

MINING DATA FROM TWITTER

Abhishanga Upadhyay

Luis Mao

Malavika Goda Krishna

Abstract

The purpose of this report is to illustrate how to data mine Twitter to anyone with a computer and Internet access. It provides step-by-step explanations so that any person can extract valuable information from Twitter.

There are 271 million active users on Twitter. There are many reasons to extract data from the users. Whether it is for advertisement, network analysis or just curiosity, this report will explain in detail how to data mine Twitter.

The report will cover specific script examples such as: what is trending, tweets relating a hash-tag and much more. There will be a detailed explanation of those scripts and also other usage possibilities the Twitter API. It will also cover some basics, such as how to get started (getting python, Tweepy and other tools).

The API is so simple and easy to use that any person with little programming background will be able to analyze the data.

Data Mining Twitter

Context

Human analysts with no special tools can no longer make sense of large volumes of data. Data mining can automate the process of finding patterns in raw data. The results can be either utilized by automated decision support systems or by manual human testing. Data mining is an integral part of science and business areas to analyze large amounts of data to discover trends.

Twitter is a great tool for social web mining because it is a rich source of social data due to its inherent openness for public consumption. It is a clean and well-documented API, rich developer tool, and has a broad appeal to users. Data mining in Twitter is simple and can bring significant value.

Data Mining is extraction of knowledge hidden from large volumes of raw data. Knowledge discovery varies from traditional information retrieval from databases. In a traditional DBMS, database records are returned in response to a query; while in knowledge discovery, what is retrieved is implicit patterns. The process of discovering such patterns is termed data mining.

Two main reasons to use data mining:

- Large amounts of data that can't be handled by individuals
- Need of making significant extrapolations from large sets of data

Overview

This project endeavors to introduce some elementary analytics functions that can be implemented by using a twitter APIs. It will emphasize on techniques and considerations for mining the large amounts of data placed away at twitter. It will be explained why Twitter is easy to use, what are some important terms like (API, REST, OAuth, etc) that need to be understood and how to use the tools needed. Several examples will be explained in great detail.

Twitter's simplicity

Twitter data is interesting because tweets happen at the "speed of thought" and are available for consumption in real time, and you can obtain data from anywhere in the world.

We chose because Twitter is predominantly suited for data mining because of three key features.

- Twitter's API is well designed and easy to access.
- Twitter data is in a convenient format for analysis.

- Twitter's terms of use for the data are relatively liberal as compared to other APIs. It is in general acceptable that tweets are public and reachable to anyone. It is based on the asymmetric following model that allows access to any account without request for approval.

Twitter data is open to public scrutiny, and is subject to further elaboration on the broad number of data mining possibilities by providing a succinct collection of recipes in a convenient problem/solution format that can be easily influenced and readily applied to a wide range of problems and applications.

We have familiarized ourselves with the process of setting up a development environment with Python, survey Twitter's API, and can explore the following:

- Twitter's developer platform and how to make API requests
- Tweet metadata and how to use it
- Extracting entities such as user mentions, hashtags, and URLs from tweets
- Techniques for performing frequency analysis with Python

Twitter API

In computer programming, an Application Programming Interface (API) requires a software component in terms of its operations, their inputs and outputs and underlying types. Its main purpose is to define a set of functionalities that are independent of their respective implementation, allowing both definition and implementation to vary without compromising each other. An application-programming interface (API) is a set of programming instructions and standards for accessing a Web-based software application or Web tool. A software company releases its API to the public so that other software developers can design products that are powered by its service.

In addition to accessing databases or computer hardware, such as hard disk drives or video cards, an API can be used to ease the work of programming graphical user interface components, to allow integration of new features into existing applications (a so-called "plug-in API"), or to share data between otherwise distinct applications. In practice, many times an API comes in the form of a library that includes specifications for routines, data structures, object classes, and variables. In some other cases, notably for SOAP and REST services, an API comes as just a specification of remote calls exposed to the API consumers.¹

Twitter bases its Application Programming interface (API) of the Representational State Transfer (REST) Architecture. REST architecture refers to a collection of network design principles that define resources and ways to address and access data. The architecture is a design philosophy, not a set of blueprints there's no single prearranged arrangement of

computers, servers and cables. For Twitter, a REST architecture in part means that the service works with most Web syndication formats.²

Using twitter's API we can answer questions like:

- How many friends/followers do I have?
- Who am I following that is not following me back?
- Who is following me that I am not following back?
- Who are the friendliest and least friendly people in my network?
- Who are my "mutual friends"?
- Given all of my followers and all of their followers, what is my potential influence if I get retweeted?

RESTful and OAuth

REST stands for Representational State Transfer. It is a client-server, communications protocol and in virtually all cases, the HTTP protocol is used.

REST is an architecture style for designing networked applications. The idea is that, rather than using complex mechanisms such as CORBA, RPC or SOAP to connect between machines, simple HTTP is used to make calls between machines.

RESTful applications use HTTP requests to post data (create and/or update), make queries, and delete data. Thus, REST uses HTTP for all four Create/Read/Update/Delete operations.

REST is not a "standard" form. There will never be a World Wide Web Consortium (W3C) recommendation for REST, for example. And while there are REST programming frameworks, working with REST is so simple that you can often come up with standard library features in languages like Perl, Java, or C#.

OAuth is short for Open Authorization. OAuth provides a way for you to authorize an application to access data you have stored away in another application without having to share your username and password.

- You (the end user) want to authorize an application of some sort (the client) to access some of your data (a scope) that's managed by a web service (the resource owner).
- Instead of asking for your password, the client redirects you to the resource owner, and you authorize a scope for the client directly with the resource owner.
- Assuming the end user authorizes the client, the client is notified and given an authorization code confirming that the end user has authorized it to access a scope.

- The client presents the authorization code it just received along with its client identifier and corresponding client secret to the resource owner and gets back an access token. The combination of client identifier, client secret, and authorization code ensures that the resource owner can positively identify the client and its authorization.
- The client uses the access token to make requests on behalf of the end user until the access token is revoked or expires.³

REST API VS. STREAMING API

The set of streaming APIs offered by Twitter give developers low latency access to Twitter's global stream of Tweet data. A proper implementation of a streaming client will be pushed messages indicating Tweets and other events have occurred.⁴

Twitter offers several streaming endpoints, each customized to certain use cases.

