Twitter HOWTO

Overview

This document is an overview of how to use NLTK to collect and process Twitter data. It was written as an IPython notebook, and if you have IPython installed, you can download the source of the notebook (https://raw.githubusercontent.com/nltk/nltk/develop/nltk/test/twitter.ipynb) from the NLTK GitHub repository and run the notebook in interactive mode.

Most of the tasks that you might want to carry out with 'live' Twitter data require you to authenticate your request by registering for API keys. This is usually a once-only step. When you have registered your API keys, you can store them in a file on your computer, and then use them whenever you want. We explain what's involved in the section First Steps.

If you have already obtained Twitter API keys as part of some earlier project, storing your keys explains how to save them to a file that NLTK will be able to find. Alternatively, if you just want to play around with the Twitter data that is distributed as part of NLTK, head over to the section on using the `twitter-samples` corpus reader.

Once you have got authentication sorted out, we'll show you how to use NLTK's `Twitter` class. This is made as simple as possible, but deliberately limits what you can do.

First Steps

As mentioned above, in order to collect data from Twitter, you first need to register a new *application* — this is Twitter's way of referring to any computer program that interacts with the Twitter API. As long as you save your registration information correctly, you should only need to do this once, since the information should work for any NLTK code that you write. You will need to have a Twitter account before you can register. Twitter also insists that you add a mobile phone number to your Twitter profile (https://support.twitter.com/articles/110250-adding-your-mobile-number-to-your-account-via-web) before you will be allowed to register an application.

These are the steps you need to carry out.

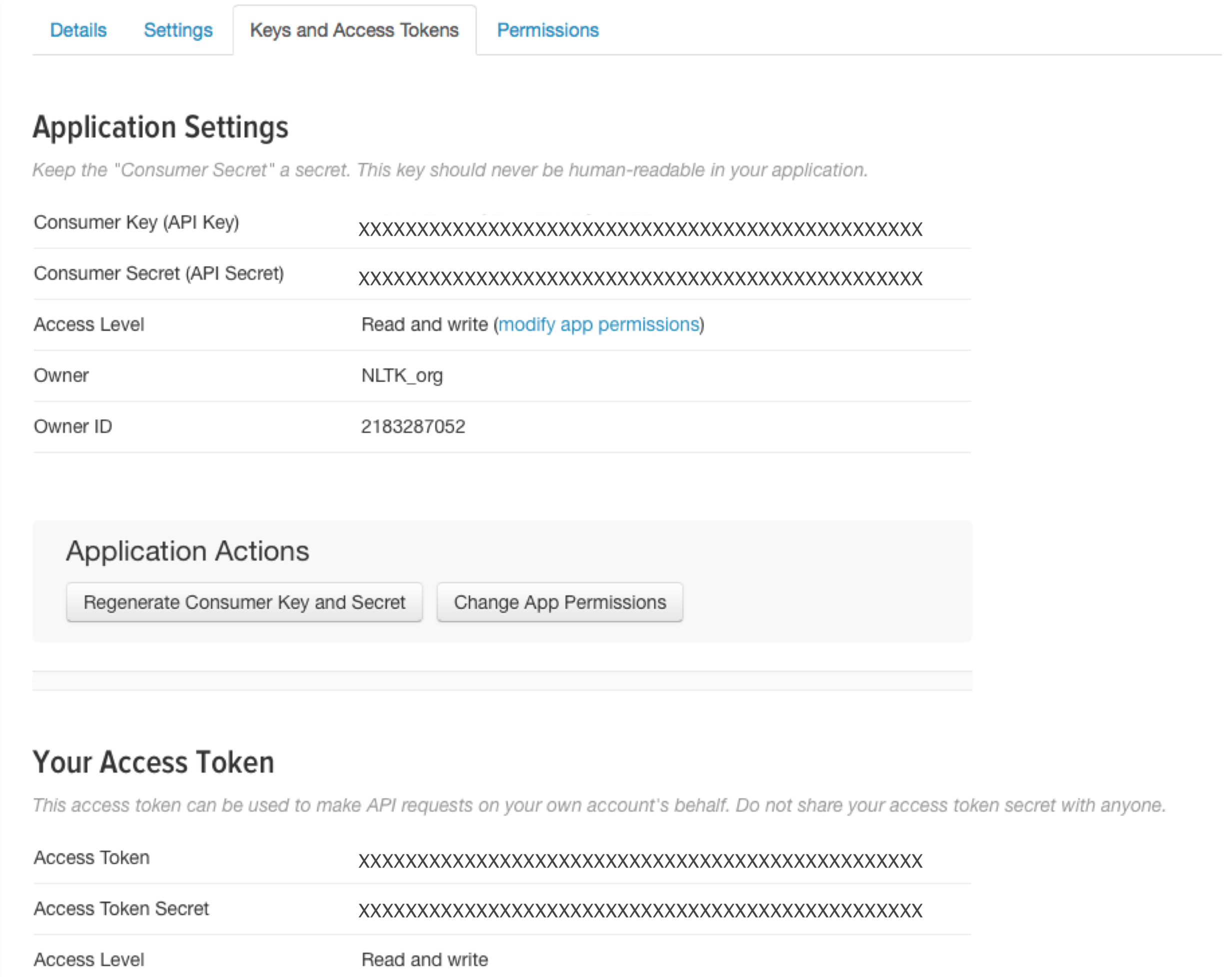
Getting your API keys from Twitter

1. Sign in to your Twitter account at https://apps.twitter.com (https://apps.twitter.com). You should then get sent to a screen that looks something like this:

