# Sam Moghadam

# Investigative Techniques

January 30, 2018

Assignment 2

**API: Spotify Web API**

Spotify is a popular music streaming service. The Spotify Web API allows applications to get data from Spotify’s catalogue of artists/albums/tracks, and to get and write data related to users’ playlists and saved music. The API returns metadata in JSON format.

To obtain a Spotify API key, go to [developer.spotify.com](http://developer.spotify.com) (or [beta.developer.spotify.com](http://beta.developer.spotify.com) for a newer version of the page), and sign up for a developer account. Next, click on “create an app” and fill out the form. Once you have created your app, click on it to see your Client ID and Client Secret, which serve as your API key and secret key, respectively.

The base address of the API is <https://api.spotify.com>. Endpoints appended to that address allow for calls to the API. Requests can be in the form of: GET (retrieve resources), POST (create resources), PUT (change or replace resources), DELETE (delete resources).

Spotify API requests require an access token (the Client ID and Client Secret). For requests related to a specific user, the user must grant permission as well.

The API has rate limiting based on the client ID, though it is not clear precisely what the rate limit is. The status code 429 occurs when too many requests are sent to the API server. One way to fetch a greater volume of information without running into the rate limit is to use the API’s “get several” endpoints, which allow the client to request data about multiple artists/albums/tracks rather than individual ones.

For this assignment, I decided to use Spotipy, a popular third-party Python library for the Spotify Web API. My query examples will therefore be in the form of Python code using this library.

The Spotify API documentation can be found at <https://developer.spotify.com/web-api/>. The documentation includes tutorials and even has an interactive API console, which can be found at <https://developer.spotify.com/web-api/console/>.

——————

**Query 1: Return the three most popular tracks in [some country] by [some artist], along with a Spotify link to the track.**

#####

# query1.py

# Returns a list of top tracks by an artist.

# Adapted from: https://stackoverflow.com/questions/46966932/how-to-work-in-spotify-python-api

#####

import sys

from spotipy.oauth2 import SpotifyClientCredentials

import spotipy

import pprint

# Access tokens

client\_id = “[my client ID]”

client\_secret = “[my client secret]”

client\_credentials\_manager = SpotifyClientCredentials(client\_id=client\_id, client\_secret=client\_secret)

sp = spotipy.Spotify(client\_credentials\_manager=client\_credentials\_manager)

# Hard-coding the artist to be Radiohead and the

# country to be the U.S.

artist\_id = 'spotify:artist:4Z8W4fKeB5YxbusRsdQVPb'

country = 'US'

# Getting information from the API

results = sp.artist\_top\_tracks(artist\_id=artist\_id, country=country)

# Print the top 3 tracks and provide a URL to an mp3 sample

for track in results['tracks'][:3]:

print ('TRACK: ' + track['name'])

print ('LINK: ' + track['preview\_url'])

print()

———

**Result:**

TRACK: Creep

LINK: https://p.scdn.co/mp3-preview/e7eb60e9466bc3a27299ea8803aadf4fa9cf795c?cid=0d95a04638084b27a789a328a8b96362

TRACK: Karma Police

LINK: https://p.scdn.co/mp3-preview/5a09f5390e2862af1687f3671db3d6f84a813a45?cid=0d95a04638084b27a789a328a8b96362

TRACK: High And Dry

LINK: <https://p.scdn.co/mp3-preview/7cc39826315239401b8bac4317c14c35ecdbcd58?cid=0d95a04638084b27a789a328a8b96362>

—————

The results here consist of track titles and an accompanying link to an mp3 sample for each track.

In this example, there are a few parameters that I have hard-coded:

* + - * artist\_id: The unique ID string for the particular artist whose tracks I want to see (in this case, Radiohead)
      * country: The country for which the data is returned (in this case, these are the three most popular Radiohead songs, by play count, in the U.S.)

In this case, the query parameters are very straightforward. I’m interested in this particular band and live in this country, and I want to see their top-performing tracks on the streaming service. Of course, I could see the same thing by simply logging into Spotify as a user, but this still serves as a basic example of a GET call to the API.