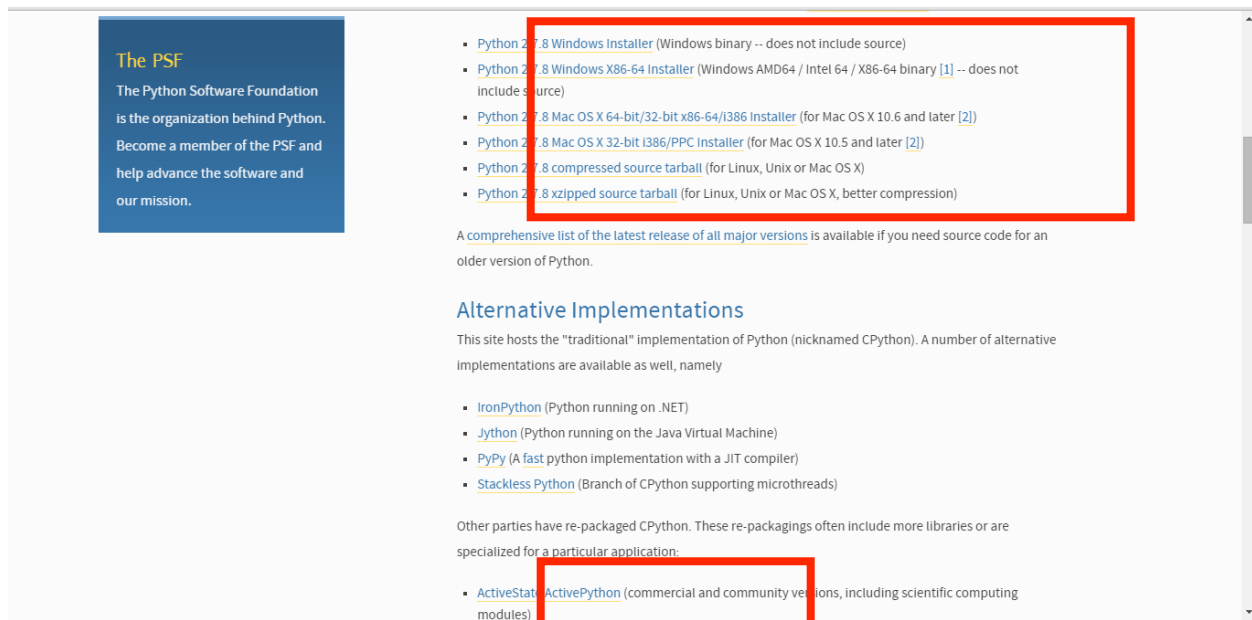
Public Streams	Streams of the public data flowing through Twitter. Suitable for following specific users or topics, and data mining.
User Streams	Single-user streams, containing roughly all of the data corresponding with a single user's view of Twitter.
Site Streams	The multi-user version of user streams. Site streams are intended for servers which must connect to Twitter on behalf of many users.

Tools Needed

We need to first set up the working environment before collecting and analyzing twitter data. Scripts used to extract data from twitter are written in Python so the first step is python installation. Also, we will need to download Tweepy, which is an open-sourced library that has useful functions for our project. Lastly, we need to get Access tokens which are the permissions given by Twitter to access their API.

Python

Python can be downloaded and installed at <http://www.python.org/download/>. It is usually recommended that Windows users install ActivePython, which automatically adds Python to your path at the Windows Command Prompt and comes with **easy_install**. This allows you to effortlessly install Python packages instead of downloading, building, and installing them from source. **Pip** is another package manager similar to easy_install.



- It is recommended to install 2.7 version of Python as many packages still support python 2 and not the 3.1x version.
- Once Python is installed you should check that the installation was successful. The following example code can be tried out to make sure that Python is functioning properly:

```
>>> print "Hello World"
```

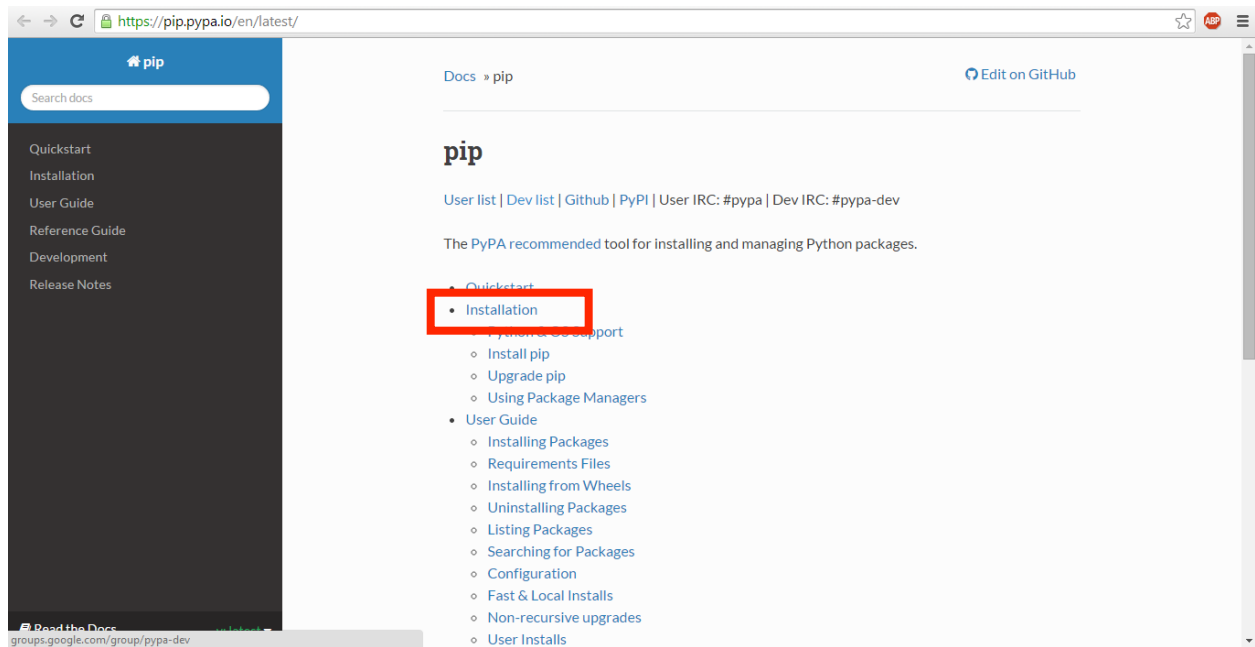
```
Hello World
```

```
>>> for i in range(0,10): # a loop
```