   Clicking on the **Create New App** button should take you to the following screen:

   The information that you provide for **Name**, **Description** and **Website** can be anything you like.
2. Make sure that you select **Read and Write** access for your application (as specified on the *Permissions* tab of Twitter's Application Management screen):

3. Go to the tab labeled **Keys and Access Tokens**. It should look something like this, but with actual keys rather than a string of Xs:



Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Consumer Secret (API Secret)	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Access Level	Read and write (modify app permissions)
Owner	NLTK_org
Owner ID	2183287052

Application Actions

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Access Token Secret	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Access Level	Read and write

As you can see, this will give you four distinct keys: consumer key, consumer key secret, access token and access token secret.

Storing your keys

1. Create a folder named `twitter-files` in your home directory. Within this folder, use a text editor to create a new file called `credentials.txt`. Make sure that this file is just a plain text file. In it, you should create which you should store in a text file with the following structure:

   ```
   app_key=YOUR CONSUMER KEY
   app_secret=YOUR CONSUMER SECRET
   oauth_token=YOUR ACCESS TOKEN
   oauth_token_secret=YOUR ACCESS TOKEN SECRET
   ```

   Type the part up to and includinge the '=' symbol exactly as shown. The values on the right-hand side of the '=' — that is, everything in caps — should be cut-and-pasted from the relevant API key information shown on the Twitter **Keys and Access Tokens**. Save the file and that's it.
2. It's going to be important for NLTK programs to know where you have stored your credentials. We'll assume that this folder is called `twitter-files`, but you can call it anything you like. We will also assume that this folder is where you save any files containing tweets that you collect. Once you have decided on the name and location of this folder, you will need to set the `TWITTER` environment variable to this value.

   On a Unix-like system (including MacOS), you will set the variable something like this:

```
export TWITTER="/path/to/your/twitter-files"
```

Rather than having to give this command each time you start a new session, it's advisable to add it to your shell's configuration file, e.g. to `.bashrc`.

On a Windows machine, right click on “My Computer” then select `Properties > Advanced > Environment Variables > User Variables > New...`

One important thing to remember is that you need to keep your `credentials.txt` file private. So do **not** share your `twitter-files` folder with anyone else, and do **not** upload it to a public repository such as GitHub.

3. Finally, read through Twitter's Developer Rules of the Road (https://dev.twitter.com/overview/terms/policy). As far as these rules are concerned, you count as both the application developer and the user.

Install Twython

The NLTK Twitter package relies on a third party library called Twython (https://twython.readthedocs.org/). Install Twython via pip (https://pip.pypa.io):

```
$ pip install twython
```

or with easy_install (https://pythonhosted.org/setuptools/easy_install.html):

```
$ easy_install twython
```

We're now ready to get started. The next section will describe how to use the `Twitter` class to talk to the Twitter API.

More detail: Twitter offers are two main authentication options. OAuth 1 is for user-authenticated API calls, and allows sending status updates, direct messages, etc, whereas OAuth 2 is for application-authenticated calls, where read-only access is sufficient. Although OAuth 2 sounds more appropriate for the kind of tasks envisaged within NLTK, it turns out that access to Twitter's Streaming API requires OAuth 1, which is why it's necessary to obtain *Read and Write* access for your application.

Using the simple `Twitter` class

Dipping into the Public Stream

The `Twitter` class is intended as a simple means of interacting with the Twitter data stream. Later on, we'll look at other methods which give more fine-grained control.

The Twitter live public stream is a sample (approximately 1%) of all Tweets that are currently being published by users. They can be on any topic and in any language. In your request, you can give keywords which will narrow down the Tweets that get delivered to you. Our first example looks for Tweets which include either the word *love* or *hate*. We limit the call to finding 10 tweets. When you run this code, it will definitely produce different results from those shown below!

In [2]:

```
from nltk.twitter import Twitter
tw = Twitter()
tw.tweets(keywords='love, hate', limit=10) #sample from the public stream
```

```
Sana magkakaisa na ang mga Kapamilya at Kapuso.  Spread love, not hate
 #ShowtimeKapamiIyaDay #ALDubEBforLOVE
@Real_Liam_Payne Please follow me , you mean the world to me and words can't
describe how much i love you x3186
Love my ugly wife
RT @ansaberano: We Found Love
#PushAwardsLizQuen
RT @yungunmei: people want to fall in love but don't understand the concept
I don't care, I love It  #EMABiggestFans1D
RT @bryan_white: I'm not in the Philippines Yet but we are making a very BIG
announcement in 2 days! Get ready! Love you! #GGMY #ALDubEBfor…
I whole heartedly HATE @lakiamichelle like really HATE her 😩 who wants to be
her friend because I DONT
RT @lahrose23: I love yu to  https://t.co/dfsRwSp1IC
RT @alone_in_woods: ahoj, já jsem tvůj pes a tohle je náš love song /// Zrní
- Já jsem tvůj pes https://t.co/7L0XPHeA2d via @YouTube
Written 10 Tweets
```

