Webscraping

Scraping, Visualizing, and Analyzing the Web

Twitter Scraping Tips

Twitter is a micro-blogging site where users can broadcast status updates of 140 characters or less. If you aren't that familiar with the site, you can explore it here.

While there are many social networking sites that hold rich information for research, Twitter is an ideal space because:

1. **Most profiles are public:** Other sites like Facebook and Instagram may have interesting data. However, depending on users' privacy setting, you will only be able to collect certain information, potentially skewing your findings.

2. **Tweets are more than just text:** Each time you mine data from Twitter you are open to collecting a lot of information. This may be tweet-based content such as photos, links, and geo-locations. And it also may be user-based content such as profile picture,

Tools

There is an ever-changing array of tools and platforms to mine Twitter data. Listed below are some software packages that are free, or at least offer a free version, and that are a good place to start.

We often advise students and researchers to start with NodeXL. It is easy to use and reliable, although it recently limited the capabilities of its free version. As summarized in the "Visualization and Analysis" section of this guide, NodeXL is a useful tool because it not only scrapes the data, but also helps with visualization and analysis.

This guide also covers scrapping with programming scripts, in particular the Python package Twarc. Although scripting methods require coding experience or the willingness to learn, for more in-depth or customized studies, it is often beneficial to learn enough code to use one of these approaches.

Scraping with Twarc

What it is: a package for Python created and maintained by Ed Summers at MITH

Where it is: https://github.com/edsu/twarc

Note: The creator of this package has already provided instructions about installation and usage on the github site where it is hosted. The purpose of this document is to fill in the gaps for people who might also be new to Python and want to use this tool.

Creating a Python environment

In order to use Twarc, you will need the ability to run Python from a command line interface. There are many ways to do this, but if you are new to Python, the simplest way that I've found is to install Anaconda. Anaconda is a free software that combines a full installation of Python with 300 or so of the most popular Python add-on packages.

- Download Anaconda here https://www.continuum.io/downloads
- Install Anaconda: double click the .exe file that was downloaded and follow the instructions on the screen
- Start Anaconda: you should be able to find the Anaconda icon wherever your programs are stored (eg, the start menu in Windows) and click it to run it. If you can't find it by sight, trying search the computer for it.
- Install twarc: in the command line interface of anaconda enter

  pip install twarc
- Eventually you will see a message saying it was successfully installed, and the command prompt will return.
- Run twarc for the first time: python Scripts\twarc.py

Note: this works because Anaconda runs in the working directory C:\USERNAME\Anaconda, which is where the twarc script defaults to downloading in the Scripts folder. If this throws an error, you most likely need to locate the

number of followers, and date of sign up.

3. **Tweets can only be 140 characters:** The fact that each tweet entry can only be 140 characters may seem limiting at first, but it is actually quite helpful in the analysis process. Further, users are learning how to adapt to the smaller space; therefore you are not really missing out on any important data that 141+ characters would have allowed.

4. **Twitter users can have both friends and followers:** Unlike a site like Facebook where friending is reciprocal, on Twitter users can gain followers without adding them to their friends list. Because of this, potential audiences are better analyzed and network maps can be more dynamic, revealing more information.

Some Scraping Tools

TAGS

- automatically pull results from a Twitter search into a Google Spreadsheet
- can then upload data to TAGSExplorer and create visualizations
- no coding experience needed at first but becomes necessary for deeper analysis

backtweets

- use for a quick and simple search of recent tweets by keyword
- hard to export data
- best when you are in the project design phases
- best when used in Chrome
- no coding experience needed

NODEXL

- Excel plug-in
- search by keyword or by Twitter handle(s)
- uploads and organizes data into spreadsheet
- some statistics automatically calculated with others just a few clicks away
- creates a nodes and a vertices tab automatically
- no coding experience needed

Scraping Scripts

I have found that some studies require more customized Twitter mining. For this, it becomes necessary to learn a programming language and write a script that perfectly scrapes the information that you need.