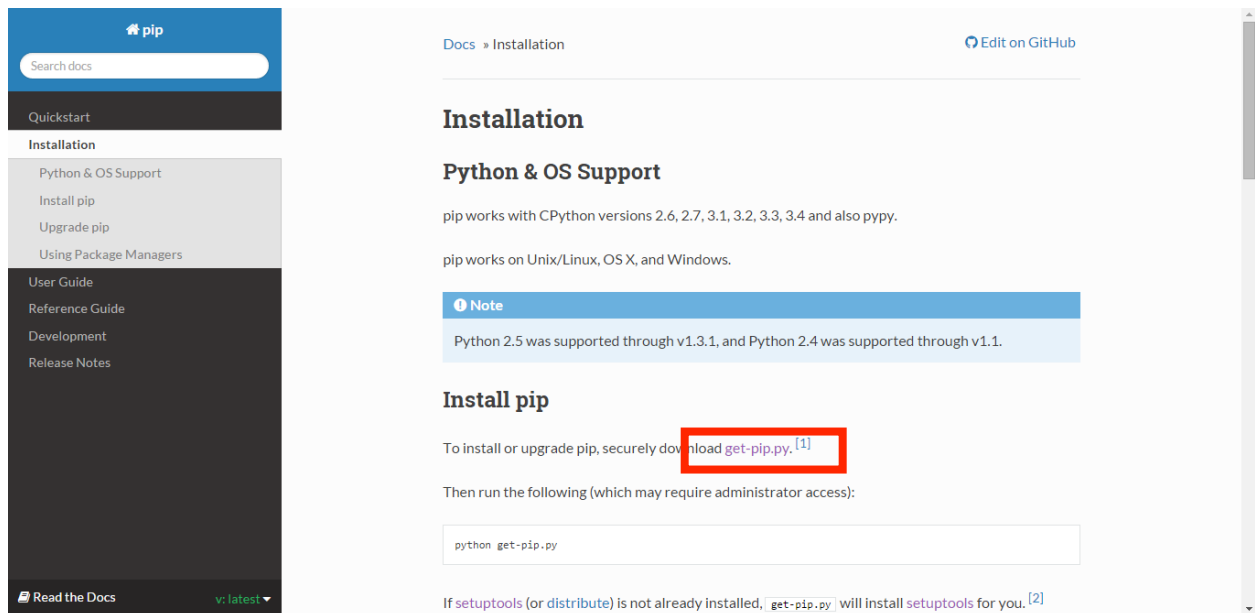
```
...     print i, # the comma suppresses line breaks ...
```

```
0 1 2 3 4 5 6 7 8 9
```

- The code could also be written in an editor like notepad++ and saved in .py format. You could then run the code using `python <name_of_file>.py` in the command shell.
- <http://learnpythonthehardway.org/> is a good website that can help and learning Python if you are unfamiliar with it.
- From <http://pypi.python.org/pypi/setuptools> you can download the latest version of easy_install from where there are specific instructions for each platform.
- To install pip you could go to www.pip-installer.org/en/latest/ and click on installation.



- In the installation page click on get-pip.py



- After this just run the python file and since ActivePython is already installed Python27/Scripts is already included as a part of the Path.
- The last step is to type pip install requests in the shell or terminal
- You can type pip in the shell to see if it has been properly installed. You should see something like this


```
Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Malavika>pip

Usage:
  pip <command> [options]

Commands:
  install           Install packages.
  uninstall         Uninstall packages.
  freeze            Output installed packages in requirements format.
  list              List installed packages.
  show              Show information about installed packages.
  search            Search PyPI for packages.
  wheel             Build wheels from your requirements.
  zip               DEPRECATED. Zip individual packages.
  unzip            DEPRECATED. Unzip individual packages.
  bundle            DEPRECATED. Create pybundles.
  help             Show help for commands.

General Options:
  -h, --help        Show help.
  -v, --verbose      Give more output. Option is additive, and can be
                    used up to 3 times.
  -U, --version      Show version and exit.
  -q, --quiet        Give less output.
  --log-file <path> Path to a verbose non-appending log, that only
                    logs failures. This log is active by default at
                    C:\Users\Malavika\pip\pip.log.
  --log <path>       Path to a verbose appending log. This log is
                    inactive by default.
  --proxy <proxy>     Specify a proxy in the form
                    [user:passwd@]proxy.server:port.
  --timeout <sec>     Set the socket timeout (default 15 seconds).
  --exists-action <action> Default action when a path already exists:
                    (s)witch, (i)gnore, (w)ipe, (b)ackup.
  --cert <path>       Path to alternate CA bundle.

C:\Users\Malavika>
```

Tweepy

Next step is to install Tweepy. Tweepy is open-sourced, hosted on [GitHub](https://github.com/tweepy/tweepy) and enables Python to communicate with Twitter platform and use its API.

- To install tweepy go to <https://github.com/tweepy/tweepy> and download the zip file

GitHub Explore Features Enterprise Blog [Sign up](#) [Sign in](#)

tweepy / tweepy ★ Star 2,236 🍴 Fork 788

Twitter for Python! <http://tweepy.org>

850 commits 4 branches 20 releases 80 contributors

branch: master tweepy / +

Fix __init__.py
Aaron1011 authored 4 days ago latest commit d9d5a8648d

File	Description	Time
docs	Use tweepy.Cursor instead of Cursor in cursor tutorial	5 months ago
examples	Change to make sure tweepy use SSL	2 months ago
tests	add access_type kwarg to OAuthHandler.get_authorization_url(), pass t...	2 months ago
tweepy	Fix __init__.py	4 days ago
.coveragerc	Fix coverage reports to only include tweepy code.	a year ago
.gitignore	Fix coverage reports to only include tweepy code.	a year ago
.travis.yml	Install requirements.txt on Travis CI	5 months ago
ASCII_LOGO.txt	New README.	2 years ago
CHANGELOG.md	Add more changelog notes for 2.2 release.	9 months ago
CONTRIBUTORS	Add contributor.	9 months ago

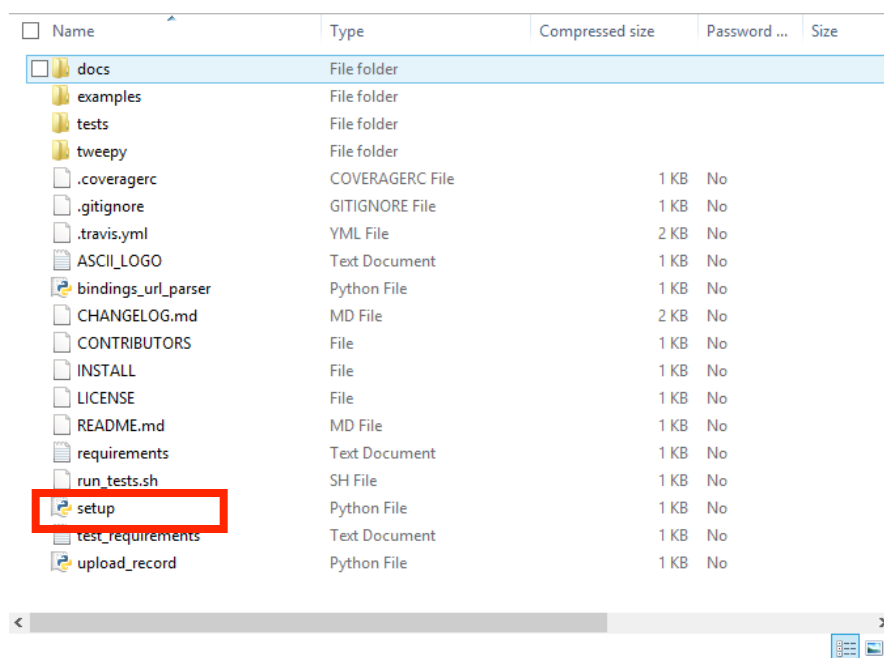
Code
Issues 31
Pull Requests 2
Pulse
Graphs

HTTPS clone URL
<https://github.com/1>

You can clone with HTTPS or Subversion.

