Principles/Social Media Mining

CIS 600

Week 7: Twitter Streaming API

`

Edmund Yu, PhD
Associate Teaching Professor
esyu@syr.edu

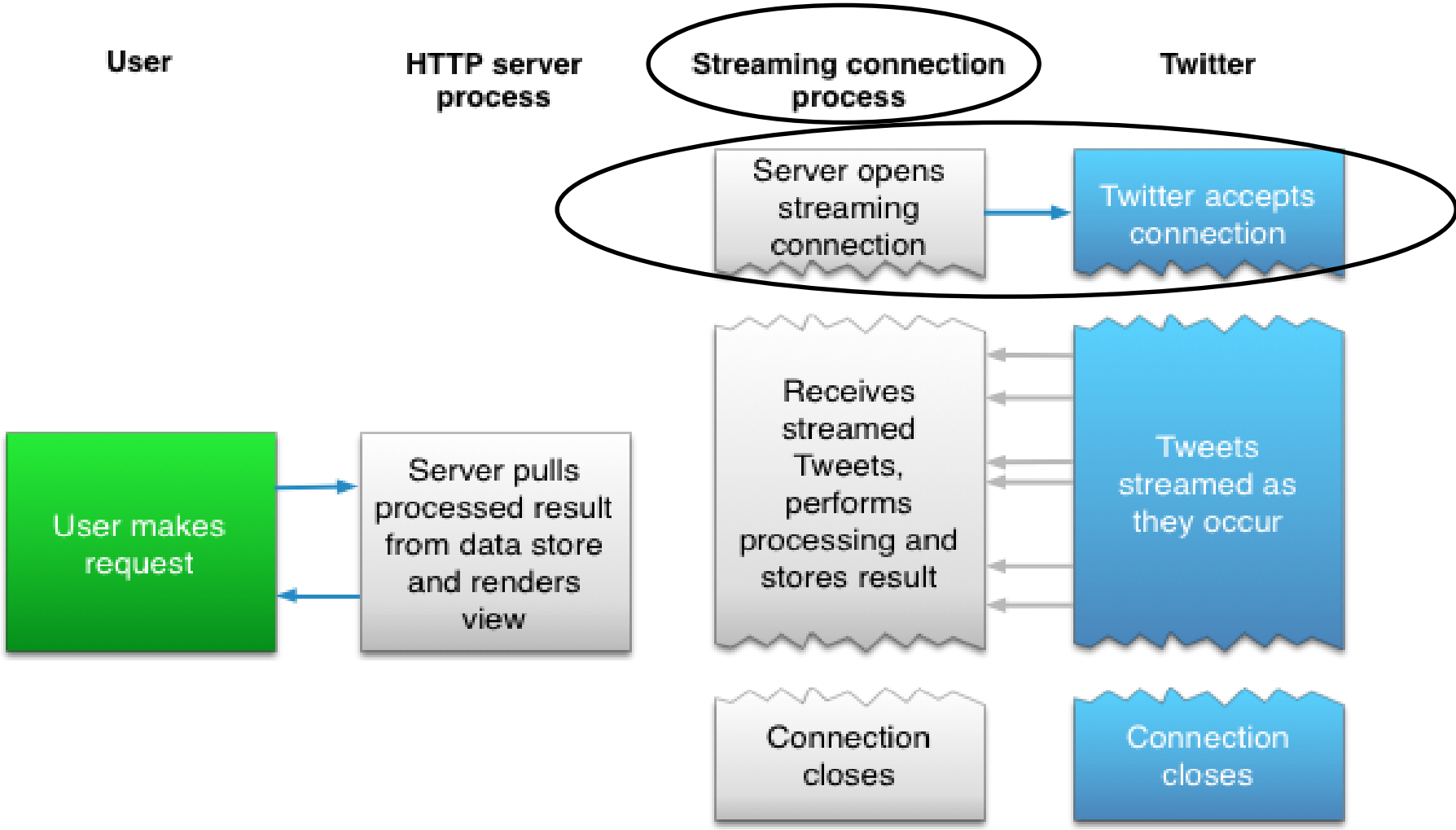
October 6, 2020

Twitter Streaming API

- The **Streaming API** is designed for developers with data intensive needs
  - Most suited for building data mining products or conducting analytics research
  - It allows for large quantities of keywords to be specified and tracked, retrieving geo-tagged tweets from a certain region, or have the public statuses of a user set returned
  - As mentioned before, if your Search API based apps are hitting the rate limits, move over to the Streaming API
    - This requires you to establish a long-lived HTTP connection and maintain that connection