——————

**Query 2: Return all but the [x] most recent singles by [some artist]; return no more than [y] singles.**

#####

# query2.py

# Returns releases by an artist

# Adapted from: https://github.com/plamere/spotipy/blob/master/examples/simple1.py

#####

import sys

from spotipy.oauth2 import SpotifyClientCredentials

import spotipy

# Access tokens

client\_id = “[my client ID]”

client\_secret = “[my client secret[“

client\_credentials\_manager = SpotifyClientCredentials(client\_id=client\_id, client\_secret=client\_secret)

sp = spotipy.Spotify(client\_credentials\_manager=client\_credentials\_manager)

# Hard-coding the artist to be Radiohead and the

# release type to be singles

artist\_id = 'spotify:artist:4Z8W4fKeB5YxbusRsdQVPb'

release\_type = 'single'

# Don't include the 10 latest releases;

# don't return more than 20 releases

results = sp.artist\_albums(artist\_id, release\_type, limit=20, offset=10)

releases = results['items']

# Get the results from the API

while results['next']:

results = sp.next(results)

releases.extend(results['items'])

# Print all release names

for release in releases:

print((release['name']))

———

**Result:**

Supercollider / The Butcher

Harry Patch (In Memory Of)

Spectre

Daydreaming

Burn the Witch

The Daily Mail / Staircase

These Are My Twisted Words

Knives Out

Pyramid Song

No Surprises

Karma Police

Paranoid Android

Street Spirit (Fade Out)

Just

High & Dry / Planet Telex

Fake Plastic Trees

My Iron Lung

Anyone Can Play Guitar

Creep

—————

The result is simply a list of the names of the singles returned from the API server.

This example features the artist\_id parameter, as well as two new parameters:

* + - * album\_type: On Spotify, there are two types of releases: “albums” (i.e., full-length albums) and “singles” (i.e., singles or EPs, which are the essentially very short albums).
      * limit: An integer that limits the number of results returned.
      * offset: An index of where to begin the results. In this case, I know that Spotify lists releases in reverse-chronological order (latest releases are first). Setting the offset to 10, as I did here, is equivalent of saying, “Give me all but the 10 latest singles.”

Again, I have hard-coded the artist\_id for Radiohead, and I have hard-coded the album\_type as “single,” so that the result includes all available singles/EPs by Radiohead. This example is mainly used to demonstrate that we can drill down into types of results. In this case, there are multiple types of albums, and we might only be interested in full-length releases vs. singles/EPs, depending on what we’re trying to accomplish. Furthermore, we might only be interested in an artist’s earlier or later work, which is where the offset comes in.

——————

**Query 3: Provide an acoustic analysis of [some track].**

#####

# query3.py

# Returns audio analysis of a track

# Adapted from: https://github.com/plamere/spotipy/blob/master/examples/audio\_analysis\_for\_track.py

#####

from \_\_future\_\_ import print\_function # (at top of module)

from spotipy.oauth2 import SpotifyClientCredentials

import json

import spotipy

import time

import sys

# Access tokens

client\_id = “[my client ID]”

client\_secret = “[my client secret]”

client\_credentials\_manager = SpotifyClientCredentials(client\_id=client\_id, client\_secret=client\_secret)

sp = spotipy.Spotify(client\_credentials\_manager=client\_credentials\_manager)

# Hard-coding the track to be "The Present Tense," by Radiohead

track\_id = 'spotify:track:4eruRiSfDY1jdT03hjyi0i'

# Get audio analysis from API

analysis = sp.audio\_analysis(track\_id)

# Print analysis

print(json.dumps(analysis, indent=4))

———

**Result (sample):**

{

"start": 281.14277,

"duration": 0.29905,

"confidence": 0.169,

"loudness\_start": -16.366,

"loudness\_max\_time": 0.04236,

"loudness\_max": -12.539,

"pitches": [

0.153,

1.0,

0.35,

0.219,

0.058,

0.034,

0.04,

0.07,

0.209,

0.102,

0.049,

0.071

],

"timbre": [

45.413,

-76.298,

-65.215,

-37.905,

24.65,

-40.693,

-30.195,

-1.54,

21.297,

-0.721,

-3.789,

1.202

]

},

—————

This example features the track\_id parameter, which, like the artist\_id parameter, is a unique string identifier for a particular track on Spotify.

In this case, the track has been hard-coded to “The Present Tense,” by Radiohead.

Although this query is very simple in terms of its parameter inputs, it provides the most interesting output of the three examples. Here, we see an acoustic breakdown of moments in the song (only one such 0.3-second snippet is shown above, but the actual output will include hundreds of such moments throughout the track). We see such features as the pitches and timbres used during that piece of the track, as well as the maximum decibel level of the snippet in question, the duration of the snippet, etc. It’s particularly interesting to get such information from the API, because, unlike in the other two query examples, this is *not* something the user can see by simply browsing Spotify. This kind of information could be used to do very interesting, granular musical analyses of different tracks.