Clone in Desktop
Download ZIP

- When the folder is downloaded, copy it to the desktop and save it in a folder say tweepy_master or you can leave it wherever it is but you will have to mention the correct path to the setup file in the terminal for tweepy to install.

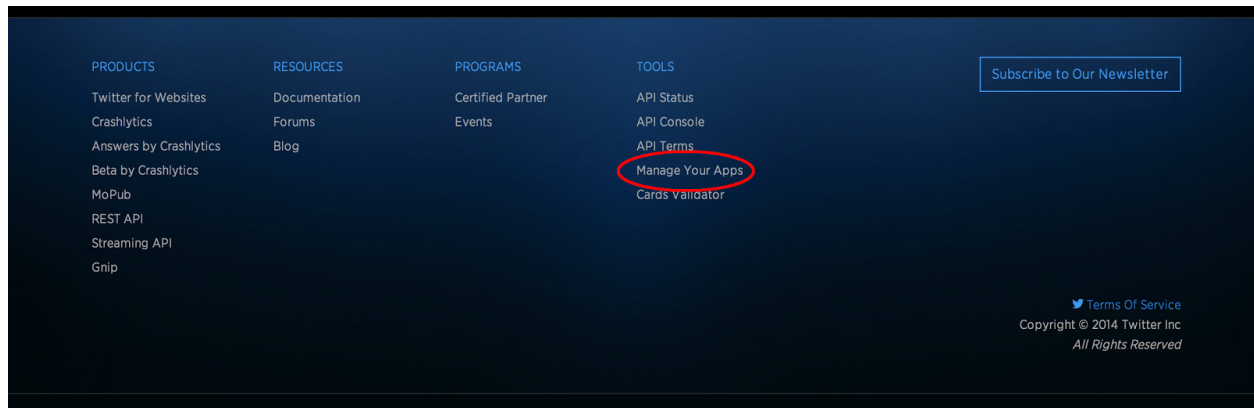


- The next step is to run this setup file in the terminal command line. To this you will have to first change the directory to the desktop: type `cd desktop`. Then you will have to change the directory to `tweepy_master` so type: `cd tweepy_master`
- To run the setup file type `python setup.py install`
- To check if tweepy is installed correctly open up the python interpreter by typing `python` in the terminal.
- Then type `import tweepy`. If this command does not throw any errors which means tweepy has been installed correctly.
- Now we are all set to type the python code to extract data from twitter

Access tokens

We need to get permission access tokens to use Twitter API. To do this, the first step is to make an account on twitter (if you don't have one).

- Go to <https://dev.twitter.com/>
- Sign into your twitter account.
- Go to manage my apps at the bottom of the page Go to <https://dev.twitter.com/>



- Click on create a new app



- Now enter the information to fill up the form and agree to the terms and conditions

Application Details

Name *

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.

(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

Where should we return after successfully authenticating? [OAuth 1.0a](#) applications should explicitly specify their `oauth_callback` URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

Developer Rules of the Road

Last Update: September 16, 2014.

Twitter maintains an open platform that supports the millions of people around the world who are sharing and discovering what's happening now. We want to empower our ecosystem partners to build valuable businesses around the information flowing through Twitter. At the same time, we aim to strike a balance between encouraging interesting development and protecting both Twitter's and users' rights.

So, we've come up with a set of Developer Rules of the Road ("Rules") that describes the policies and philosophy around what type of innovation is permitted with the content and information shared on Twitter.

The Rules will evolve along with our ecosystem as developers continue to innovate and find new, creative ways to use the Twitter API, so please check back periodically to see the current version. Don't do anything prohibited by the Rules and talk to us if you think we should make a change or give you an exception.

If your application will eventually need more than 1 million user tokens, or you expect your [embedded Tweets](#) and [embedded timelines](#) to exceed 10 million daily impressions, you will need to talk to us directly about your access to the Twitter API as you may be subject to additional terms. Furthermore, applications that attempt to replicate Twitter's core user experience (as described in Section I.5 below) will need our permission to have more than 100,000 user tokens and are subject to additional terms.

☐ Yes, I agree

Create your Twitter application

- This will generate a consumer key and consumer secret which we will use in the script.
- Next we should generate the access tokens

Application Actions

[Regenerate Consumer Key and Secret](#) [Change App Permissions](#)

Your Access Token

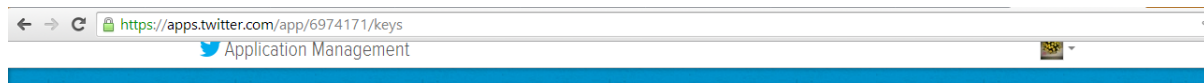
You haven't authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

Token Actions

[Create my access token](#)

- Access tokens, consumer key and consumer secret can be found in the keys and access tokens tab.



Data Mining AbhMal

[Test OAuth](#)