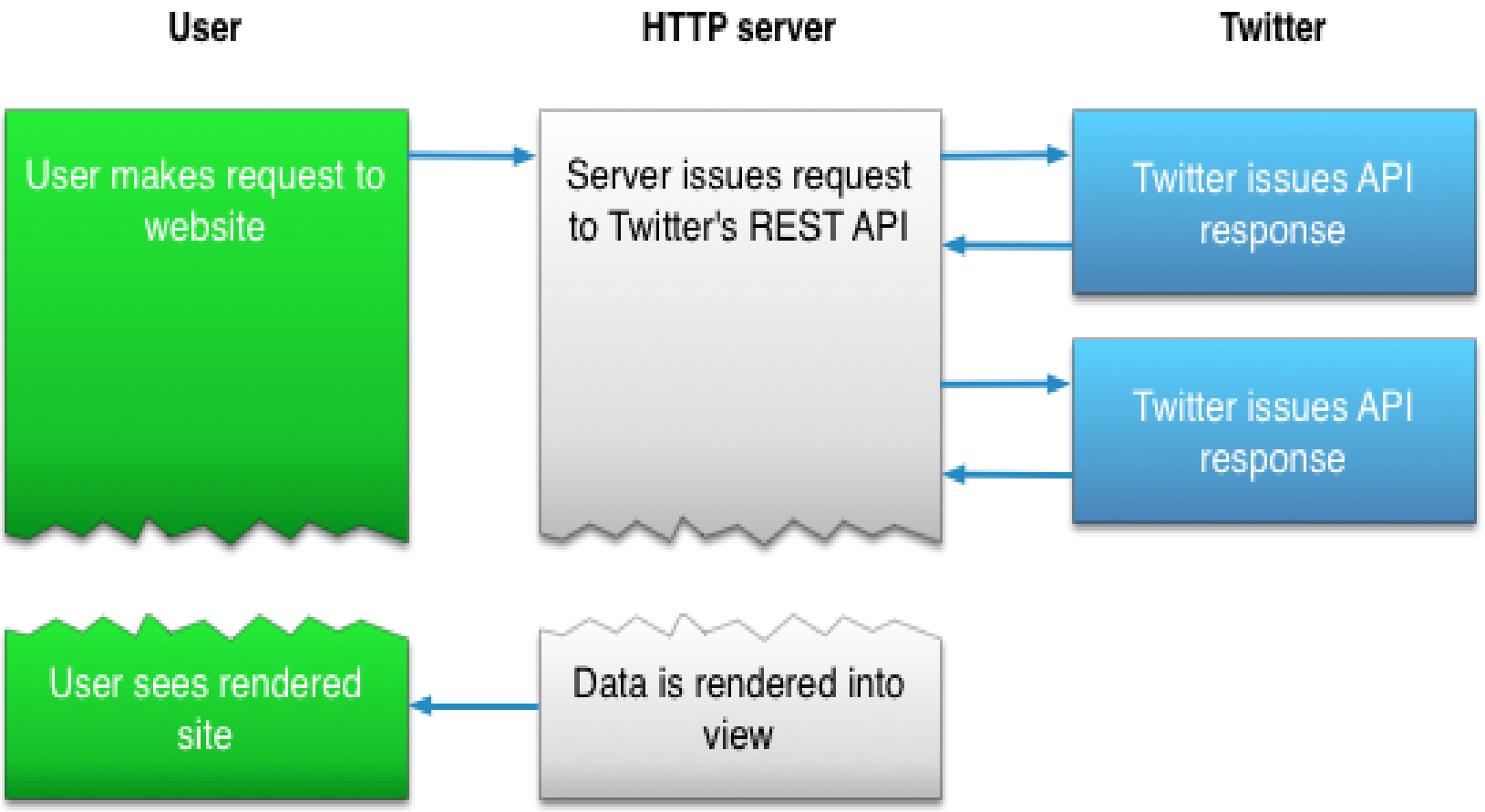
Twitter Streaming API

Connecting to the streaming API requires keeping a persistent HTTP connection open.



Twitter REST API

Rest API doesn't support persistent HTTP connection. (Stateless)



Filter realtime Tweets

A related endpoint is available in Labs. If you haven't done so, we invite you to join Labs to provide feedback. For more details, see Filtered stream.

The Twitter API platform offers two options for streaming realtime Tweets. Each option offers a varying number of filters and filtering capabilities - see the below summary for more details:

API	Category	Number of filters	Filtering operators	Rule management
statuses/filter	Standard	400 keywords, 5,000 userids and 25 location boxes	Standard operators	One filter rule on one allowed connection, disconnection required to adjust rule
PowerTrack	Enterprise	Up to 250,000 filters per stream, up to 2,048	Premium operators	Thousands of rules on a single connection, no disconnection needed to

Please note:

We launched a new version of the standard statuses/filter endpoint as part of Twitter API v2: Early Access. If you are currently using this endpoints, you can use our migration materials to start working with the new Twitter API.

POST statuses/filter

Returns public statuses that match one or more filter predicates. Multiple parameters may be specified which allows most clients to use a single connection to the Streaming API. Both GET and POST requests are supported, but GET requests with too many parameters may cause the request to be rejected for excessive URL length. Use a POST request to avoid long URLs.

The track, follow, and locations fields should be considered to be combined with an OR operator. `track=foo&follow=1234` returns Tweets matching "foo" OR created by user 1234.

