Twitter Data Analysis
(Documentation)

The twitter data analysis assignment focused on three major things:

- ➔ Accessing the twitter API (using python)
- ➔ Estimating the sentiment of a particular term (word)
- ➔ Analyzing the number of times a term occurred (frequency), top ten hash-tags of the downloaded twitter data sample and the mood (sentiment) of a location

Problem 1:

The first problem deals with accessing the twitter data. For accessing this data, an account on the twitter developer's app had been created. Using the individual twitter app credentials (API key, API secret, Access token, Access token secret), twitter data flows in (command used : $ python twitterstream.py > output.json). The data is saved in the .json file and used to solve the remaining problems.
One problem with the data flowing in was that it kept on getting disconnected within a time span of 2-3 minutes, therefore not a huge amount of data could be collected at once.

Problem 2:

For this part, the sentiment of each individual tweets were analyzed. AFINN-111.txt contained words and their sentiment scores. New words (not in the AFINN-111.txt) were kept at a score of 0 (in this problem). First of all, the tweets were checked to ensure it was in English. Python dictionaries played a huge rule in solving the problem.
Following are a few lines of code to show what this part of the program is executing:

```
# input: tweet as json
# output: tweet terms as list

words = tweet_json ['text']
        # unicode to str
```

```
         encodeWords = words.encode('utf-8')
        tweet_terms =  re.findall(r'\w+', encodeWords) # re.findall is used to get
the tweet terms/words as list

# Get score from the tweet
# input: tweet as json, sentiment dictionary (hashmap and/or map, python
dictionary)
# output: tweet score

    tweet_score = 0
    tweet_terms = getENTweet(tweet_json)

    for term in tweet_terms:
         term = term.lower()
        tweet_score = tweet_score + sentDict.get(term,0)
```

Basically, the tweets are taken in as json and then separated into a list of terms. Matching the term with the existing terms in the sentiment dictionary (AFINN-111.txt) and adding them up for each individual tweet, the sentiment of the tweet is derived.

Problem 4:

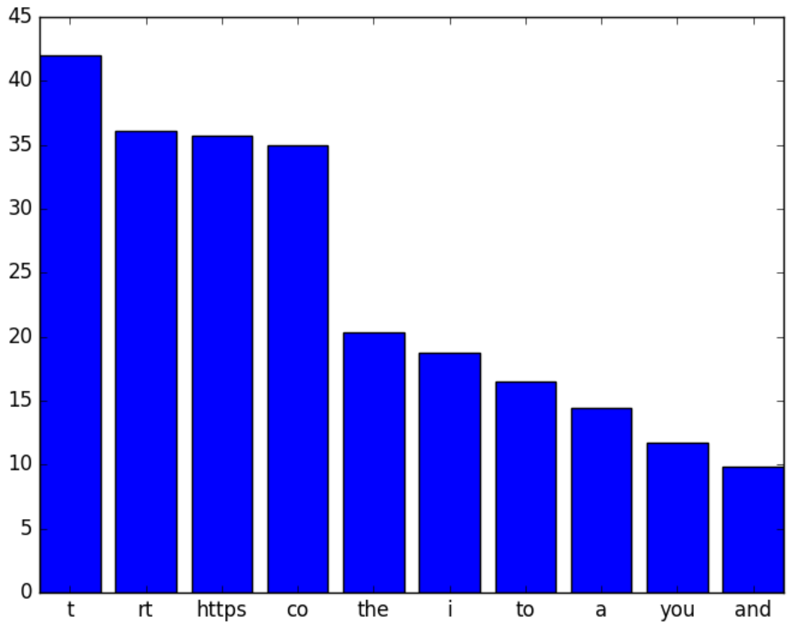
Problem 4 deals with term frequency, meaning the number of times a specific term has occurred throughout the entire twitter data collected. The total number of words in the tweet terms' list is summed (I found this an easier way to calculate all the terms in all the tweets). An empty dictionary called freq{} is used to store the terms and increment them each time it is repeated. After going through the whole data, the probability of a term occurring is calculated using the formula : [# of occurrences of the term in all tweets]/[# of occurrences of all terms in all tweets]

```
# Input: The downloaded tweets file
# Output: The freq dictionary {key: tweet terms, value: frequency as
probability}

        total_words = total_words + len(tweet_terms)
# if the term (key) exist in the freq dict, the value is incremented to keep count
of the number of times the term's occurred
             try:
```

```
           freq[term] += 1
         # new term is added and the value set to 1 for that term (key)
          except:
            freq[term] = 1

    # Finds frequency of each term in the file
    for term in freq:
          freq[term] = (freq[term]/(total_words))*1000 # multiplied by 1000 to
give a larger value
```



The graph shows the average number of terms that have occurred from the total number of words present in the tweet file {(freq[term] (total_words))*1000)}

Figure : Shows the most frequently occurred terms in the downloaded tweets

As the graph already demonstrates, the most occurred terms are not necessarily proper English words. To get proper English words, Text Blob can be used.

Problem 6:

For the top ten hash tags in the data, it was extracted similarly as the terms were (keeping the hash-tags in a hashTagFreq dictionary). Twitter API contained information on how to extract hash tags. The following implementation ignores the unicode characters in the hash tag list:

```
      # hashtag entities is being extracted
        hashtag = tweet_json['entities']['hashtags']
        #print hashtag
        for htags in hashtag:
          getHashtags = htags ['text']
           # reference : https://stackoverflow.com/questions/15321138/removing-unicode-u2026-like-characters-in-a-string-in-python2-7
          storeHashtag = getHashtags.encode('ascii','ignore')
           hashTagFreq[storeHashtag] = 1 + hashTagFreq.get(storeHashtag, 0)
```

References:

- https://stackoverflow.com/questions/10288016/usage-of-unicode-and-encode-functions-in-python
- https://dev.twitter.com/overview/api/entities-in-twitter-objects#hashtags
- https://stackoverflow.com/questions/15321138/removing-unicode-u2026-like-characters-in-a-string-in-python2-7
- https://dev.twitter.com/oauth/overview/authentication-by-api-family
- https://dev.twitter.com/rest/reference/get/search/tweets
- https://docs.python.org/2/library/stdtypes.html#typesmapping
- https://dev.twitter.com/overview/api/tweets