The next example filters the live public stream by looking for specific user accounts. In this case, we 'follow' two news organisations, namely `@CNN` and `@BBCNews`. As advised by Twitter (https://dev.twitter.com/streaming/reference/post/statuses/filter), we use *numeric userIDs* for these accounts. If you run this code yourself, you'll see that Tweets are arriving much more slowly than in the previous example. This is because even big new organisations don't publish Tweets that often.

A bit later we will show you how to use Python to convert usernames such as `@CNN` to userIDs such as `759251`, but for now you might find it simpler to use a web service like TweeterID (http://tweeterid.com) if you want to experiment with following different accounts than the ones shown below.

```
In [3]: tw = Twitter()
        tw.tweets(follow=['759251', '612473'], limit=10) # see what CNN and BBC are talking about
```

```
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
RT @CNN: Judge grants petition allowing @Caitlyn_Jenner to officially change
her name and gender. http://t.co/HpCbAQ64Mk http://t.co/BPaKy2…
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
Written 10 Tweets
```

Saving Tweets to a File

By default, the `Twitter` class will just print out Tweets to your computer terminal. Although it's fun to view the Twitter stream zipping by on your screen, you'll probably want to save some tweets in a file. We can tell the `tweets()` method to save to a file by setting the flag `to_screen` to `False`.

The `Twitter` class will look at the value of your environmental variable `TWITTER` to determine which folder to use to save the tweets, and it will put them in a date-stamped file with the prefix `tweets`.

```
In [4]: tw = Twitter()
        tw.tweets(to_screen=False, limit=25)
```

```
Writing to /Users/ewan/twitter-files/tweets.20150926-154251.json
Written 25 Tweets
```

So far, we've been taking data from the live public stream. However, it's also possible to retrieve past tweets, for example by searching for specific keywords, and setting `stream=False`:

```
In [5]: tw.tweets(keywords='hilary clinton', stream=False, limit=10)
```

```
“Girls” Creator Lena Dunham Interviews Hilary Clinton About… Lenny Kravitz’s
Junk [Video] http://t.co/eY4GgKS3ak
“Girls” Creator Lena Dunham Interviews Hilary Clinton About… Lenny Kravitz’s
Junk [Video] http://t.co/Pflf7A6Tr6
“Girls” Creator Lena Dunham Interviews Hilary Clinton About… Lenny Kravitz’s
Junk [Video] http://t.co/mibYfNISBT http://t.co/9ElX70F4St
Photo: “Girls” Creator Lena Dunham Interviews Hilary Clinton About… Lenny Kra
vitz’s Junk [Video]: Hillary... http://t.co/qIiWGk1jbM
lena dunham and hilary clinton talking about feminism... l o l theyre the two
most hypocritical and clueless about what feminism actually is
“Girls” Creator Lena Dunham Interviews Hilary Clinton About… Lenny Kravitz’s
Junk [Video]:
Hillary Clinton An... http://t.co/31shf6VeEu
“Girls” Creator Lena Dunham Interviews Hilary Clinton About… Lenny Kravitz’s
Junk [Video]:
Hillary Clinton An... http://t.co/uvft4LDS0t
“Girls” Creator Lena Dunham Interviews Hilary Clinton About… Lenny Kravitz’s
Junk [Video]:
Hillary Clinton An... http://t.co/uEbc25V3E3
“Girls” Creator Lena Dunham Interviews Hilary Clinton About… Lenny Kravitz’s
Junk [Video]:
Hillary Cl... http://t.co/RNgziN9eWA #bossip
“Girls” Creator Lena Dunham Interviews Hilary Clinton About… Lenny Kravitz’s
Junk [Video]:
Hillary Clinton An... http://t.co/gkB5aLEJJP
Written 10 Tweets
```

Onwards and Upwards

In this section, we'll look at how to get more fine-grained control over processing Tweets. To start off, we will import a bunch of stuff from the `twitter` package.

```
In [6]: from nltk.twitter import Query, Streamer, Twitter, TweetViewer, TweetWriter, c
        redsfromfile
```

In the following example, you'll see the line

```
oauth = credsfromfile()
```

This gets hold of your stored API key information. The function `credsfromfile()` by default looks for a file called `credentials.txt` in the directory set by the environment variable `TWITTER`, reads the contents and returns the result as a dictionary. We then pass this dictionary as an argument when initializing our client code. We'll be using two classes to wrap the clients: `Streamer` and `Query`; the first of these calls the Streaming API (https://dev.twitter.com/streaming/overview) and the second calls Twitter's Search API (https://dev.twitter.com/rest/public) (also called the REST API).