The default access level allows up to 400 track keywords, 5,000 follow userids and 25 0.1-360 degree location boxes. If you need access to more rules and filtering tools, please apply for enterprise access.

Resource URL

POST statuses/filter

Returns public statuses that match one or more filter predicates. Multiple parameters may be specified which allows most clients to use a single connection to the Streaming API. Both GET and POST requests are supported, but GET requests with too many parameters may cause the request to be rejected for excessive URL length. Use a POST request to avoid long URLs.

The track, follow, and locations fields should be considered to be combined with an OR operator. `track=foo&follow=1234` returns Tweets matching "foo" OR created by user 1234.

The default access level allows up to 400 track keywords, 5,000 follow userids and 25 0.1-360 degree location boxes. If you need access to more rules and filtering tools, please apply for enterprise access.

Resource URL

`https://stream.twitter.com/1.1/statuses/filter.json`

Resource Information

Response formats	JSON
Requires authentication?	Yes (user context only)
Rate limited?	Yes

Parameters

Parameters

Name	Required	Description
follow	optional	A comma separated list of user IDs, indicating the users to return statuses for in the stream. See follow for more information.
track	optional	Keywords to track. Phrases of keywords are specified by a comma-separated list. See track for more information.
locations	optional	Specifies a set of bounding boxes to track. See locations for more information.
delimited	optional	Specifies whether messages should be length-delimited. See delimited for more information.
stall_warnings	optional	Specifies whether stall warnings should be delivered. See stall_warnings for more information.

