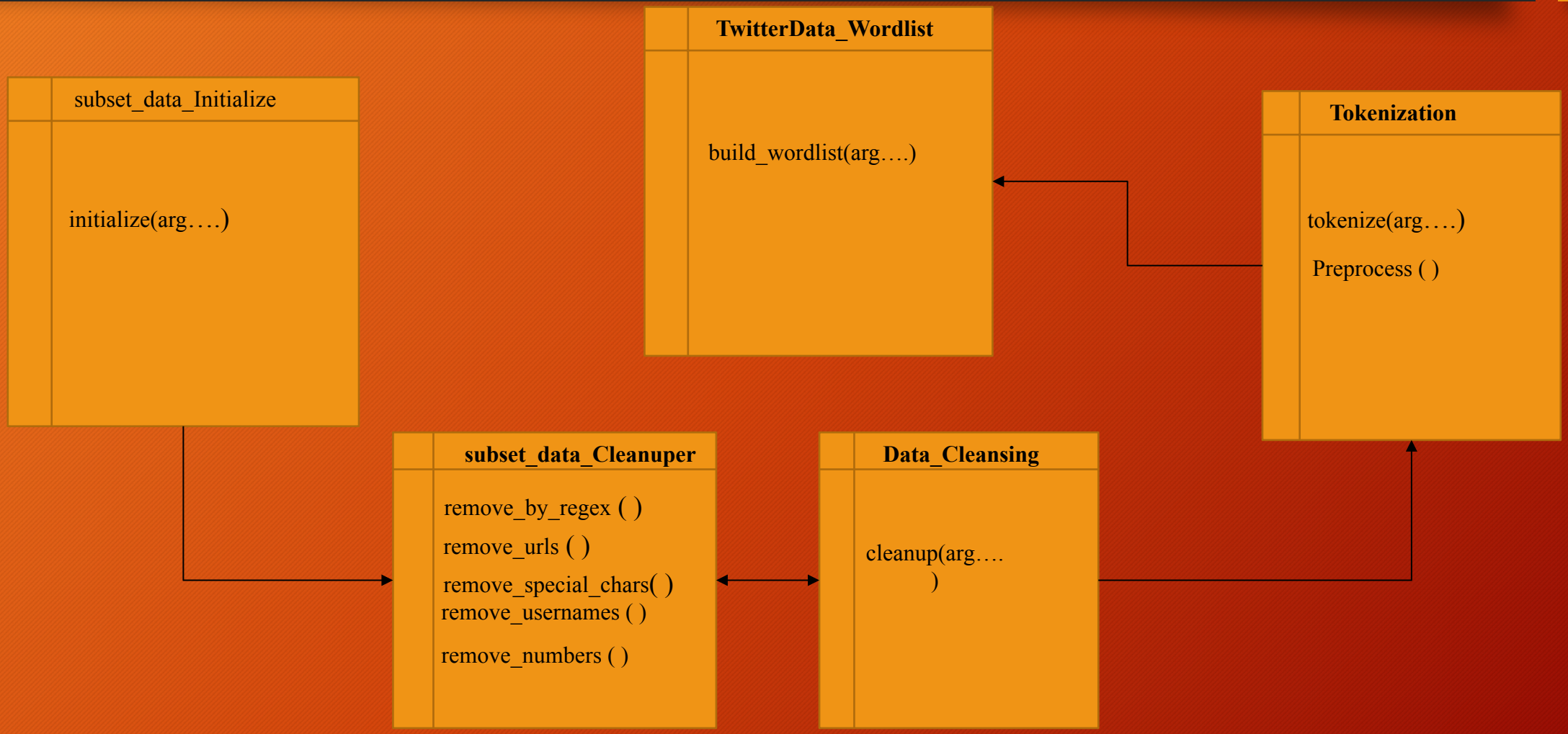
In [59]:
```
# Call an instance of the class and se
data_class_instance = subset_data_Initialize()
data_class_instance.initialize("../data/subset.csv")
data_class_instance.processed_data.head(10)
```

Out[59]:

	id	location	text
0	8.550000e+17	South Africa	RT @ZackieAchmat: #UitmetZille #UitMetZuma - a...
1	8.550000e+17	South Africa	RT @ZackieAchmat: #UitmetZille #UitMetZuma - a...
2	8.550000e+17	South Africa	RT @ReclaimCT: Tonight @ReclaimCT came for sup...
3	8.550000e+17	South Africa	RT @ReclaimCT: Tonight @ReclaimCT came for sup...
4	8.550000e+17	cape town	The demographics in that room (beyond Reclaim ...
5	8.550000e+17	cape town	The demographics in that room (beyond Reclaim ...
6	8.550000e+17	Cape Town, South Africa	@ReclaimCT ...refer the part \rI prefer to liv...
7	8.550000e+17	Cape Town, South Africa	@ReclaimCT ...refer the part \rI prefer to liv...
8	8.550000e+17	Cape Town, South Africa	RT @ReclaimCT: Tonight @ReclaimCT came for sup...
9	8.550000e+17	Cape Town, South Africa	RT @ReclaimCT: Tonight @ReclaimCT came for sup...

