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 hashtag 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.  

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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Streams</td>
<td>Streams of the public data flowing through Twitter. Suitable for following specific users or topics, and data mining.</td>
</tr>
<tr>
<td>User Streams</td>
<td>Single-user streams, containing roughly all of the data corresponding with a single user’s view of Twitter.</td>
</tr>
<tr>
<td>Site Streams</td>
<td>The multi-user version of user streams. Site streams are intended for servers which must connect to Twitter on behalf of many users.</td>
</tr>
</tbody>
</table>

### 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/](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:

```python
>>> print "Hello World"
Hello World
```
```python
>>> for i in range(0,10): # a loop
...	print i, # the comma suppresses line breaks ...
```
```plaintext
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.
Tweepy

Next step is to install Tweepy. Tweepy is open-sourced, hosted on GitHub 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
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

**Twitter Apps**

- Now enter the information to fill up the form and agree to the terms and conditions
• This will generate a consumer key and consumer secret which we will use in the script.
• Next we should generate the access tokens
• Access tokens, consumer key and consumer secret can be found in the keys and access tokens tab.
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.
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.
In the above code using the `data.split` command we can trim the data between a set of keywords. You will have to look at the `.csv` file and see which keywords define the boundary of the information you want to extract and include that in the `data.split` command.

This code extracts only the tweet and the username as shown below. You can see that all the tweets contain the search word “car” in it.
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

```
file.csv
1: "http://twitter.com/" rel="nofollow" u003cTwitter
Web Client\u003e u003c/a\u003e u003c/a\u003e
2: id:516445744985005672
3: id_str:"516445744985005672"
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:"
Web Client\u003e u003chttp://twitter.com/" rel="nofollow" u003e u003c/a\u003e 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: id: 5165216454
14: id_str:"5165216454"
15: name: "David Quelch"
16: screen_name: "QuelchTP"
17: location:"Victoria"
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 02:52:23 +0000 2013"
28: utc_offset: 36000
29: time_zone:"Valuetsk"
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.
There are many other commands in csvkit that perform 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
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.
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 used. 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:
import twitter
import json

CONSUMER_KEY =
CONSUMER_SECRET =
OAUTH_TOKEN =
OAUTH_TOKEN_SECRET =

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

twitter_api = twitter.Twitter(auth=auth)
q = '#India'
count = 100

search_results = twitter_api.search.tweets(q=q, count=count)

statuses = search_results['statuses']

# Iterate through 6 more batches of results by following the cursor

for _ in range(5):
    print "Length of statuses", len(statuses)
    try:
        next_results = search_results['search_metadata']['next_results']
    except KeyError, e: # No more results when next_results doesn't exist
        break

    kwargs = dict([kv.split('=') for kv in next_results[1:].split('&')])

    search_results = twitter_api.search.tweets(**kwargs)
    statuses += search_results['statuses']

    # Show one sample search result by slicing the list...
    print json.dumps(statuses[0], indent=1)

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.
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.
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.
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:

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

twitter_api = twitter.Twitter(auth=auth)
q = '#India'

# Count is 100

search_results = twitter_api.search.tweets(q=q, count=count)
statuses = search_results['statuses']

status_texts = [ status['text']
                for status in statuses ]
screen_names = [ user_mention['screen_name']
                for status in statuses
                for user_mention in status['entities']['user_mentions'] ]

hashtags = [ hashtag['text']
            for status in statuses
            for hashtag in status['entities']['hashtags'] ]

# Compute a collection of all words from all tweets
words = [ w
          for t in status_texts
          for w in t.split() ]

# 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.
```

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.
<table>
<thead>
<tr>
<th>Word</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>#India</td>
<td>52</td>
</tr>
<tr>
<td>in</td>
<td>35</td>
</tr>
<tr>
<td>RT</td>
<td>26</td>
</tr>
<tr>
<td>#india</td>
<td>22</td>
</tr>
<tr>
<td>for</td>
<td>20</td>
</tr>
<tr>
<td>to</td>
<td>19</td>
</tr>
<tr>
<td>the</td>
<td>17</td>
</tr>
<tr>
<td>a</td>
<td>16</td>
</tr>
<tr>
<td>and</td>
<td>13</td>
</tr>
<tr>
<td>of</td>
<td>13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Screen Name</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBMMoney_com</td>
<td>5</td>
</tr>
<tr>
<td>narendramodi</td>
<td>4</td>
</tr>
<tr>
<td>BillGates</td>
<td>3</td>
</tr>
<tr>
<td>IKhadejaBJP</td>
<td>2</td>
</tr>
<tr>
<td>veuck</td>
<td>2</td>
</tr>
<tr>
<td>prashantktm</td>
<td>2</td>
</tr>
<tr>
<td>ZaidZamanHamid</td>
<td>2</td>
</tr>
<tr>
<td>IndiaFactsOrg</td>
<td>2</td>
</tr>
<tr>
<td>AdityaRajKaul</td>
<td>2</td>
</tr>
<tr>
<td>PakkuIntl</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hashtag</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>India</td>
<td>77</td>
</tr>
<tr>
<td>india</td>
<td>22</td>
</tr>
<tr>
<td>Pakistan</td>
<td>5</td>
</tr>
<tr>
<td>Economy</td>
<td>5</td>
</tr>
<tr>
<td>business</td>
<td>5</td>
</tr>
<tr>
<td>WorldBank</td>
<td>4</td>
</tr>
<tr>
<td>Philippines</td>
<td>3</td>
</tr>
<tr>
<td>Modi</td>
<td>3</td>
</tr>
<tr>
<td>Goa</td>
<td>2</td>
</tr>
<tr>
<td>ceasefire</td>
<td>2</td>
</tr>
</tbody>
</table>
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:


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