Example Request

```
None https://stream.twitter.com/1.1/statuses/filter.json?track=twitter
```

Using the Streaming API: Filter

```
# Finding topics of interest by using the filtering capabilities it offers.
# Describe when to use search versus when to use streaming api - two different
# use cases.
import twitter
# Query terms
q = 'COVID-19 vaccine,CDC,FDA,Pfizer,Moderna' #comma separated list of terms
# Returns an instance of twitter.Twitter
twitter_api = oauth_login()
# Reference the self.auth parameter
twitter_stream = twitter.TwitterStream(auth=twitter_api.auth)
# See https://developer.twitter.com/en/docs/tweets/filter-realtime/overview
stream = twitter_stream.statuses.filter(track=q)
for tweet in stream:
    print tweet['text'] # Save to a file or database in a particular collection
```

Administrator: Command Prompt - python

```
>>>
>>>
>>> help(twitter.stream)
Help on module twitter.stream in twitter:

NAME
    twitter.stream - # encoding: utf-8

CLASSES
    builtins.object
        HttpChunkDecoder
        JsonDecoder
        SockReader
        Timer
        TwitterJSONIter
    twitter.api.TwitterCall(builtins.object)
        TwitterStream

    class HttpChunkDecoder(builtins.object)
     |  Methods defined here:
     |
     |  __init__(self)
     |      Initialize self.  See help(type(self)) for accurate signature.
     |
     |  decode(self, data)
     |
     |  ----------------------------------------------------------------------
     |  Data descriptors defined here:
     |
     |  __dict__
     |      dictionary for instance variables (if defined)
     |
     |  __weakref__
     |      list of weak references to the object (if defined)

    class JsonDecoder(builtins.object)
     |  Methods defined here:
```

Administrator: Command Prompt - python

```
    |  __init__(self)

>>> help(twitter.TwitterStream)
Help on class TwitterStream in module twitter.stream:

class TwitterStream(twitter.api.TwitterCall)
   |  The TwitterStream object is an interface to the Twitter Stream
   |  API. This can be used pretty much the same as the Twitter class
   |  except the result of calling a method will be an iterator that
   |  yields objects decoded from the stream. For example::
   |
   |      twitter_stream = TwitterStream(auth=OAuth(...))
   |      iterator = twitter_stream.statuses.sample()
   |
   |      for tweet in iterator:
   |          # ...do something with this tweet...
   |
   |  Per default the ``TwitterStream`` object uses
   |  [public streams](https://dev.twitter.com/docs/streaming-apis/streams/public).
   |  If you want to use one of the other
   |  [streaming APIs](https://dev.twitter.com/docs/streaming-apis), specify the URL
   |  manually:
   |
   |  - [Public streams](https://dev.twitter.com/docs/streaming-apis/streams/public): stream.twitter.com
   |  - [User streams](https://dev.twitter.com/docs/streaming-apis/streams/user): userstream.twitter.com
   |  - [Site streams](https://dev.twitter.com/docs/streaming-apis/streams/site): sitestream.twitter.com
   |
   |  Note that you require the proper
   |  [permissions](https://dev.twitter.com/docs/application-permission-model) to
   |  access these streams. E.g. for direct messages your
   |  [application](https://dev.twitter.com/apps) needs the "Read, Write & Direct
   |  Messages" permission.
   |
   |  The following example demonstrates how to retrieve all new direct messages
   |  from the user stream::
   |
   |      auth = OAuth(
```

Using the Streaming API: Filter

- A **comma-separated list of phrases** which will be used to determine what Tweets will be delivered on the stream. (‘the,twitter’ = the **OR** twitter)
- A **phrase** may be one or more terms separated by spaces, and a phrase will match if all of the terms in the phrase are present in the Tweet, regardless of order and ignoring case. (e.g. ‘the twitter’ = the **AND** twitter)
- Each phrase must be between 1 and 60 bytes, inclusive.
- Exact matching of **phrases** (equivalent to quoted phrases in most search engines) is **not supported**.
- Punctuation and special characters will be considered part of the term they are adjacent to. (‘hello.’ != ‘hello’)
- Punctuation is not considered to be part of a #hashtag or @mention, so a track term containing punctuation will **not** match either #hashtags or @mentions.
- **UTF-8** characters will match exactly, even in cases where an "equivalent" ASCII character exists. (‘touché’ != ‘touche’)
- Non-space separated languages, such as **CJK** are currently unsupported.

