Mini Project
5th Semester
Twitter Sentiment Analysis

Design and develop an application to detect sentiments of the tweets.

Introduction:-

Twitter represents a fundamentally new instrument to make social measurements. Millions of people voluntarily express opinions across any topic imaginable --- this data source is incredibly valuable for both research and business.

For example, researchers have shown that the "mood" of communication on twitter reflects biological rhythms and can even used to predict the stock market. A student here at UW used geocoded tweets to plot a map of locations where "thunder" was mentioned in the context of a storm system in Summer 2012.

Researchers from Northeastern University and Harvard University studying the characteristics and dynamics of Twitter **have an excellent resource** for learning more about how Twitter can be used to analyze moods at national scale.

In this assignment, you will

- access the twitter Application Programming Interface(API) using python
- estimate the public's perception (the *sentiment*) of a particular term or phrase
- analyze the relationship between location and mood based on a sample of twitter data

Some points to keep in mind:

- This assignment is open-ended in several ways. You'll need to make some decisions about how best to solve the problem and implement them carefully.
- Your code will be run in a protected environment, so you should only use the Python standard libraries unless you are specifically instructed otherwise. Your code should also not rely on any external libraries or web services.

The Twitter Application Programming Interface

Twitter provides a very rich REST API for querying the system, accessing data, and control your account. You can read more about the Twitter API.

Python 2.7.3 environment

If you are new to Python, you may find it valuable to work through the codeacademy Python tutorials. Focus on tutorials 1-9, plus tutorial 12 on File IO. In addition, many students have recommended Google's Python class.

You will need to establish a Python programming environment to complete this assignment. You can install Python yourself by downloading it from the Python website.

Unicode strings

Strings in the twitter data prefixed with the letter "u" are unicode strings. For example:

```
u"This is a string”
```

Unicode is a standard for representing a much larger variety of characters beyond the roman alphabet (greek, russian, mathematical symbols, logograms from non-phonetic writing systems such as kanji, etc.)

In most circumstances, you will be able to use a unicode object just like a string.
If you encounter an error involving printing unicode, you can use the encode method to properly print the international characters, like this:

```
unicode_string = u"aaaàçççñññ"
encoded_string = unicode_string.encode('utf-8')
print encoded_string
```

Problem 1: Get Twitter Data

Important: For this assignment, we hope to have you work with the live Twitter stream using the development API. However, as of 2014, Twitter requires you to attach a mobile phone number to your account in order to use the developer API. If you do not have mobile phone, or if your carrier is not supported, you may find this workaround useful. However, if you cannot get access to Twitter data due to problems with their login verification protocol, you can instead use the sample dataset in the course github repository called three_minute_tweets.json.
you can install it yourself in your Python environment. (The command $ pip install oauth2 should work for most environments.)
The steps below will help you set up your twitter account to be able to access the live 1% stream.

1. Create a twitter account if you do not already have one.
2. Go to **https://dev.twitter.com/apps** and log in with your twitter credentials.
3. Click "Create New App"
4. Fill out the form and agree to the terms. Put in a dummy website if you don't have one you want to use.
5. At this point you will be prompted to attach a mobile phone number to your account if you have not previously done so. Follow the instructions at the link provided. If you cannot complete this protocol, you may use the sample dataset in the repository, but we encourage you to connect to Twitter if possible so you can build your own applications.
6. On the next page, click the "Keys and Access Tokens" tab along the top, then scroll all the way down until you see the section "Your Access Token"
7. Click the button "Create My Access Token". You can **Read more about Oauth authorization.**
8. You will now copy four values into the file twitterstream.py. These values are your "**Consumer Key (API Key)**", your "**Consumer Secret (API Secret)**", your "**Access token**" and your "**Access token secret**". All four should now be visible on the "Keys and Access Tokens" page. (You may see "Consumer Key (API Key)" referred to as either "Consumer key" or "API Key" in some places in the code or on the web; all three are synonyms.) Open twitterstream.py and set the variables corresponding to the api key, api secret, access token, and access secret. You will see code like the below:

```
api_key = "<Enter api key>"
api_secret = "<Enter api secret>"
access_token_key = "<Enter your access token key here>"
access_token_secret = "<Enter your access token secret here>"
```

Now that your keys are ready, you are ready to test your access.