Schematic Overview: Class Modules

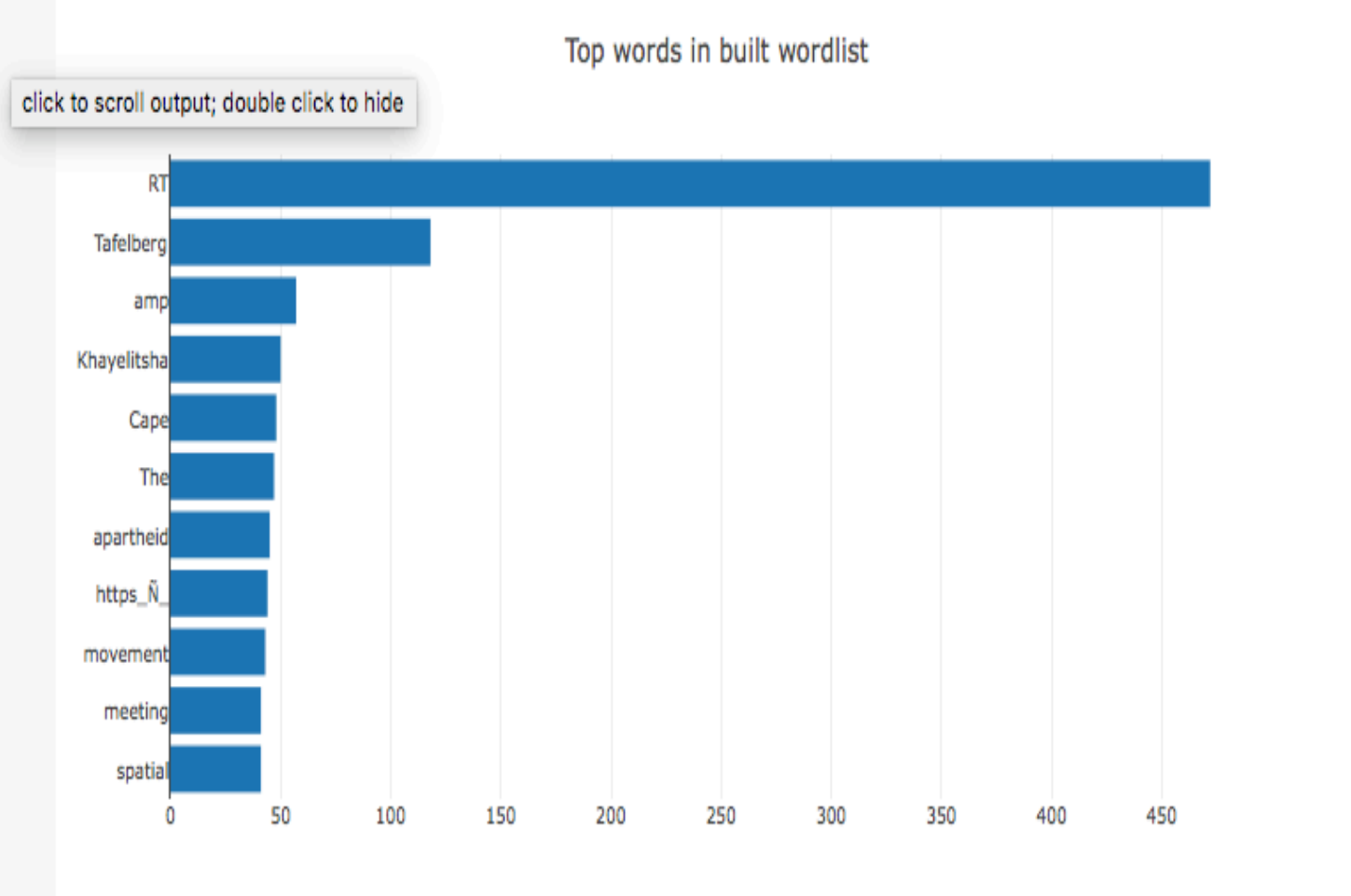
This program uses an "object oriented programming" to structure the program as shown below;



Result Overview: Data Representation Graphically

The most common words (as expected) are the typical English stop words. We filter them out, and as purpose of this analysis is to determine words like ”Cape Town" and ”South African". Having this in mind, these words are represented in the graph.