Track examples:

Parameter value	Will match...	Will not match...
Twitter	**TWITTER** twitter “Twitter” twitter. #twitter @twitter http://twitter.com	**TwitterTracker** #newtwitter
Twitter’s	I like Twitter’s new design	Someday I’d like to visit @Twitter’s office
twitter api,twitter streaming	**The Twitter API is awesome** The twitter streaming service is fast Twitter has a streaming API	I’m new to Twitter
example.com	Someday I will visit example.com	There is no example.com/foobarbaz
example.com/foobarbaz	**example.com/foobarbaz** www.example.com/foobarbaz	example.com
www.example.com/foobarbaz		www.example.com/foobarbaz
example com	**example.com** www.example.com foo.example.com foo.example.com/bar I hope my startup isn’t merely another example	

For Search API

standard search:

Operator	Finds Tweets...
watching now	containing both “watching” and “now”. This is the default operator.
“happy hour”	containing the exact phrase “happy hour”.
love OR hate	containing either “love” or “hate” (or both).
beer -root	containing “beer” but not “root”.
#haiku	containing the hashtag “haiku”.
from:interior	sent from Twitter account “interior”.
list:NASA/astronauts-in-space-now	sent from a Twitter account in the NASA list astronauts-in-space-now
to:NASA	a Tweet authored in reply to Twitter account “NASA”.
@NASA	mentioning Twitter account “NASA”.
politics filter:safe	containing “politics” with Tweets marked as potentially sensitive removed.
puppy filter:media	containing “puppy” and an image or video.
puppy -filter:retweets	containing “puppy”, filtering out retweets
puppy filter:native_video	containing “puppy” and an uploaded video, Amplify video, Periscope, or Vine.
puppy filter:periscope	containing “puppy” and a Periscope video URL.

Parameters

Name	Required	Description
follow	optional	A comma separated list of user IDs, indicating the users to return statuses for in the stream. See follow for more information.
track	optional	Keywords to track. Phrases of keywords are specified by a comma-separated list. See track for more information.
locations	optional	Specifies a set of bounding boxes to track. See locations for more information.
delimited	optional	Specifies whether messages should be length-delimited. See delimited for more information.
stall_warnings	optional	Specifies whether stall warnings should be delivered. See stall_warnings for more information.

Example Request

```
None https://stream.twitter.com/1.1/statuses/filter.json?track=twitter
```

Using the Streaming API: Filter by IDs

```
import twitter
# Returns an instance of twitter.Twitter
twitter_api = oauth_login()

# Reference the self.auth parameter
twitter_stream = twitter.TwitterStream(auth=twitter_api.auth)
# See https://developer.twitter.com/en/docs/tweets/filter-realtime/overview

# ladygaga's user id: 14230524
id = 14230524

stream = twitter_stream.statuses.filter(follow=id)  # or follow=str(id)
for tweet in stream:
    print tweet['text'] # Save to a file or database in a particular collection
```

Using the Streaming API: Filter by IDs

- You could also use a **comma-separated** list of **user IDs**, indicating the users whose Tweets should be delivered on the stream.

```
import twitter
twitter_api = oauth_login()
twitter_stream = twitter.TwitterStream(auth=twitter_api.auth)

# ladygaga's user id: 14230524; justinbieber: 27260086; katyperry: 21447363
ids = '14230524,27260086,21447363'
stream = twitter_stream.statuses.filter(follow=ids)

for tweet in stream:
    print tweet['text'] # Save to a file or database in a particular collection
```

Using the Streaming API: Filter by IDs

- Following protected users is not supported.
- For each user specified, the stream **will** contain:
  - Tweets created by the user.
  - Tweets which are retweeted by the user.
  - Replies to any Tweet created by the user.
  - Retweets of any Tweet created by the user.
  - Manual replies created without pressing a reply button:
    - e.g. “@twitterapi I agree”
- The stream **will not** contain:
  - Tweets mentioning the user (e.g. “Hello @twitterapi!”).
  - Manual retweets created without pressing a Retweet button
    - e.g. “RT @twitterapi The API is great”
  - Tweets by protected users.