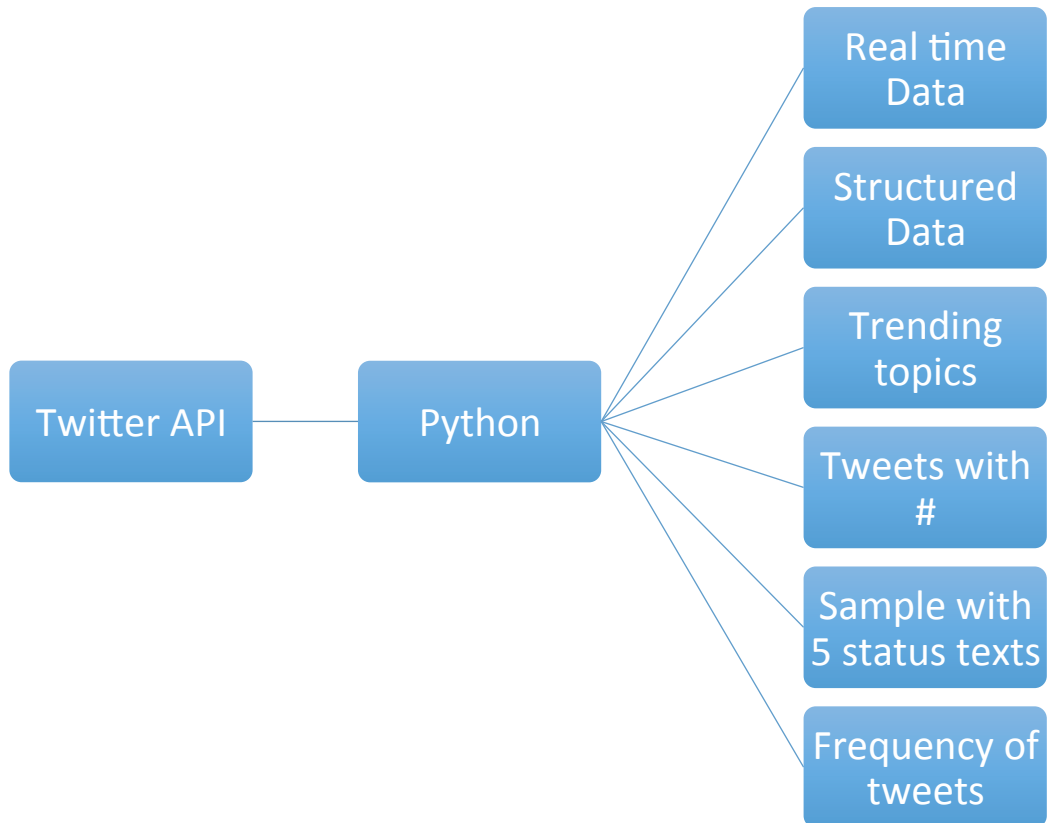
[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)

Application Settings

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

Scripts

Once all the tools needed are obtained, we can start using Python to extract data from Twitter. There are six specific examples that are described in this report. The following graph represents them.



Tweets containing the word “car” (real time)

Open up any editor like notepad++ to type the code and save the file in .py format. The following code basically extracts huge amount of data from twitter in real time.

```
1 from tweepy import Stream
2 from tweepy import OAuthHandler
3 from tweepy.streaming import StreamListener
4
5 ckey =
6 csecret
7 atoken =
8 asecret
9
10 class listener(StreamListener):
11
12     def on_data(self, data):
13         print data
14         return True
15
16     def on_error(self, status):
17         print status
18
19 auth = OAuthHandler(ckey, csecret)
20 auth.set_access_token(atoken, asecret)
21 twitterStream = Stream(auth, listener())
22 twitterStream.filter(track=["car"])
```

The above code extracts all tweets that are related to the word “car” in real time. It will extract a large amount of data from twitter constantly until we decide to stop. This huge amount of data is due to the fact that “car” is a very common word

The set of streaming APIs offered by Twitter give developers low latency access to Twitter’s global stream of Tweet data. A proper implementation of a streaming client will be pushed messages indicating Tweets and other events have occurred.

Splitting text

There are functions that can help us clean up the data that we obtain in a more understandable manner.

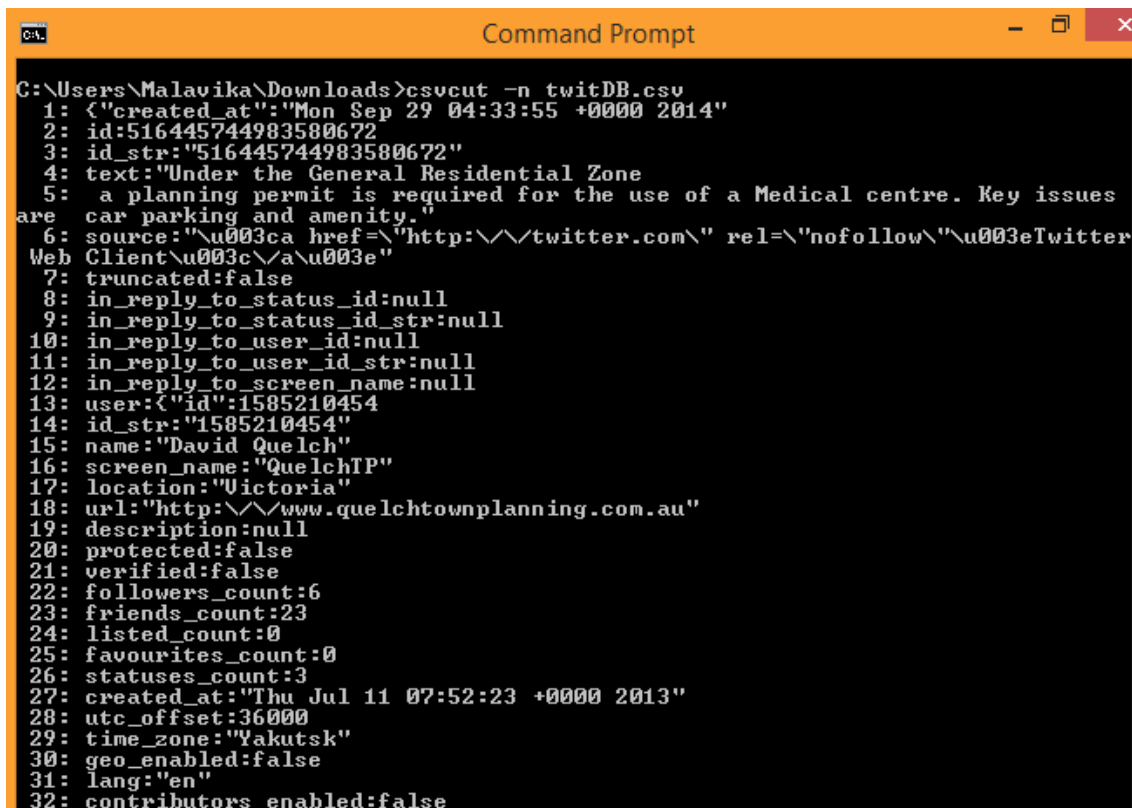
CSVKIT tool

In order to properly understand this data and make sense of what exactly this .csv file contains we used a tool called CSVKIT which is a suite of utilities for working with csv files. It can be installed by using the command `pip install csvkit`.

The first basic command is `csvlook` which just looks at your csv file and just displays it. You'll see a mess of data, pipe character and dashes. That's because this dataset has many columns and they won't all fit in the terminal at once. To fix this we need to learn how to reduce our dataset before we look at it. `csvcut` is a command used to reduce the amount of data displayed and look at only what you need. The following version of the command is used to view all the columns in the csv file.