More detail: For more detail, see this blog post on The difference between the Twitter Firehose API, the Twitter Search API, and the Twitter Streaming API (http://www.brightplanet.com/2013/06/twitter-firehose-vs-twitter-api-whats-the-difference-and-why-should-you-care/)

After initializing a client, we call the `register()` method to specify whether we want to view the data on a terminal or write it to a file. Finally, we call a method which determines the API endpoint to address; in this case, we use `sample()` to get a random sample from the the Streaming API.

In [27]:
```
oauth = credsfromfile()
client = Streamer(**oauth)
client.register(TweetViewer(limit=10))
client.sample()
```

```
RT @EPVLatino: ¿Y entonces? El gobierno sigue importando carros mientras las
plantas Chery los tiene acumulados http://t.co/bBhrawqHe7
RT @AbrahamMateoMus: . @menudanochetv aquí se suman nuestros Abrahamers MEXIC
AN@S!! 👏
#MenudaNocheConAM 😉 http://t.co/8DMw31wZ5i
RT @Joeyclipstar: ** FRESH ** Bow Wow Signs to Bad Boy Records - The Breakfas
t Club http://t.co/3w58p6Sbx2 RT http://t.co/LbQU2brfpf
#شاركونا
اشي مستحيل تكمل يومك بدونه ... ؟ 🌚
#Manal
RT @techjunkiejh: MEAN Stack Tutorial #mongodb #ExpressJS #angularjs #nodejs
#javascript http://t.co/4gTFsj2dtP http://t.co/a86hmb4mRx
Only @MariamDiamond would reply to a spider on twitter 😂😂
RT @CJLeBlanc: @SeanCarrigan greets the new day..full spirit, verve and no sm
all amount of vodka! GO TEAM #YR! … http://t.co/bQIglZVDxR
んぐぅおぉ、はらみーライブ楽しかったようで何より。行きたかったンゴ〜
RT @NicoleRaine8: @maine_richards @MaLuisaMiranda1 count me in ngkakape nyaha
ha #ALDubEBforLOVE
RT @RadioDelPlata: [AHORA] "Me amputaron los 4 miembros" Perla Pascarelli sob
re #Malapraxis a #MónicayCésar http://t.co/StUhpxDeM3
Written 10 Tweets
```

The next example is similar, except that we call the `filter()` method with the `track` parameter followed by a string literal. The string is interpreted as a list of search terms where comma indicates a logical OR (https://dev.twitter.com/streaming/overview/request-parameters#track). The terms are treated as case-insensitive.

```
In [30]: client = Streamer(**oauth)
         client.register(TweetViewer(limit=10))
         client.filter(track='refugee, germany')
```

```
European countries at heart of refugee crisis seek to ease tensions: Hungary
announces removal of razor wire f... http://t.co/PavCKddtY2
RT @onlinewweman: Germany told students to wear "modest clothing" bc they do
n't want the refugees to have "misunderstandings." That's a wei…
RT @El_consciente: El cuento ha cambiado. A pinocho le crecía la nariz si men
tía. A los políticos europeos sus fortunas. Made in Germany ht…
VIDEO=&gt; Finns Attack “Refugee” Bus with Rocks and Fireworks – Refugees Tur
n Back to Sweden https://t.co/94KqhyCNjJ http://t.co/e3kmeGjRFn
RT @El_consciente: Merkel al volante de Europa. Fabricación en cadena de prod
uctos fraudulentos. Made in Germany http://t.co/SJ5BYQ7lIu htt…
European countries at heart of refugee crisis seek to ease tensions: Hungary
announces rem... http://t.co/5BmOYNK3Kj (via @EricBarbosa11
@SirCorgis @matty_is @RT_com but will Poland blame the ppl actually causing t
he refugee crisis? Cause and effect is a bitch innit?
RT @El_consciente: Merkel al volante de Europa. Fabricación en cadena de prod
uctos fraudulentos. Made in Germany http://t.co/SJ5BYQ7lIu htt…
♥ https://t.co/CyoWdON0li
RT @mjesusgz: Castle Germany http://t.co/scs5dJE1Gk
Written 10 Tweets
```

Whereas the Streaming API lets us access near real-time Twitter data, the Search API lets us query for past Tweets. In the following example, the value `tweets` returned by `search_tweets()` is a generator; the expression `next(tweets)` gives us the first Tweet from the generator.

Although Twitter delivers Tweets as JSON (http://www.json.org) objects, the Python client encodes them as dictionaries, and the example pretty-prints a portion of the dictionary corresponding the Tweet in question.

```
In [31]: client = Query(**oauth)
         tweets = client.search_tweets(keywords='nltk', limit=10)
         tweet = next(tweets)
         from pprint import pprint
         pprint(tweet, depth=1)
```

```
{'contributors': None,
 'coordinates': None,
 'created_at': 'Sat Sep 26 14:25:12 +0000 2015',
 'entities': {...},
 'favorite_count': 0,
 'favorited': False,
 'geo': None,
 'id': 647778955005665280,
 'id_str': '647778955005665280',
 'in_reply_to_screen_name': None,
 'in_reply_to_status_id': None,
 'in_reply_to_status_id_str': None,
 'in_reply_to_user_id': None,
 'in_reply_to_user_id_str': None,
 'is_quote_status': False,
 'lang': 'en',
 'metadata': {...},
 'place': None,
 'possibly_sensitive': False,
 'retweet_count': 0,
 'retweeted': False,
 'source': '<a href="http://www.techwars.io" rel="nofollow">TechWars</a>',
 'text': 'We compared #gate vs #nltk - see results: http://t.co/jvQ4Ph85L1',
 'truncated': False,
 'user': {...}}
```