1. Run the following and make sure you see data flowing and that no errors occur.$ python twitterstream.py > output.txtThis command pipes the output to a file. Stop the program with Ctrl-C, but wait at least **3 minutes** for data to accumulate. Keep the file output.txt for the duration of the assignment; we will be reusing it in later problems. Don't use someone else's file; we will check for uniqueness in other parts of the assignment.
2. If you wish, modify the file to use the twitter search API to search for specific terms. For example, to search for the term "microsoft", you can pass the following url to the twitterreq function: https://api.twitter.com/1.1/search/tweets.json?q=microsoft

Problem 2: Derive the sentiment of each tweet

For this part, you will compute the sentiment of each tweet based on the sentiment scores of the terms in the tweet. The sentiment of a tweet is equivalent to the sum of the sentiment scores for each term in the tweet.
You are provided with a skeleton file tweet_sentiment.py which accepts two arguments on the command line: a *sentiment file* and a tweet file like the one you generated in Problem 1. You can run the skeleton program like this:

```
$ python tweet_sentiment.py AFINN-111.txt output.txt
```

The file AFINN-111.txt contains a list of pre-computed sentiment scores. Each line in the file contains a word or phrase followed by a sentiment score. Each word or phrase that is found in a tweet but not found in AFINN-111.txt should be given a sentiment score of 0. See the file AFINN-README.txt for more information.

To use the data in the AFINN-111.txt file, you may find it useful to build a dictionary. Note that the AFINN-111.txt file format is tab-delimited, meaning that the term and the score are separated by a tab character. A tab character can be identified a "\t".The following snippet may be useful:

```
afinnfile = open("AFINN-111.txt")
scores = {} # initialize an empty dictionary
for line in afinnfile:
  term, score  = line.split("\t")  # The file is tab-delimited. "\t"
     means "tab character"
  scores[term] = int(score)  # Convert the score to an integer.
print scores.items() # Print every (term, score) pair in the
    dictionary
```

The data in the tweet file you generated in Problem 1 is represented as *JSON*, which stands for JavaScript Object Notation. It is a simple format for representing nested structures of data --- lists of lists of dictionaries of lists of .... you get the idea.
Each line of output.txt represents a streaming message. Most, but not all, will be tweets. (The skeleton program will tell you how many lines are in the file.)
It is straightforward to convert a JSON string into a Python data structure; there is a library to do so called json.
To use this library, add the following to the top of tweet_sentiment.py

```
import json
```

Then, to parse the data in output.txt, you want to apply the function json.loads to every line in the file.
This function will parse the json data and return a python data stucture; in this case, it returns a dictionary. If needed, take a moment to **read the documentation for Python dictionaries**.

You can read the Twitter documentation to understand what information each tweet contains and how to access it, but it's not too difficult to deduce the structure by direct inspection.
Your script should print to stdout the sentiment of each tweet in the file, one numeric sentiment score per line. The first score should correspond to the first tweet, the second score should correspond to the second tweet, and so on. If you sort the scores, they won't match up. If you sort the tweets, they won't match up. If you put the tweets into a dictionary, the order will not be preserved. Once again: **The nth line of the file you submit should contain only a single number that represents the score of the nth tweet in the input file!**

NOTE: You must provide a score for **every** tweet in the sample file, even if that score is zero. You can assume the sample file will only include English tweets and no other types of streaming messages.

Hint: This is real-world data, and it can be messy! Refer to the twitter documentation to understand more about the data structure you are working with.

Problem 3: Derive the sentiment of new terms

In this part you will be creating a script that computes the sentiment for the terms that **do not** appear in the file AFINN-111.txt.
Here's how you might think about the problem: We know we can use the sentiment-carrying words in AFINN-111.txt to deduce the overall sentiment of a tweet. Once you deduce the sentiment of a tweet, you can work backwards to deduce the sentiment of the non-sentiment carrying words that do *not* appear in AFINN-111.txt. For example, if the word *soccer* always appears in proximity with positive words like great and fun, then we can deduce that the term *soccer* itself carries a positive sentiment.

Don't feel obligated to use it, but the following paper may be helpful for developing a sentiment metric. Look at the Opinion Estimation subsection of the Text Analysis section in particular.

O'Connor, B., Balasubramanyan, R., Routedge, B., & Smith, N. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. (ICWSM), May 2010.

You are provided with a skeleton file term_sentiment.py which accepts the same two arguments as tweet_sentiment.py and can be executed using the following command:

```
$ python term_sentiment.py AFINN-111.txt output.txt
```

Your script should print output to stdout. Each line of output should contain a term, followed by a space, followed by the sentiment. That is, each line should be in the format <term:string> <sentiment:float>
For example, if you have the pair ("foo", 103.256) in Python, it should appear in the output as:

```
foo 103.256
```

The order of your output does not matter.

How we will grade Part 3: We will run your script on a file that contains strongly positive and strongly negative tweets and verify that the non-sentiment-carrying terms in the strongly positive tweets are assigned a higher score than the non-sentiment-carrying terms in negative tweets. Your scores need not (and likely will not) exactly match any specific solution.