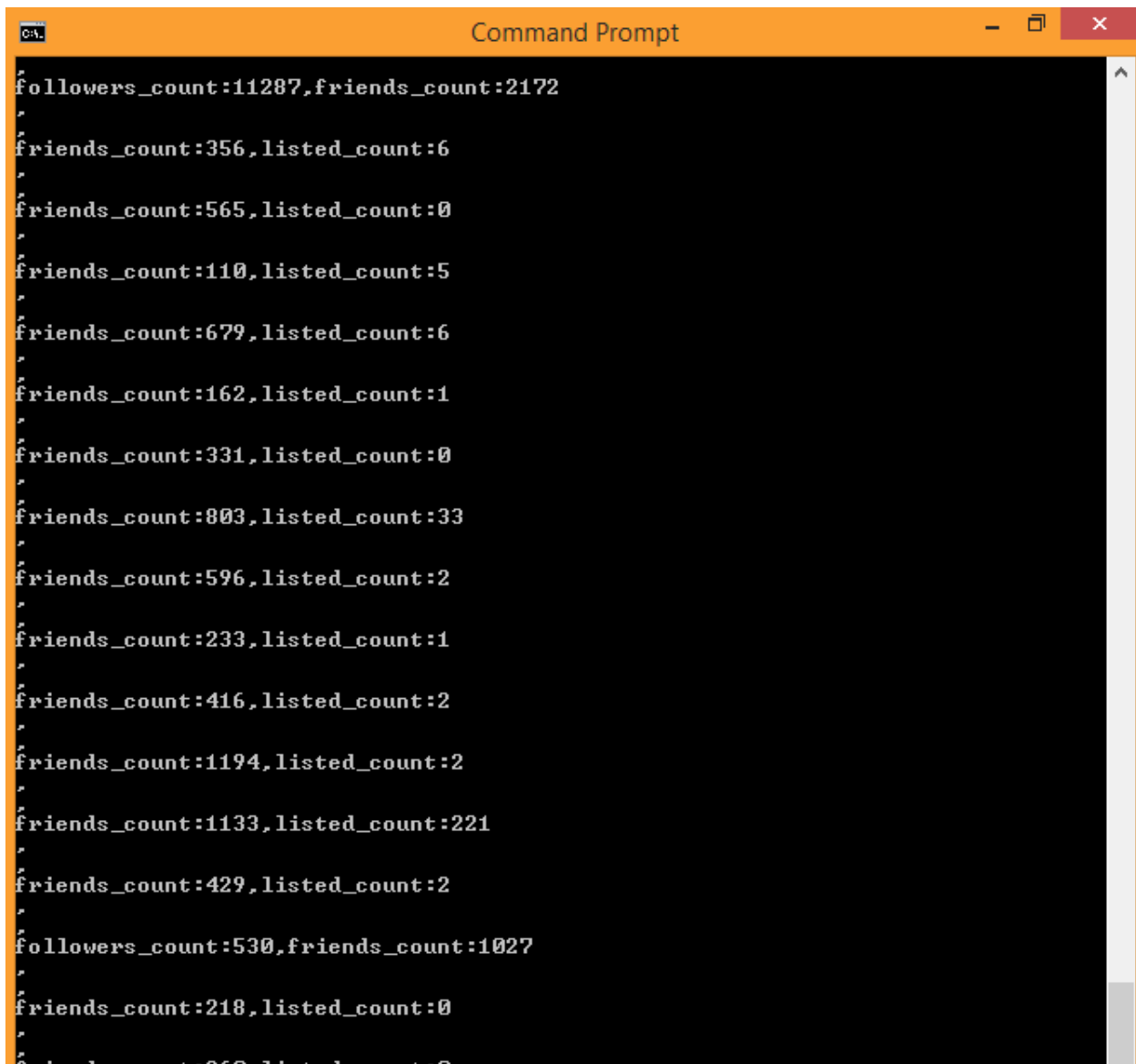
```
csvcut -n filename.csv
```

The output of this command is as follows



```
C:\Users\Malavika\Downloads>csvcut -n twitDB.csv
1: {"created_at":"Mon Sep 29 04:33:55 +0000 2014"
2: id:516445744983580672
3: id_str:"516445744983580672"
4: text:"Under the General Residential Zone
5:   a planning permit is required for the use of a Medical centre. Key issues
are car parking and amenity."
6: source:"\u003ca href=\"http://twitter.com\" rel=\"nofollow\" \u003eTwitter
Web Client\u003c/a\u003e"
7: truncated:false
8: in_reply_to_status_id:null
9: in_reply_to_status_id_str:null
10: in_reply_to_user_id:null
11: in_reply_to_user_id_str:null
12: in_reply_to_screen_name:null
13: user:{\"id\":1585210454
14: id_str:\"1585210454\"
15: name:\"David Quelch\"
16: screen_name:\"QuelchTP\"
17: location:\"Victoria\"
18: url:\"http://www.quelchtownplanning.com.au\"
19: description:null
20: protected:false
21: verified:false
22: followers_count:6
23: friends_count:23
24: listed_count:0
25: favourites_count:0
26: statuses_count:3
27: created_at:\"Thu Jul 11 07:52:23 +0000 2013\"
28: utc_offset:36000
29: time_zone:\"Yakutsk\"
30: geo_enabled:false
31: lang:\"en\"
32: contributors_enabled:false
```

There are other commands available on the csvkit that can be used to perform different things like `csvcut -c 22,23 filename.csv`, it displays the contents only of column 22 and 23 in this case columns 22 and 23 are friends and followers.



```
followers_count:11287,friends_count:2172
friends_count:356,listed_count:6
friends_count:565,listed_count:0
friends_count:110,listed_count:5
friends_count:679,listed_count:6
friends_count:162,listed_count:1
friends_count:331,listed_count:0
friends_count:803,listed_count:33
friends_count:596,listed_count:2
friends_count:233,listed_count:1
friends_count:416,listed_count:2
friends_count:1194,listed_count:2
friends_count:1133,listed_count:221
friends_count:429,listed_count:2
followers_count:530,friends_count:1027
friends_count:218,listed_count:0
friends_count:1068,listed_count:0
```

There are many other commands in csvkit that performs many other functions that can be used to sort out data in a csv file.

Extracting Trending Topics

This following section demonstrates how to ask Twitter for the topics that are currently trending worldwide. API can easily be parameterized to constrain the topics to more specific locations. The device for constraining queries is via Yahoo! GeoPlanet's Where On Earth (WOE) ID. Yahoo! GeoPlanet is API unto itself that aims to provide a way to map a unique identifier to any place on Earth for example the WOE ID for the whole world is 1, for US it is 23424977.