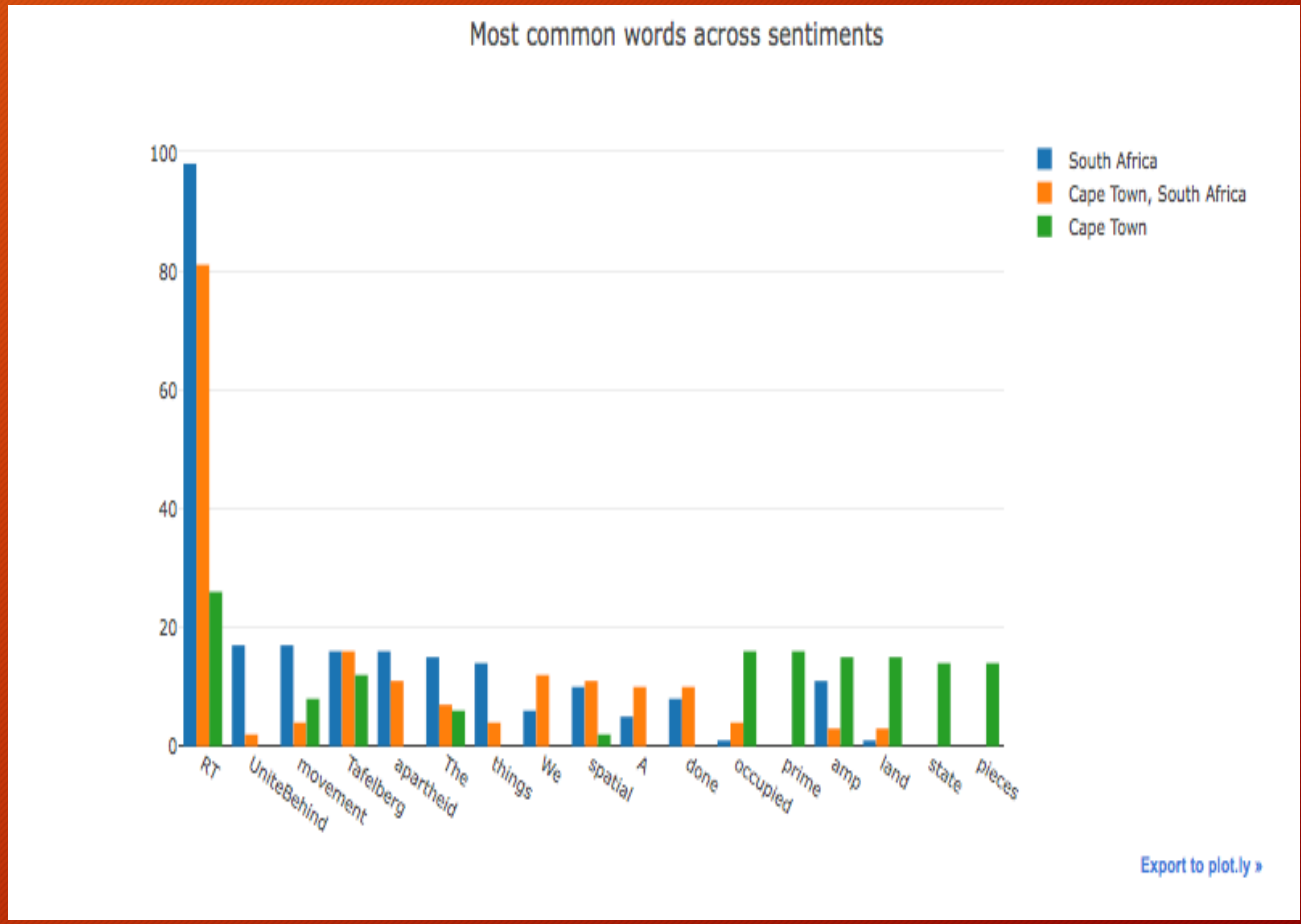
To access the graph interactively, click the following links; https://plot.ly/~boraton/3/



Result Overview: Data Representation Graphically

Some of the most common words show high distinction between classes like retweets and apartheid and other are occurring in similar amount for every class. The most common words hashtag (Retweets, RT) is unique in South African in general. At this point, it's clear that words across several locations are challenges.

To access the above graph interactively; click the following link: https://plot.ly/~boraton/1/



Summary

We showed that certain text is a non-trivial task can be parse further for machine learning. Moreover, the huge data require thorough preprocessing and algorithm to simulate the most frequent tweets in Cape Town, South. This analysis require further and advance processing.

Project Repository

This post is compiled version of Jupyter Notebook, which can be download here:

https://github.com/boratonAJ/CT_Tweets_Analysis_Report.git

References

Website References:

1. https://github.com/bonzanini/nlp-tutorial
2. https://marcobonzanini.com/2015/03/09/mining-twitter-data-with-python-part-2/
3. https://github.com/bonzanini/Book-SocialMediaMiningPython
4. https://stackoverflow.com/questions/35074222/python-read-csv-into-dataframe-cant-convert-object-into-float
5. https://stackoverflow.com/questions/31729288/pandas-dataframe-and-u-u2019
6. https://stackoverflow.com/questions/38134643/data-frame-object-has-no-attribute
7. https://regexone.com/references/python