https://developer.twitter.com/en/docs/twitter-api/v1/tweets/filter-realtime/guides/basic-stream-parameters

Parameters

Name	Required	Description
follow	optional	A comma separated list of user IDs, indicating the users to return statuses for in the stream. See follow for more information.
track	optional	Keywords to track. Phrases of keywords are specified by a comma-separated list. See track for more information.
locations	optional	Specifies a set of bounding boxes to track. See locations for more information.
delimited	optional	Specifies whether messages should be length-delimited. See delimited for more information.
stall_warnings	optional	Specifies whether stall warnings should be delivered. See stall_warnings for more information.

Example Request

```
None https://stream.twitter.com/1.1/statuses/filter.json?track=twitter
```

Streaming API: Filter by Locations

- A comma-separated list of **longitude,latitude pairs** specifying a set of bounding boxes to filter Tweets by.
- On geolocated Tweets falling within the requested bounding boxes will be included Unlike the Search API, the user's location field is not used to filter tweets.
- Each bounding box should be specified as a pair of longitude and latitude pairs, with the **southwest** corner of the bounding box coming first. For example:

Parameter value	Tracks Tweets from...
-122.75,36.8,-121.75,37.8	San Francisco
-74,40,-73,41	New York City
-122.75,36.8,-121.75,37.8,-74,40,-73,41	San Francisco OR New York City
-180,-90,180,90	Any geotagged Tweet

https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters

Streaming API: Filter by Locations

```
import twitter
# Returns an instance of twitter.Twitter
twitter_api = oauth_login()

# Reference the self.auth parameter
twitter_stream = twitter.TwitterStream(auth=twitter_api.auth)

# NYC coordinates: -74,40,-73,41
# San Francisco: -122.75,36.8,-121.75,37.8
loc = '-122.75,36.8,-121.75,37.8,-74,40,-73,41'
stream = twitter_stream.statuses.filter(locations=loc)
for tweet in stream:
    print tweet['text'] # Save to a file or database in a particular collection
```

Streaming API: Filter by Locations

- Bounding boxes do not act as filters for other filter parameters.
  - For example:

track=Twitter**&**locations=-122.75,36.8,-121.75,37.8

would match any tweets containing the term Twitter (even non-geo tweets) OR coming from the San Francisco area.

Please note:

We launched a new version of the standard statuses/filter endpoint as part of Twitter API v2: Early Access. If you are currently using this endpoints, you can use our migration materials to start working with the new Twitter API.

POST statuses/filter

Returns public statuses that match one or more filter predicates. Multiple parameters may be specified which allows most clients to use a single connection to the Streaming API. Both GET and POST requests are supported, but GET requests with too many parameters may cause the request to be rejected for excessive URL length. Use a POST request to avoid long URLs.

The track, follow, and locations fields should be considered to be combined with an OR operator. `track=foo&follow=1234` returns Tweets matching "foo" OR created by user 1234.

The default access level allows up to 400 track keywords, 5,000 follow userids and 25 0.1-360 degree location boxes. If you need access to more rules and filtering tools, please apply for enterprise access.

Resource URL

Streaming API: Filter by Locations

- The streaming API uses the following heuristic to determine whether a given Tweet falls within a bounding box:
  - If the **coordinates field** is populated, the values there will be tested against the bounding box. (next 3 slides)
  - If coordinates field is empty but the **place field** is populated, the region defined in place is checked for intersection against the locations bounding box. Any overlap will match.
  - If none of the rules listed above match, the Tweet does not match the location query.

Data dictionary

Overview

Overview contents

Tweet Object

Tweets are the basic atomic building block of all things Twitter. Tweets are also known as “status updates.” The Tweet object has a long list of ‘root-level’ attributes, including fundamental attributes such as `id`, `created_at`, and `text`. Tweet objects are also the ‘parent’ object to several child objects. Tweet child objects include `user`, `entities`, and `extended_entities`. Tweets that are geo-tagged will have a `place` child object.