The script we used to extract trending topics from twitter is as follows

```

import twitter
import json
CONSUMER_KEY = '6DDkehPkroU1IYwB2DJIEoYue'
CONSUMER_SECRET = '56sC9z6R4dmUtDbyBOsKolGLrPeyQHooj7TcDI3udJbzzvHYrK'
OAUTH_TOKEN = '117137313-3OqzlliamkfF1DrCF9gDfmnRbmhFGvSGsLqKiwXe'
OAUTH_TOKEN_SECRET = '8S8RRWfkSjcDk03ucCvFpvug7L7KUR9QczWQTtI4XFLGV'

auth = twitter.oauth.OAuth(OAUTH_TOKEN, OAUTH_TOKEN_SECRET,
                           CONSUMER_KEY, CONSUMER_SECRET)

twitter_api = twitter.Twitter(auth=auth)

WORLD_WOE_ID = 1
US_WOE_ID = 23424977

# Prefix ID with the underscore for query string parameterization.
# Without the underscore, the twitter package appends the ID value
# to the URL itself as a special case keyword argument.

world_trends = twitter_api.trends.place(_id=WORLD_WOE_ID)
us_trends = twitter_api.trends.place(_id=US_WOE_ID)

print json.dumps(world_trends, indent=1)
print
print json.dumps(us_trends, indent=1)

```

The output file comes in JSON format. JSON (Java Script Object Notation) is a data exchange format that you will encounter on a regular basis. In a nutshell, JSON provides a way to arbitrarily store maps, lists, primitives such as numbers and strings, and combinations thereof. In other words, you can theoretically model just about anything with JSON should you desire to do so. `Json.dumps` will take a python object and serialize it to JSON.

Output of the script is as follows. It contains a URL for a trending topic represented as a search query that corresponds to the hashtag #UNCvsND or #AVvsMSST, where %23 is the URL encoding for the hashtag symbol. These represent topics that most people were talking about at the time when the data was extracted.

```
{
  "url": "http://twitter.com/search?q=%23AUvsMSST",
  "query": "%23AUvsMSST",
  "name": "#AUvsMSST",
  "promoted_content": null
},
{
  "url": "http://twitter.com/search?q=%23UNCvsND",
  "query": "%23UNCvsND",
  "name": "#UNCvsND",
  "promoted_content": null
}
```

On a side note, Twitter imposes rate limits on how many requests you can make to its API in a given time period. Each API resource states its rate limit for our convenience. It is usually about 15 requests for 15 minutes for this particular trending topic requests. The developer documentation states that the results of a Trends API query are updated only once every five minutes, so it is useless to make requests more often than that.

Searching for Tweets

This script is used to extract tweets related to a particular hashtag. One of the most common searches includes this search to fetch tweets containing a particular hashtag. Using this script we can extract all the tweets related to a particular hashtag say **#India** in this case.

There is more to a tweet than meets the eye. A tweet basically contains two other additional metadata: entity and places. Entities are any media files that are included in the tweet that have no association with twitter and places are the geo location included in the tweet by the user. So when we extract data related to a particular hashtag all this information along with number of friends, followers and a whole lot of other information is extracted.

Although we're just passing in a hashtag to the Search API at this point, we should pay attention to the fact that it contains a number of powerful operators that allow you to filter queries according to the existence or nonexistence of various keywords, origin of the tweet, location, etc. Search results are obtained by querying the API which returns the response as search metadata. We should include a function call such as `twitter_api.search.tweets(q='%23India', include_entities=1, max_id=313519052523986943)`. The code for this is as shown below:

```

1  import twitter
2  import json
3  CONSUMER_KEY = 
4  CONSUMER_SECRET = 
5  OAUTH_TOKEN = '
6  OAUTH_TOKEN_SECRET = 
7
8  auth = twitter.oauth.OAuth(OAUTH_TOKEN, OAUTH_TOKEN_SECRET,
9                             CONSUMER_KEY, CONSUMER_SECRET)
10
11  twitter_api = twitter.Twitter(auth=auth)
12  q = '#India'
13  count = 100
14
15  search_results = twitter_api.search.tweets(q=q, count=count)
16
17  statuses = search_results['statuses']
18
19  # Iterate through 5 more batches of results by following the cursor
20
21  for _ in range(5):
22      print "Length of statuses", len(statuses)
23      try:
24          next_results = search_results['search_metadata']['next_results']
25      except KeyError, e: # No more results when next_results doesn't exist
26          break
27
28      kwargs = dict([ kv.split('=') for kv in next_results[1:].split("&") ])
29
30      search_results = twitter_api.search.tweets(**kwargs)
31      statuses += search_results['statuses']
32
33  # Show one sample search result by slicing the list...
34  print json.dumps(statuses[0], indent=1)
35

```

Output of the file is as follows. As mentioned before it does not contain just the tweet but in fact 5KB of extra information represented in uncompressed JSON format.

```

125,
147
1,
"expanded_url": "http://twitter.com/viaguru",
"display_url": "twitter.com/viaguru"
}
1
}
}
"followers_count": 149,
"profile_sidebar_border_color": "EEEEEE",
"id_str": "363041348",
"profile_background_color": "B2DFDA",
"listed_count": 5,
"is_translation_enabled": false,
"utc_offset": 19800,
"statuses_count": 6700,
"description": "Top News & Press Release Media for Business, Technology, Culture & Trending Topics from Community http://t.co/62em6IeWQ0 AND http://t.co/o51wdThg1H",
"friends_count": 219,
"location": "India",
"profile_link_color": "93A644",
"profile_image_url": "http://pbs.twimg.com/profile_images/1515789581/news_pressguru_normal.png",
"following": false,
"geo_enabled": false,
"profile_background_image_url": "http://pbs.twimg.com/profile_background_images/320195084/twitter-pressguru-viaguru-right.gif",
"screen_name": "pressguru",
"lang": "en",
"profile_background_tile": false,
"favourites_count": 0,
"name": "PressGuru",
"notifications": false,
"url": "http://t.co/BggU5iFSh7",
"created_at": "Sat Aug 27 12:38:38 +0000 2011",
"contributors_enabled": false,
"time_zone": "New Delhi",
"protected": false,
"default_profile": false,
"is_translator": false
},
"geo": null,
"in_reply_to_user_id_str": null,
"possibly_sensitive": false,
"lang": "und",
"created_at": "Tue Oct 07 04:07:33 +0000 2014",
"in_reply_to_status_id_str": null,
"place": null,
"metadata": {
"iso_language_code": "und",
"result_type": "recent"
}
}

