## Application idea

The idea of this application is to tunnel into Twitter live tweets through Twitter API and to break up sentences from tweets into words then to filter the words and send them to a wordcount script and then send word to an already created postgres sql dataframe.

This word count script will take the word and check if it is already existing in the dataframe, and if it is then it will update the dataframe with the new word count, if the word does not exist in the dataframe then it will create a new entry for the word with a count of 1.

The application will continue to run until it is exited from (ctrl  + c), but while the application is running it is live streaming data from Twitter tweets and converting them into postgres table entries. Postprocessing can be down on the postgres table via two scripts histogram.py and finalresults.py (discussed later).

## directory and file structure

```
exercise_2
   |___>  extweetwordcount
      |___>  src
         |___>  bolts
            |___>  __init__.py
            |___>  parse.py
            |___>  wordcount.py
         |___>  spouts
            |___> __init__.py
            |___> tweets.py
      |___>  topologies
            |___>  wordcount.clj
      |___>  virtualenvs
            |___>  wordcount.txt
   |___>  finalresult.py
   |___>  histogram.py
```

Above is a look into the directory/file structure of the application. Blue text dictates that it is a folder, black text dictates that it is a file used in the twitter application sparse run, and red shows that is a postprocessing code for application.

Files updated to run application:
parse.py,
wordcount.py
tweets.py
wordcount.clj
wordcount.txt

## File dependencies

parse.py
import re
from streamparse.bolt import Bolt

wordcount.py
from collections import Counter
from streamparse.bolt import Bolt
import psycopg2
Proper postgress database and dataframe connection

tweets.py
import itertools, time
import tweepy, copy
import Queue, threading
from streamparse.spout import Spout
Also: Twitter credentials to access twitter api

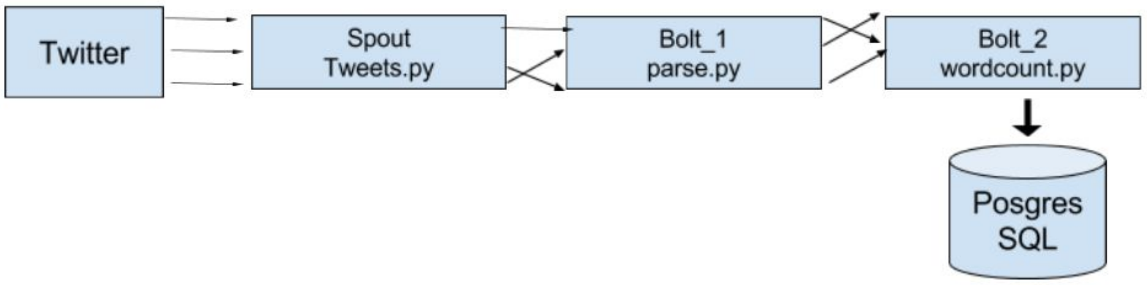
- consumer_key
- consumer_secret
- access_token
- access_token_secret

wordcount.clj
parse.py
wordcount.py
tweets.py

wordcount.txt
streamparse # always required for streamparse projects
tweepy # required for tapping into tweet source

Need python packages to run application:
Tweepy: pip install tweepy
Psycopg: pip install psycopg2==2.6.2

## Description of the architecture



The flow:

1. Run tweets.py
    a. This python script is used in order to attach itself to the twitter feed with authentication twitter credentials and listens to the twitter feed and sends it to bolt_1
2. Then it runs parse.py
    a. This python script takes the tweet feed that has been emitted from tweets.py and splits the Twitter feed into individual words and parses out special character that aren't considered words
3. Then it sends information to wordcount.py
    a. This python script takes the filtered words from parse.py and either creates a new entry into a dataframe(already created) with count 1, or if the word already exists in the sql database/dataframe it inserts a new count (ex. Word, count += 1)
4. This will run until killed (ctrl + c) since it is streaming from twitter feed
5. Postprocessing scripts can be used then to parse Postgres database and dataframe
    a. Finalresult.py
        i. If input:
            1. Python finalresult.py -> returns list of all words and their counts
            2. Python finalresult.py <list of words separated by spaces> -> returns all words in the space separated input and their individual counts
    b. Histogram.py
        i. Input:
            1. Python histogram.py <Number1>,<Number2> -> returns list of words that have counts in between the two number ranges

### Any necessary information to run the application

Need to make sure streamparse, python (including package dependencies), apache storm are all on your AWS instance. More information on those can be gathered from lab6 and exercize2 readme provided by instructors of w205. Also need postgres set up and started with proper connection to postgres server.

Readme.md shows how to run the scripts needed.