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**](http://www.nytimes.com/2011/09/30/science/30twitter.html) and can even used to [**predict the stock market**](http://arxiv.org/pdf/1010.3003&embedded=true). 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.**](http://cliffmass.blogspot.com/2012/07/thunderstorm-fest.html)

Researchers from Northeastern University and Harvard University studying the characteristics and dynamics of Twitter [**have an excellent resource**](http://www.ccs.neu.edu/home/amislove/twittermood/) for learning more about this area.

In this assignment, you will

* Access the twitter Application Programming Interface 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.

This assignment is open-ended in several ways. You'll need to make some decisions about how best to solve the problem.

It is perfectly acceptable to discuss your solution on the forum, but don't share code.

You may work in groups if you prefer. When you turn in the assignment, cc your group members.

### 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**](https://dev.twitter.com/docs).

### Python environment

This assignment assumes you have a basic familiarity with Python and a functioning Python environment.

If you need help setting up a python environment, post a mesage to the forum and the instructor or one of your classmates will respond.

If you need help with getting up to speed with Python, send the instructor an email.

### 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, mathmatical 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 unicode, you can use the encode method to just replace the international characters, like this:

a=u"aaaàçççñññ" a.encode('ascii','replace') 'aaa???????'

## Problem 0: Query Twitter with Python

Use the urllib and json libraries in python to access the basic twitter search API and return JSON data. The url is:

http://search.twitter.com/search.json?q=microsoft

The format of the result is *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.

As you might imagine, it is fairly straightforward to convert JSON data into a Python data structure. Indeed, there is a convenient library to do so, called json, which we will use.

Twitter provides only [**partial documentation for understanding this data format.**](http://www.google.com/url?q=https%3A%2F%2Fdev.twitter.com%2Fdocs%2Fplatform-objects%2Ftweets&sa=D&sntz=1&usg=AFQjCNHBeFjTVk8gwD6YqU780ssQocg2Rg) But it's not difficult to deduce the structure.

Using this library, the json data is parsed and converted to a Python dictionary representing the entire result set. (If needed, take a moment to [read the documentation for Python dictionaries](http://docs.python.org/2/library/stdtypes.html#typesmapping)). The "results" key of this dictionary corresponds holds the actual tweets; each tweet is another dictionary.

a) Write a program to print out the text of each tweet in the result.

What to turn in: Report the number of tweets returned by this query.

b) Generalize your program to fetch 10 pages of results. Note that you can return a different page of results by passing an additional argument in the url:

http://search.twitter.com/search.json?q=microsoft&page=2

Problem 1: Write a script to calculate the sentiment of a given term

Download and unzip the file [**python\_assignment.tar.gz**](https://sites.google.com/a/cs.washington.edu/uw_datascience_fall2012/schedule/week-6/assignment-6/assignment6.tar.gz?attredirects=0).

**Part A**

To access the live stream, you will need to install the [**oauth2 library**](http://www.google.com/url?q=http%3A%2F%2Fpypi.python.org%2Fpypi%2Foauth2%2F&sa=D&sntz=1&usg=AFQjCNELegpKckAJzHK_NqMdcLt-CHDeOA) so you can properly authenticate.

The steps below will help you set up your twitter account to be able to access the live 1% stream.

* Create a twitter account if you do not already have one.
* Go to [**https://dev.twitter.com/apps**](http://www.google.com/url?q=https%3A%2F%2Fdev.twitter.com%2Fapps&sa=D&sntz=1&usg=AFQjCNGg_XBNmlLBed-qVvrSJzJO2QlOhg) and log in with your twitter credentials.
* Click "create an application"
* Fill out the form and agree to the terms. Put in a dummy website if you don't have one you want to use.
* On the next page, scroll down and click "Create my access token"
* Copy your "Consumer key" and your "Consumer secret" into twitterstream.py
* Click "Create my access token." You can [**Read more about Oauth authorization.**](http://www.google.com/url?q=https%3A%2F%2Fdev.twitter.com%2Fdocs%2Fauth&sa=D&sntz=1&usg=AFQjCNEmpfOUG6OOWl45UGOoVyxab6tF7w)
* Open twitterstream.py and set the variables corresponding to the consumer key, consumer secret, access token, and access secret.
* Run the following and make sure you see data flowing.