When the following Tweet is rendered in JSON:

coordinates	Coordinates	*Nullable.* Represents the geographic location of this Tweet as reported by the user or client application. The inner coordinates array is formatted as geoJSON (longitude first, then latitude). Example: `"coordinates": { "coordinates": [ -75.14310264, 40.05701649 ], "type":"Point" }`
place	Places	*Nullable* When present, indicates that the tweet is associated (but not necessarily originating from) a Place . Example: `"place": { "attributes":{}, "bounding_box":`

place	Places	*Nullable* When present, indicates that the tweet is associated (but not necessarily originating from) a Place . Example:


```
  "place":
{
  "attributes":{},
   "bounding_box":
   {
      "coordinates":
      [[
            [-77.119759,38.791645],
            [-76.909393,38.791645],
            [-76.909393,38.995548],
            [-77.119759,38.995548]
      ]],
      "type":"Polygon"
   },
    "country":"United States",
    "country_code":"US",
    "full_name":"Washington, DC",
    "id":"01fbe706f872cb32",
    "name":"Washington",
    "place_type":"city",
    "url":"http://api.twitter.com/1/geo/id/0172cb32.
json"
}
```

Search API: Filter by Locations

geocode

- Returns tweets by users located within a given radius of the given latitude/longitude.
- The location is preferentially taking from the Geotagging API, but will fall back to their Twitter profile.
- The parameter value is specified by "latitude,longitude,radius", where radius units must be specified as either "mi" (miles) or "km" (kilometers).
  - **Example Values**: 37.781157,-122.398720,10mi

Search API: Filter by Locations

```
import twitter
import json
from TwitterCookbook import oauth_login, twitter_search
twitter_api = oauth_login()
q = ‘Syracuse University'
results = twitter_api.search.tweets(q=q, count=100, geocode='43.0,-76.1,10mi')['statuses']
# results = twitter_search(twitter_api, q, max_results=100, geocode='43.0,-76.1,10mi')
# Show one sample search result by slicing the list...
# print json.dumps(results[0], indent=1)
tweets = [(r['text'], r['created_at']) for r in results]
for i, t in enumerate(tweets):
    try:
        print(i, t)
    except:
        pass
```

delimited

This parameter may be used on all streaming endpoints, unless explicitly noted.

Setting this to the string length indicates that statuses should be delimited in the stream, so that clients know how many bytes to read before the end of the status message. Statuses are represented by a length, in bytes, a newline, and the status text that is exactly length bytes. Note that "keep-alive" newlines may be inserted before each length.

As an example, consider this response to a request to https://stream.twitter.com/1.1/statuses/filter.json?delimited=length&track=twitterapi:

The 1953 indicates how many bytes to read off of the stream to get the rest of the Tweet (including rn). The next length delimiter will occur exactly after 1953 bytes.

stall_warnings

This parameter may be used on all streaming endpoints, unless explicitly noted.

Setting this parameter to the string true will cause periodic messages to be delivered if the client is in danger of being disconnected. These messages are only sent when the client is falling behind, and will occur at a maximum rate of about once every 5 minutes. This parameter is most appropriate for clients with high-bandwidth connections.

Such warning messages will look like:

```
{
 "warning" : {
   "code" : "FALLING_BEHIND" ,
   "message" : "Your connection is falling behind and messages are being queued for deli
   "percent_full" : 60
```

Sample realtime Tweets

Overview Guides API reference

Overview contents

Sample stream

Decahose stream

Streaming likes

Please note:

We launched a new version of the standard statuses/sample endpoint as part of Twitter API v2: Early Access. If you are currently using any of these endpoints, you can use our migration materials to start working with the newer endpoint.

GET statuses/sample

Returns a small random sample of all public statuses. The Tweets returned by the default access level are the same, so if two different clients connect to this endpoint, they will see the same Tweets.