Twitter's own documentation provides a useful overview of all the fields in the JSON object (https://dev.twitter.com/overview/api/tweets) and it may be helpful to look at this visual map of a Tweet object (http://www.scribd.com/doc/30146338/map-of-a-tweet).

Since each Tweet is converted into a Python dictionary, it's straightforward to just show a selected field, such as the value of the `'text'` key.

```
In [32]: for tweet in tweets:
             print(tweet['text'])
```

```
Slammer an immigration lawyer seattle wa protection if purusha this morning p
olaric deportation?: Nltk
Python Text Processing with NLTK 2.0 Cookbook / Jacob Perkins
http://t.co/0gUjlTWA7G

49
RT @tjowens: DHbox http://t.co/skIzU3Nm6C "Ready-to-go configurations of Omek
a, NLTK, IPython, R Studio, and Mallet" #odh2015 http://t.co/6…
RT @tjowens: DHbox http://t.co/skIzU3Nm6C "Ready-to-go configurations of Omek
a, NLTK, IPython, R Studio, and Mallet" #odh2015 http://t.co/6…
RT @tjowens: DHbox http://t.co/skIzU3Nm6C "Ready-to-go configurations of Omek
a, NLTK, IPython, R Studio, and Mallet" #odh2015 http://t.co/6…
RT @tjowens: DHbox http://t.co/skIzU3Nm6C "Ready-to-go configurations of Omek
a, NLTK, IPython, R Studio, and Mallet" #odh2015 http://t.co/6…
RT @ideaofhappiness: Interesting! @DH_Box is a Docker container for digital h
umanities computational work, pre-equipped with IPython, RStud…
RT @ideaofhappiness: Interesting! @DH_Box is a Docker container for digital h
umanities computational work, pre-equipped with IPython, RStud…
RT @dimazest: Stanford dependency parser support is merged into @NLTK_org htt
ps://t.co/aN6b1lFGPf
```

```
In [11]: client = Query(**oauth)
         client.register(TweetWriter())
         client.user_tweets('timoreilly', 10)
```

```
Writing to /Users/ewan/twitter-files/tweets.20150926-154337.json
```

Given a list of user IDs, the following example shows how to retrieve the screen name and other information about the users.

```
In [12]: userids = ['759251', '612473', '15108702', '6017542', '2673523800']
         client = Query(**oauth)
         user_info = client.user_info_from_id(userids)
         for info in user_info:
             name = info['screen_name']
             followers = info['followers_count']
             following = info['friends_count']
             print("{}, followers: {}, following: {}".format(name, followers, following
         ))
```

```
CNN, followers: 19806095, following: 1102
BBCNews, followers: 4935491, following: 105
ReutersLive, followers: 307337, following: 55
BreakingNews, followers: 7949242, following: 541
AJELive, followers: 1117, following: 19
```

A list of user IDs can also be used as input to the Streaming API client.

```
In [13]: client = Streamer(**oauth)
         client.register(TweetViewer(limit=10))
         client.statuses.filter(follow=userids)
```

```
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
RT @bbcweather: Cameras at the ready for #supermoon #eclipse on 27/28th Sept,
next one won't be until 2033! http://t.co/SPucnmBqaD http://t…
RT @BreakingNews: Alleged Libya-Europe people smuggler killed in shootout, Li
bya officials say Italy behind asssassination - @guardian http…
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
@CNN white water, Monica, emails, Benghazi. A family/foundation of lies and c
rime. Indict Hillary for breaking laws
RT @CNN: Bill Clinton on email scrutiny: 'I've never seen so much expended on
so little.'
http://t.co/XkLP0IHeOG
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
RT @CNN: Sunday's #supermoon #eclipse has some excited, but for others, it's
an ominous "blood moon." http://t.co/2B1wdQru0q http://t.co/Aw…
RT @BreakingNews: Alleged Libya-Europe people smuggler killed in shootout, Li
bya officials say Italy behind asssassination - @guardian http…
Written 10 Tweets
```

To store data that Twitter sents by the Streaming API, we register a `TweetWriter` instance.

```
In [14]: client = Streamer(**oauth)
         client.register(TweetWriter(limit=10))
         client.statuses.sample()
```

```
Writing to /Users/ewan/twitter-files/tweets.20150926-154408.json
Written 10 Tweets
```

Here's the full signature of the `Tweetwriter`'s `__init__()` method:

```
def __init__(self, limit=2000, upper_date_limit=None, lower_date_limit=None,
                 fprefix='tweets', subdir='twitter-files', repeat=False,
                 gzip_compress=False):
```

If the `repeat` parameter is set to `True`, then the writer will write up to the value of `limit` in file `file1`, then open a new file `file2` and write to it until the limit is reached, and so on indefinitely. The parameter `gzip_compress` can be used to compress the files once they have been written.