```

In order to trim this data and extract only a subset of the data like for example the first five search results we use the following code. It distills the text of the tweet and its entities into separate data structures and you can mention a range say [0:5] representing the number of tweets you want to see.

```

8  auth = twitter.oauth.OAuth(OAUTH_TOKEN, OAUTH_TOKEN_SECRET,
9                               CONSUMER_KEY, CONSUMER_SECRET)
10
11  twitter_api = twitter.Twitter(auth=auth)
12  q = '#India'
13
14  count = 100
15
16  # See https://dev.twitter.com/docs/api/1.1/get/search/tweets
17
18  search_results = twitter_api.search.tweets(q=q, count=count)
19
20  statuses = search_results['statuses']
21
22  status_texts = [ status['text']
23                  for status in statuses ]
24
25  screen_names = [ user_mention['screen_name']
26                  for status in statuses
27                  for user_mention in status['entities']['user_mentions'] ]
28
29  hashtags = [ hashtag['text']
30              for status in statuses
31              for hashtag in status['entities']['hashtags'] ]
32
33  # Compute a collection of all words from all tweets
34  words = [ w
35           for t in status_texts
36           for w in t.split() ]
37
38  # Explore the first 5 items for each...
39
40  print json.dumps(status_texts[0:5], indent=1)
41  print json.dumps(screen_names[0:5], indent=1)
42  print json.dumps(hashtags[0:5], indent=1)
43  print json.dumps(words[0:5], indent=1)
44

```

In the above script the syntax: `status_texts[0:5]`, indicates slicing, whereby you can easily extract items from lists or substrings from strings. In this particular case, `[0:5]` indicates that you want to see the first five search results. The output of this script is as follows. It contains only five search results.

```

C:\Users\Abhishanga\Desktop\EE599>python ee5993.py
[
  "#Pakistan urges #India to show restraint; condemns #LoC deaths.\n\nhttp://t.co/jWkikysLit",
  "A new broom sweeps in India\u2019s clean-up drive - http://t.co/mT997fssXv http://t.co/CejPxNY18x #swachhbharat #mycleanindia #sanitation #india",
  "Read the new and compare twitch ISPR statement... #pakistan #india who is right\n\nhttp://t.co/35IRY9YHMM.",
  "RT @SabinaEngland: my film, DEAF BROWN GURL is finally here! WATCH IT HERE http://t.co/NNtMXDDcFn #india #filmmaking #deaf #ASL",
  "RT @SandraLongman: In The Pink by Alice http://t.co/mKu36hS2KT via @Etsy #hexagon #homedesign #travel #Rome #Milan #Italy #India #Jaka\u2026"
]
[
  "SabinaEngland",
  "SandraLongman",
  "Etsy",
  "W7U0A",
  "BillGates"
]
[
  "Pakistan",
  "India",
  "LoC",
  "swachhbharat",
  "mycleanindia"
]
[
  "#Pakistan",
  "urges",
  "#India",
  "to",
  "show"
]
C:\Users\Abhishanga\Desktop\EE599>

```

Frequency Distribution of Tweets

Virtually all analysis boils down to the simple exercise of counting things on some level, so that it can be counted and further manipulated in meaningful ways.

Counting something is the starting point for any kind of statistical filtering or manipulation that strives to find out something meaningful in a mess of data. We just extracted the first 5 items just to get a feel for the data, let's now take a closer look at what's in the data by computing a frequency distribution and looking at the top 10 items in each list.

In order to make reviewing the results a little easier to comprehend we convert the results into a tabular format. You can install a package called *prettytable* by typing ***pip install prettytable*** in the terminal.

The script is as shown below

```
10 auth = twitter.oauth.OAuth(OAUTH_TOKEN, OAUTH_TOKEN_SECRET,
11                             CONSUMER_KEY, CONSUMER_SECRET)
12
13 twitter_api = twitter.Twitter(auth=auth)
14 q = '#India'
15
16 count = 100
17
18
19 search_results = twitter_api.search.tweets(q=q, count=count)
20
21 statuses = search_results['statuses']
22
23 status_texts = [ status['text']
24                  for status in statuses ]
25
26 screen_names = [ user_mention['screen_name']
27                  for status in statuses
28                  for user_mention in status['entities']['user_mentions'] ]
29
30 hashtags = [ hashtag['text']
31              for status in statuses
32              for hashtag in status['entities']['hashtags'] ]
33 # Compute a collection of all words from all tweets
34 words = [ w
35           for t in status_texts
36           for w in t.split() ]
37
38 for label, data in (('Word', words),
39                    ('Screen Name', screen_names),
40                    ('Hashtag', hashtags)):
41     pt = PrettyTable(field_names=[label, 'Count'])
42     c = Counter(data)
43     [ pt.add_row(kv) for kv in c.most_common()[:10] ]
44     pt.align[label], pt.align['Count'] = 'l', 'r' # Set column alignment
45     print pt
46
```

The output of this file is as follows. It shows a neat list of the frequency of tweets in a clean tabular format. Here RT is a pretty common token demonstrating that number of retweets was large. #India (#india) is mentioned about 74 times. This includes #india too which has tweeted 22 times as seen from the output table.

```
C:\Users\Abhishanga\Desktop>cd EE599
```

```
C:\Users\Abhishanga\Desktop\EE599>python ee5994.py
```

Word	Count
#India	52
in	35
RT	26
#india	22
for	20
to	19
the	17
a	16
and	13
of	13

Screen Name	Count
IBNMoney_com	5
narendramodi	4
BillGates	3
IKJadejaBJP	2
vevck	2
prashantktm	2
ZaidZamanHamid	2
IndiaFactsOrg	2
AdityaRajKaul	2
PakkuIntl	2

Hashtag	Count
India	77
india	22
Pakistan	5
Economy	5
business	5
WorldBank	4
Philippines	3
Modi	3
Goa	2
ceasefire	2

Conclusion

The sample code in this report should be enough to get anyone started in using Twitter's API. It was illustrated how simple it is to use Python to interactively explore and analyze Twitter data. We provided some starting templates that you can use for mining tweets.

We showed how to create an authenticated connection and then progressed through a series of examples that illustrated how to discover trending topics for particular locales, how to search for tweets that might be interesting, and how to analyze those tweets using some elementary but effective techniques based on frequency analysis and simple statistics. Even what seemed like a somewhat arbitrary trending topic turned out to lead us down worthwhile paths with lots of possibilities for additional analysis.

As the examples show, Twitter API is simple to use and easily accessible. There are infinite possibilities on what one can do. There are some limitations on the amount of queries and data that Twitter allows you to get every 15 minutes, however an easy way around that is to get more access tokens.

REFERENCES:

(1) "Application Programming Interface." *Wikipedia*. Wikimedia Foundation, 23 Oct. 2014. Web. 24 Oct. 2014.

(2) "Twitter's API - HowStuffWorks." *HowStuffWorks*. N.p., n.d. Web. 24 Oct. 2014.

(3) Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More-Matthew A. Russell

(4) "The Streaming APIs." *Twitter Developers*. N.p., n.d. Web. 23 Oct. 2014.

"Sentdex." *Sentdex*. N.p., n.d. Web. 24 Oct. 2014.

"Abhishanga/Twitter-API-Python." *GitHub*. N.p., n.d. Web. 24 Oct. 2014.