Sampling Tweets

stream = twitter_stream.statuses.filter(track=q)

stream = twitter_stream.statuses.sample()

9.8: Sampling the Twitter Firehose

Problem

- You want to analyze what people are tweeting about right now from a real-time stream of tweets as opposed to querying the Search API for what might be slightly dated information.
- Or, you want to begin accumulating nontrivial amounts of data about a particular topic for later analysis.

Solution

Use Twitter's Streaming API to sample public data from the Twitter **firehose**.

9.8: Sampling the Twitter Firehose

Discussion

- Twitter makes up to 1% of all tweets available in real time through a random sampling technique that represents the larger population of tweets and exposes these tweets through the Streaming API.
- Unless you want to go to a third-party provider such as GNIP *(bought by Twitter, see next slide)* or DataSift (*discontinued*), which may actually be well worth the cost in many situations, this is about as good as it gets
- For a broad enough topic, actually storing all of the tweets you sample could quickly become more of a problem than you might think.
- But even access to up to 1% of all public tweets is significant.

Enterprise data

Unleash the power of Twitter data

Twitter's enterprise API platform delivers real-time and historical social data to power your business at scale.

Our enterprise solutions are customized with predictable pricing to meet the needs of your business. The annual contracts include account management. To get started, apply for enterprise access, and our team will be in touch.

Apply for enterprise access

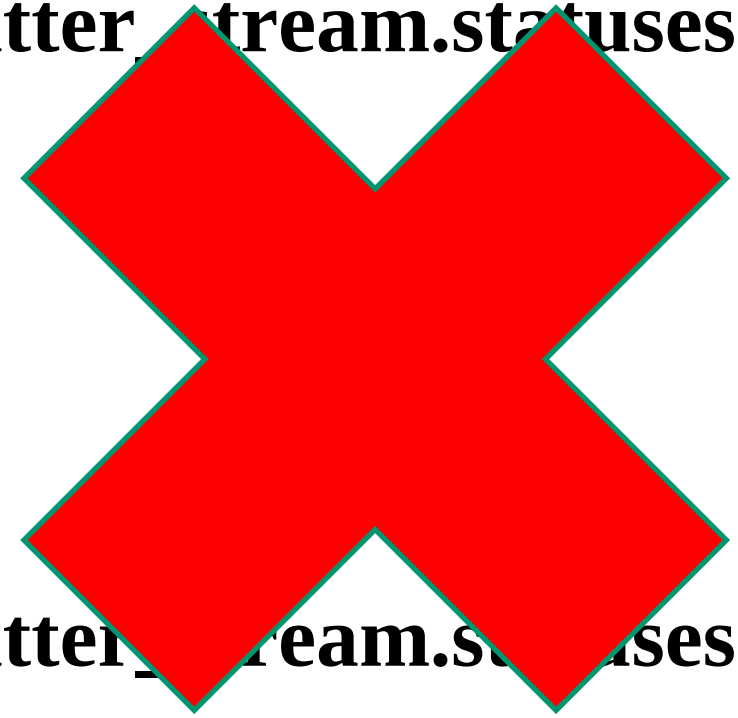
Firehose

stream = twitter_stream.statuses.sample()

stream = twitter_stream.statuses.firehose()

Firehose

stream = twitter_stream.statuses.sample()



stream = twitter_stream.statuses.firehose()

Filter realtime Tweets

A related endpoint is available in Labs. If you haven't done so, we invite you to join Labs to provide feedback. For more details, see Filtered stream.

The Twitter API platform offers two options for streaming realtime Tweets. Each option offers a varying number of filters and filtering capabilities - see the below summary for more details:

API	Category	Number of filters	Filtering operators	Rule management
statuses/filter	Standard	400 keywords, 5,000 userids and 25 location boxes	Standard operators	One filter rule on one allowed connection, disconnection required to adjust rule
PowerTrack	Enterprise	Up to 250,000 filters per stream, up to 2,048 characters each	Premium operators	Thousands of rules on a single connection, no disconnection needed to add/remove rules using Rules API