Popular programming languages for social media mining include Ruby, Python, and R. A quick internet search will provide you with a plethora of written walk-throughs and video tutorials to help you learn these programming languages for free.

Once you have an idea of how they work, it is easy to search around for others who have shared their scraping scripts and then personalize for your own needs. Of course, if you learn the language well enough, you can

twarc.py file and include the correct path in this command.

Note: If, when you install Anaconda, you have admin privileges and you select the option to install for all users, Anaconda will run from C:\Anaconda

Enter access tokens:

- You must have a twitter account.
- Go to https://apps.twitter.com/ and log in to register an app
- Once you have registered, it will generate the four keys required
- Enter those keys into prompts from twarc

Suggestion: these keys are very long and difficult to accurately transcribe by sight across windows, because most command line interfaces will not allow copy paste. Instead, you can enter whatever you want into these four prompts (xxxx) and then manually open the file where they are stored and copy and paste the keys in there.

How to find this file:

- look for the .twarc file saved in your home directory (C:\USERNAME)
- Anaconda will also tell you where the file is saved in an output immediately after you enter the final key
- open that file using a text editor (eg Notepad)
- copy and paste Twitter keys directly in
- save & close the file

You should now be able to run basic commands listed on the documentation page of the code using the basic format

python Scripts\twarc.py --search searchterm > tweets.json

Results will be stored in a file named (as you might surmise) tweets.json that will also live in your Anaconda directory unless you tell it to do otherwise. Results willl be in JSON format and will need additional manipulation

Scraping with NodeXL

NODEXL

NodeXL is a data scraping, visualization, and

write your own scripts!

Beyond just learning your chosen programming language, you must also familiarize yourself with Twitter's API (Application Programming Interface). This is Twitter's programming language that you will use in your scripts so that Twitter knows exactly what information they need to send back to you when you request it. This language is already integrated into the scraping software packages that I summarize above but invisible to the user.

Below, I share with you a Ruby script that I employed in a recent study regarding Twitter users and the show *Cosmos*. "XXX" denotes a space where you must insert your own information. "fname" is the file name to which your data will be written, and "term" is the search term on which this scrape was based. Mysql is a database management tool.

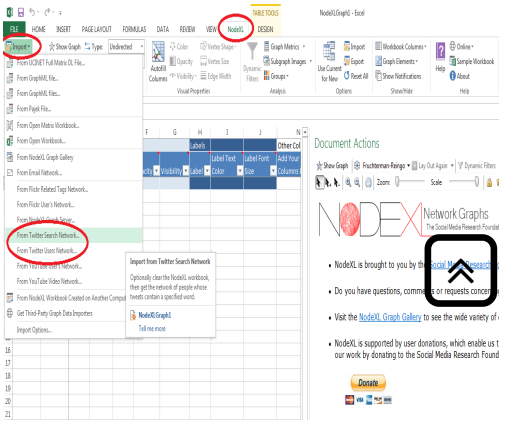
```
require 'twitter'
require 'net/http'
require 'json'
require "mysql"

fname = "cosmos_finale.txt"
term = "#watchingcosmos"

my = Mysql.connect('localhost',
'root', 'XXX', 'cosmos')

client =
Twitter::Streaming::Client.new do
|config|
    config.consumer_key
= "XXX"
    config.consumer_secret
 = "XXX"
    config.access_token
= "XXX"
    config.access_token_secret
 = "XXX"
end

Thread.new do
    while line = STDIN.gets
        break if line.chomp ==
'x'
    end
    exit
end
print "\n\n\tEnter \"x\" to
```

NodeXL is a data scraping, visualization, and analysis tool created by researchers at Microsoft Cambridge.

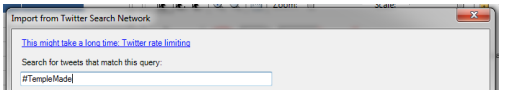
Click on the above logo to learn more about the software and to download the Excel plugin.

Here is a brief outline of the steps to take when employing NodeXL. I would suggest playing with this tool a bit before officially starting your Twitter mining. Choose a search term or user that you know (from browsing Twitter) will provide you with dynamic data. Or, you can try some exploratory searches to "test" your methods.