Using a Tweet Corpus

NLTK's Twitter corpus currently contains a sample of 20k Tweets (named '`twitter_samples`') retrieved from the Twitter Streaming API, together with another 10k which are divided according to sentiment into negative and positive.

```
In [15]: from nltk.corpus import twitter_samples
         twitter_samples.fileids()
```

```
Out[15]: ['negative_tweets.json', 'positive_tweets.json', 'tweets.20150430-223406.jso
         n']
```

We follow standard practice in storing full Tweets as line-separated JSON. These data structures can be accessed via `tweets.docs()`. However, in general it is more practical to focus just on the text field of the Tweets, which are accessed via the `strings()` method.

```
In [16]: strings = twitter_samples.strings('tweets.20150430-223406.json')
         for string in strings[:15]:
             print(string)
```

```
RT @KirkKus: Indirect cost of the UK being in the EU is estimated to be costi
ng Britain £170 billion per year! #BetterOffOut #UKIP
VIDEO: Sturgeon on post-election deals http://t.co/BTJwrpbmOY
RT @LabourEoin: The economy was growing 3 times faster on the day David Camer
on became Prime Minister than it is today.. #BBCqt http://t.co…
RT @GregLauder: the UKIP east lothian candidate looks about 16 and still has
an msn addy http://t.co/7eIU0c5Fm1
RT @thesundaypeople: UKIP's housing spokesman rakes in £800k in housing benef
it from migrants.  http://t.co/GVwb9Rcb4w http://t.co/c1AZxcLh…
RT @Nigel_Farage: Make sure you tune in to #AskNigelFarage tonight on BBC 1 a
t 22:50! #UKIP http://t.co/ogHSc2Rsr2
RT @joannetallis: Ed Milliband is an embarrassment. Would you want him repres
enting the UK?!  #bbcqt vote @Conservatives
RT @abstex: The FT is backing the Tories. On an unrelated note, here's a phot
o of FT leader writer Jonathan Ford (next to Boris) http://t.c…
RT @NivenJ1: “@George_Osborne: Ed Miliband proved tonight why he's not up to
the job” Tbf you've spent 5 years doing that you salivating do…
LOLZ to Trickle Down Wealth. It's never trickling past their own wallets. Gre
ed always wins $$$ for the greedy.  https://t.co/X7deoPbS97
SNP leader faces audience questions http://t.co/TYClKltSpW
RT @cononeilluk: Cameron "Ed Milliband hanging out with Russell Brand. He is
a joke. This is an election. This is about real people' http:/…
RT @politicshome: Ed Miliband: Last Labour government did not overspend htt
p://t.co/W9RJ2aSH6o http://t.co/4myFekg5ex
If Miliband is refusing to do any deal with the SNP, how does he plan on form
ing a government?
RT @scotnotbritt: Well thats it. LABOUR would rather have a TORY government r
ather than work with the SNP. http://t.co/SNMkRDCe9f
```

The default tokenizer for Tweets (`casual.py`) is specialised for 'casual' text, and the `tokenized()` method returns a list of lists of tokens.

```
In [17]: tokenized = twitter_samples.tokenized('tweets.20150430-223406.json')
         for toks in tokenized[:5]:
             print(toks)
```

```
['RT', '@KirkKus', ':', 'Indirect', 'cost', 'of', 'the', 'UK', 'being', 'in',
'the', 'EU', 'is', 'estimated', 'to', 'be', 'costing', 'Britain', '£', '170',
'billion', 'per', 'year', '!', '#BetterOffOut', '#UKIP']
['VIDEO', ':', 'Sturgeon', 'on', 'post-election', 'deals', 'http://t.co/BTJwr
pbmOY']
['RT', '@LabourEoin', ':', 'The', 'economy', 'was', 'growing', '3', 'times',
'faster', 'on', 'the', 'day', 'David', 'Cameron', 'became', 'Prime', 'Ministe
r', 'than', 'it', 'is', 'today', '..', '#BBCqt', 'http://t.co…']
['RT', '@GregLauder', ':', 'the', 'UKIP', 'east', 'lothian', 'candidate', 'lo
oks', 'about', '16', 'and', 'still', 'has', 'an', 'msn', 'addy', 'http://t.c
o/7eIU0c5Fm1']
['RT', '@thesundaypeople', ':', "UKIP's", 'housing', 'spokesman', 'rakes', 'i
n', '£', '800k', 'in', 'housing', 'benefit', 'from', 'migrants', '.', 'htt
p://t.co/GVwb9Rcb4w', 'http://t.co/c1AZxcLh…']
```