$ python twitterstream.py

You can pipe the output to a file, wait a few minutes, then terminate the program to generate a sample. Use the following command:

$ python twitterstream.py > output.txt

Let this script run for a minimum of **10 minutes**.

Save this data 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.

What to turn in: The downloaded data in file output.txt

**Part B**

In this part you will be creating a script that computes the sentiment for a term in a file of tweets.

You are provided with a skeleton file, sent.py, which can be executed using the following command:

$ python sent.py <term> <sentiment\_file> <tweet\_file>

The file AFINN-111.txt contains a list of sentiment scores. Each line in the file contains a word or phrase followed by a score. See the file AFINN-README.txt for more information.

Fill out sent.py by writing a function that takes a term and returns a sentiment score using the sentiment\_file and tweet\_file.

The following paper may be useful for developing a sentiment metric:

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

A simple Python hint that you may or may not need in your solution: If you want to concatenate a whole list of strings by a delimiter, you can use the join method of a string object. Note that you call join as a method of the delimiter you wish to use, not as a method of the list you wish to concatenate.

>>> ",".join(["separated", "by", "commas"]  
"separated,by,commas"  
>>> "".join(["a","b","c"])  
"abc"  
>>> " ".join(["a","b","c"])  
"a b c"

What to turn in: Your file sent.py

Problem 2: Term Frequency

Compute the term frequency histogram of the livestream data you harvested from Problem 1-A.

Write a Python script that provides a function, frequency(<term>), that takes a term and returns the computed frequency.

Frequency measurements may take phrases into account, but this is not required. We only ask that you compute frequencies for individual tokens.

Depending on your method of parsing, you may end up with frequencies for hashtags, links, stop words, phrases, etc. Some skew is acceptable for the sake of keeping parsing simple. We will take your solution as long as we can call frequency on individual English tokens and get numerical values.

What to turn in: a python script that includes a function frequency(<term>) returning a numeric value.

## Problem 3: Which State is happiest?

Write a script that provides a function, happiest\_state(), that returns the name of the happiest state as a string.

We recommend reusing your solution to Problem 1-B and generalizing it to compute sentiments for entire tweets.

Some tweets are geotagged, and you can use this information to sort tweets. However, only a small fraction of the tweets are geotagged. Another method is to use the home state of the user who posted the tweet. There are other ways that may tradeoff accuracy for simplicity. Anything is fine, but be prepared to explain your decision.

The live stream has a slightly different format from the response to the query you used in Problem 0. In this file, each line is a Tweet object, as [**described in the twitter documentation**](https://dev.twitter.com/docs/platform-objects/tweets).

Note: Not every tweet dictionary will have a text key -- real data is dirty. Be prepared to debug, and feel free to throw out tweets that your code can't handle to get something working. For example, non-English tweets.

What to turn in: a python script that includes a function happiest\_state(<term>) returning a numeric value.

Problem 4: Top ten hash tags

Write a script that provides a function, top\_ten\_hash\_tags(), that computes the ten most frequently occurring hash tags from the data you gathered in Problem 1-A.

top\_ten\_hash\_tags() should return a dictionary associating hash tag strings to numerical frequencies.

The hash tag string keys should be formatted as follows:

#<hashtag>

The resulting dictionary should contain at most ten elements, and order ties may be broken as you see fit.

What to turn in: a python script that includes a function happiest\_state(<term>) returning a numeric value. Note that the same script can be used for Problem 2,3 and 4.

Problem 5: Visualization

Compute the number of tweets per minute and plot the results using R, matplotlib and Python, or even Excel.

This simple visualization helps convey the average and the variance of rate of tweets

This problem will be graded by peer assessment.

What to turn in: an image file of your timeseries visualization.