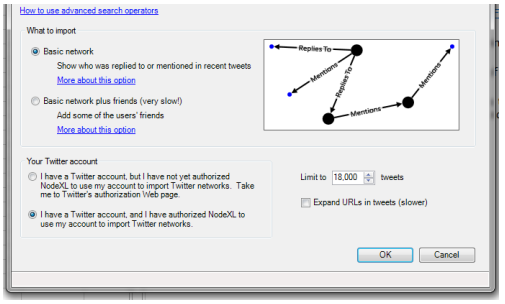
- open the NodeXL template, go to the NodeXL tab, click on "import," and choose if you want to search by user or search term



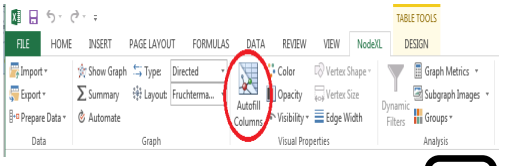
- for this example I will use the search option. Type in your search term. Here I have chosen "Basic network" because Twitter limits how much data you can pull per 15 minutes. Therefore, if I ask for too much information (such as another level of friends and followers) I may have to wait several hours or days for the scrape to complete. Notice that you do have to have a Twitter account to run this program. Once you authorize the plug-in, it will remember you. Also notice that I have told the tool to "limit" my search to 18,000 tweets. This is actually the maximum this plug-in will scrape. So, if you want to get as much tweet content as possible, 18,000 is the way to go. As you can see from the box, this tool will not only tell us who has tweeted what, it will also create a network of who was "replied to" and "mentioned."



```
quit:\n>"
File.open(fname, "w+") do |file|
    client.filter(:track => term)
do |tweet|
        rflg = 0
        mflg = 0
        txt =
tweet.text.encode('UTF-8',
:invalid => :replace, :undef =>
:replace)
        txt = txt.gsub(/\r/," ")
        txt = txt.gsub(/\n/," ")
        txt = txt.gsub(/\t/," ")
        txt =
txt.gsub(/\A[[:space:]]+|
[[:space:]]+\z/, '')
        rflg = 1 if (txt[0..2] ==
"RT ")
        mflg = 1 unless
tweet.user_mentions.empty?
        stmt = my.prepare("INSERT
INTO tweet (tweet_id, created_at,
screen_name, user_id,
retweet_flag, mentions_flag,
tweet_text)
VALUES(?,?,?,?,?,?,?)")
        stmt.execute tweet.id,
tweet.created_at,
tweet.user.screen_name,
tweet.user.id, rflg, mflg, txt
        if mflg == 1

 tweet.user_mentions.each do
|mention|
                rs =
my.query("SELECT user_id FROM
mentions WHERE user_id=#
{tweet.user.id} AND tweet_id=#
{tweet.id} AND mention_id=#
{mention.id}")
                if rs.num_rows ==
0
                    stmt =
my.prepare("INSERT INTO mentions
(tweet_id, user_id, mention_id,
user_screen_name,
mention_screen_name)
VALUES(?,?,?,?,?)")
                    stmt.execute
```



- Now take a look at your data. Notice that there are tabs in your workbook. For example, "edges" and "vertices" each get their own sheet. You may find that you have to do some cleaning.
- A great tool offered by NodeXL is the "Autofill" tool. This tool allows you to define parameters and change column settings for a whole sheet. This is obviously useful if you have thousands of rows of data.



- There are many videos online that provide great starter tutorial. For instance, check out this video.

```
tweet.id, tweet.user.id,
mention.id,
tweet.user.screen_name,
mention.screen_name
                end
            end
        end
        file.write("#
{tweet.id}\t#
{tweet.created_at}\t#
{tweet.user.id}\t#
{tweet.user.screen_name}\t#
{tweet.user_mentions}\t#{txt}\n")
if tweet.is_a?(Twitter::Tweet)
and tweet.lang == "en"
    end
    file.close
end
```