Extracting Parts of a Tweet

If we want to carry out other kinds of analysis on Tweets, we have to work directly with the file rather than via the corpus reader. For demonstration purposes, we will use the same file as the one in the preceding section, namely `tweets.20150430-223406.json`. The `abspath()` method of the corpus gives us the full pathname of the relevant file. If your NLTK data is installed in the default location on a Unix-like system, this pathname will be `'/usr/share/nltk_data/corpora/twitter_samples/tweets.20150430-223406.json'`.

```
In [18]: from nltk.corpus import twitter_samples
         input_file = twitter_samples.abspath("tweets.20150430-223406.json")
```

The function `json2csv()` takes as input a file-like object consisting of Tweets as line-delimited JSON objects and returns a file in CSV format. The third parameter of the function lists the fields that we want to extract from the JSON. One of the simplest examples is to extract just the text of the Tweets (though of course it would have been even simpler to use the `strings()` method of the corpus reader).

```
In [19]: from nltk.twitter.util import json2csv
         with open(input_file) as fp:
             json2csv(fp, 'tweets_text.csv', ['text'])
```

We've passed the filename `'tweets_text.csv'` as the second argument of `json2csv()`. Unless you provide a complete pathname, the file will be created in the directory where you are currently executing Python.

If you open the file `'tweets_text.csv'`, the first 5 lines should look as follows:

```
RT @KirkKus: Indirect cost of the UK being in the EU is estimated to be costing Bri
tain £170 billion per year! #BetterOffOut #UKIP
VIDEO: Sturgeon on post-election deals http://t.co/BTJwrpbmOY
RT @LabourEoin: The economy was growing 3 times faster on the day David Cameron bec
ame Prime Minister than it is today.. #BBCqt http://t.co…
RT @GregLauder: the UKIP east lothian candidate looks about 16 and still has an msn
addy http://t.co/7eIU0c5Fm1
RT @thesundaypeople: UKIP's housing spokesman rakes in £800k in housing benefit fro
m migrants.  http://t.co/GVwb9Rcb4w http://t.co/c1AZxcLh…
```

However, in some applications you may want to work with Tweet metadata, e.g., the creation date and the user. As mentioned earlier, all the fields of a Tweet object are described in the official Twitter API (https://dev.twitter.com/overview/api/tweets).

The third argument of `json2csv()` can specified so that the function selects relevant parts of the metadata. For example, the following will generate a CSV file including most of the metadata together with the id of the user who has published it.

```
In [20]: with open(input_file) as fp:
             json2csv(fp, 'tweets.20150430-223406.tweet.csv',
                     ['created_at', 'favorite_count', 'id', 'in_reply_to_status_id',
                     'in_reply_to_user_id', 'retweet_count', 'retweeted',
                     'text', 'truncated', 'user.id'])
```

```
In [21]: for line in open('tweets.20150430-223406.tweet.csv').readlines()[:5]:
             print(line)
```

```
created_at,favorite_count,id,in_reply_to_status_id,in_reply_to_user_id,retwee
t_count,retweeted,text,truncated,user.id

Thu Apr 30 21:34:06 +0000 2015,0,593891099434983425,,,0,False,RT @KirkKus: In
direct cost of the UK being in the EU is estimated to be costing Britain £170
billion per year! #BetterOffOut #UKIP,False,107794703

Thu Apr 30 21:34:06 +0000 2015,0,593891099548094465,,,0,False,VIDEO: Sturgeon
on post-election deals http://t.co/BTJwrpbmOY,False,557422508

Thu Apr 30 21:34:06 +0000 2015,0,593891099388846080,,,0,False,RT @LabourEoin:
The economy was growing 3 times faster on the day David Cameron became Prime
Minister than it is today.. #BBCqt http://t.co…,False,3006692193

Thu Apr 30 21:34:06 +0000 2015,0,593891100429045760,,,0,False,RT @GregLauder:
the UKIP east lothian candidate looks about 16 and still has an msn addy htt
p://t.co/7eIU0c5Fm1,False,455154030
```

The first nine elements of the list are attributes of the Tweet, while the last one, `user.id`, takes the user object associated with the Tweet, and retrieves the attributes in the list (in this case only the id). The object for the Twitter user is described in the Twitter API for users (https://dev.twitter.com/overview/api/users).

The rest of the metadata of the Tweet are the so-called entities (https://dev.twitter.com/overview/api/entities) and places (https://dev.twitter.com/overview/api/places). The following examples show how to get each of those entities. They all include the id of the Tweet as the first argument, and some of them include also the text for clarity.

In [22]:

```
from nltk.twitter.util import json2csv_entities
with open(input_file) as fp:
    json2csv_entities(fp, 'tweets.20150430-223406.hashtags.csv',
                        ['id', 'text'], 'hashtags', ['text'])

with open(input_file) as fp:
    json2csv_entities(fp, 'tweets.20150430-223406.user_mentions.csv',
                        ['id', 'text'], 'user_mentions', ['id', 'screen_name'
])

with open(input_file) as fp:
    json2csv_entities(fp, 'tweets.20150430-223406.media.csv',
                        ['id'], 'media', ['media_url', 'url'])

with open(input_file) as fp:
    json2csv_entities(fp, 'tweets.20150430-223406.urls.csv',
                        ['id'], 'urls', ['url', 'expanded_url'])

with open(input_file) as fp:
    json2csv_entities(fp, 'tweets.20150430-223406.place.csv',
                        ['id', 'text'], 'place', ['name', 'country'])

with open(input_file) as fp:
    json2csv_entities(fp, 'tweets.20150430-223406.place_bounding_box.csv',
                        ['id', 'name'], 'place.bounding_box', ['coordinates'])
```

Additionally, when a Tweet is actually a retweet, the original tweet can be also fetched from the same file, as follows:

In [23]:

```
with open(input_file) as fp:
    json2csv_entities(fp, 'tweets.20150430-223406.original_tweets.csv',
                        ['id'], 'retweeted_status', ['created_at', 'favorite_c
ount',
                        'id', 'in_reply_to_status_id', 'in_reply_to_user_id',
'retweet_count',
                        'text', 'truncated', 'user.id'])
```

Here the first id corresponds to the retweeted Tweet, and the second id to the original Tweet.

Using Dataframes

Sometimes it's convenient to manipulate CSV files as tabular data, and this is made easy with the Pandas (http://pandas.pydata.org/) data analysis library. `pandas` is not currrently one of the dependencies of NLTK, and you will probably have to install it specially.

Here is an example of how to read a CSV file into a `pandas` dataframe. We use the `head()` method of a dataframe to just show the first 5 rows.

In [24]:
```
import pandas as pd
tweets = pd.read_csv('tweets.20150430-223406.tweet.csv', index_col=2, header=0
, encoding="utf8")
tweets.head(5)
```

Out[24]:

	created_at	favorite_count	in_reply_to_status_id	in_reply_to_user
id				
593891099434983425	Thu Apr 30 21:34:06 +0000 2015	0	NaN	NaN
593891099548094465	Thu Apr 30 21:34:06 +0000 2015	0	NaN	NaN
593891099388846080	Thu Apr 30 21:34:06 +0000 2015	0	NaN	NaN
593891100429045760	Thu Apr 30 21:34:06 +0000 2015	0	NaN	NaN
593891100768784384	Thu Apr 30 21:34:07 +0000 2015	0	NaN	NaN

Using the dataframe it is easy, for example, to first select Tweets with a specific user ID and then retrieve their `'text'` value.

In [25]:
```
tweets.loc[tweets['user.id'] == 557422508]['text']
```

Out[25]:
```
id
593891099548094465    VIDEO: Sturgeon on post-election deals http://...
593891101766918144    SNP leader faces audience questions http://t.c...
Name: text, dtype: object
```

Expanding a list of Tweet IDs

Because the Twitter Terms of Service place severe restrictions on the distribution of Tweets by third parties, a workaround is to instead distribute just the Tweet IDs, which are not subject to the same restrictions. The method `expand_tweetids()` sends a request to the Twitter API to return the full Tweet (in Twitter's terminology, a *hydrated* Tweet) that corresponds to a given Tweet ID.

Since Tweets can be deleted by users, it's possible that certain IDs will only retrieve a null value. For this reason, it's safest to use a `try`/`except` block when retrieving values from the fetched Tweet.

In [26]:

```
from nltk.compat import StringIO
ids_f =\
    StringIO("""\
    588665495492124672
    588665495487909888
    588665495508766721
    588665495513006080
    588665495517200384
    588665495487811584
    588665495525588992
    588665495487844352
    88665495492014081
    588665495512948737""")

oauth = credsfromfile()
client = Query(**oauth)
hydrated = client.expand_tweetids(ids_f)


for tweet in hydrated:
        id_str = tweet['id_str']
        print('id: {}'.format(id_str))
        text = tweet['text']
        if text.startswith('@null'):
            text = "[Tweet not available]"
        print(text + '\n')
```

```
Counted 10 Tweet IDs in <_io.StringIO object at 0x107234558>.
id: 588665495508766721
RT @30SecFlghts: Yep it was bad from the jump https://t.co/6vsFIulyRB

id: 588665495487811584
@8_s2_5 おかえりなさいまし

id: 588665495492124672
O link http://t.co/u8yh4xdIAF por @YouTube é o tweet mais popular hoje na min
ha feed.

id: 588665495487844352
RT @dam_anison: 【アニサマ2014 LIVEカラオケ⑤】
μ'sのライブ映像がDAMに初登場! それは「それは僕たちの奇跡」！
μ's結成から5年間の"キセキ"を噛み締めながら歌いたい！
→http://t.co/ZCAB7jgE4L #anisama http:…

id: 588665495513006080
[Tweet not available]

id: 588665495525588992
坂道の時に限って裏の車がめっちゃ車間距離近づけて停めてくるから死ぬかと思った

id: 588665495512948737
Christina Grimmie #RisingStar
17

id: 588665495487909888
Dolgun Dudaklı Kadınların Çok İyi Bildiği 14 Şey http://t.co/vvEzTlqWOv htt
p://t.co/dsWke4uXQ3
```

Although we provided the list of IDs as a string in the above example, the standard use case is to pass a file-like object as the argument to `expand_tweetids()`.