From left to right: Laurent Weichberger, Russell Hanson, Sascha Ishikawa, Asa Wilks, James Liu, Angel Martinez, & Scot Hickey. Photo copyright (c) 2017 by L. Weichberger

Apache Spark Streaming with Twitter (and Python)

Published on May 28, 2017 Edit article | View stats

Laurent Weichberger
Changing the world one Big Data client at a time ....
31 articles

1,533 30 4

Last week during my Hortonworks training at the RAND Corp. we created a spontaneous lab exercise to integrate Python, Twitter and Spark Streaming. I had done this work before when I worked at Databricks, using Java and Twitter4j (I later refactored it with Scala), so I knew what to do. However, I had never tried to get it working with Python, so it was a challenge figuring out how to do it with Python. The project requirements were as follows:

Twitter Use Case 1:

1.a. Hook up Apache Spark Streaming to an incoming Twitter Stream.

1.b. Filter the stream for keywords.

2.a. Capture the tweets and filter them by a specific language (e.g. Arabic).

Messaging

2.b. Use the Tweet's "lang" field for this purpose. As it turns out, it was a little difficult to parse the incoming JSON (we found it vague regarding what the Tweepy code we had leveraged was sending over to us on localhost) -- it seemed that more than just a Twitter Status (e.g. that is Twitter4j lingo) came over. So, we needed a filter using key lookup for lang, and check for existence of lang key (otherwise we saw some weird side-effects).

4.a. Write that final filtered stream out to the filesystem using saveAsTextFile().

4.b. After writing out, ensure that the Arabic characters appear in a legible format, so that an Arabic speaking individual can easily open and read it.

5. Test: Arabic speaking individual opens, reads and translates the tweets.

The first student to finish was Mr. Sascha Ishikawa so we share his integrated code solution here. It was in fact a group effort, as we had roundtable discussions all along the way. We all have code involved in the solution.

In order to solve this use case we leveraged the following:

1. A package named "tweepy" which we found on a Python Twitter developer site.

2. Laurent's Twitter developer credentials to quickly grab the Twitter stream. It is easy to register as a developer at Twitter to get your own credentials.

3. The json.loads() method to find that "lang" key in the JSON version of the tweet.

4. Laurent's original base Python Spark Streaming code:

```
# From within pyspark or send to spark-submit:

from pyspark.streaming import StreamingContext

ssc = StreamingContext(sc, 5) # 5 second batch interval

IP = "localhost"        # Replace with your stream IP
Port = 5555                     # Replace with your stream port

lines = ssc.socketTextStream(IP, Port)
lines.pprint()         # Print tweets we find to the console

ssc.start()                         # Start reading the stream
ssc.awaitTermination() # Wait for the process to terminate
```

Messaging

Here is the refactored Python code we wrote as a team, and remember we leveraged anc modified the TweetRead.py from a site we found:

```
# TweetRead.py
# This first python script doesn't use Spark at all:
import os
import tweepy
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener
import socket
import json

consumer_key    = os.environ['TWITTER_CONSUMER_KEY']
consumer_secret = os.environ['TWITTER_CONSUMER_SECRET']
access_token    = os.environ['TWITTER_ACCESS_TOKEN']
access_secret   = os.environ['TWITTER_ACCESS_SECRET']

class TweetsListener(StreamListener):

    def __init__(self, csocket):
        self.client_socket = csocket

    def on_data(self, data):
        try:
            print(data.split('\n'))
            self.client_socket.send(data)
            return True
        except BaseException as e:
            print("Error on_data: %s" % str(e))
        return True

    def on_error(self, status):
        print(status)
        return True

def sendData(c_socket):
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)

    twitter_stream = Stream(auth, TweetsListener(c_socket))
    twitter_stream.filter(track=['trump'])

if __name__ == "__main__":
    s = socket.socket()     # Create a socket object
    host = "localhost"      # Get local machine name
    port = 5555             # Reserve a port for your service.
    s.bind((host, port))    # Bind to the port

    print("Listening on port: %s" % str(port))

    s.listen(5)                 # Now wait for client connection.
    c, addr = s.accept()        # Establish connection with client.

    print( "Received request from: " + str( addr ) )

    sendData( c )
```

Run that TweetRead.py shown above first. It just waits on localhost:5555 until the next script runs. The next Python script we saved as SparkDemo.py:

```
# SparkDemo.py
# This code is copyright (c) 2017 by Laurent Weichberger.
```

Messaging

```
# Authors: Laurent Weichberger, from Hortonworks and,
# from RAND Corp: James Liu, Russell Hanson, Scot Hickey,
# Angel Martinez, Asa Wilks, & Sascha Ishikawa
# This script does use Apache Spark. Enjoy...
# This code was designed to be run as: spark-submit SparkDemo.py

import time
import json
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

# Our filter function:
def filter_tweets(tweet):
    json_tweet = json.loads(tweet)
    if json_tweet.has_key('lang'): # When the lang key was not present it
        if json_tweet['lang'] == 'ar':
            return True # filter() requires a Boolean value
    return False

# SparkContext("local[1]") would not work with Streaming bc 2 threads are
sc = SparkContext("local[2]", "Twitter Demo")
ssc = StreamingContext(sc, 10) #10 is the batch interval in seconds
IP = "localhost"
Port = 5555
lines = ssc.socketTextStream(IP, Port)

# When your DStream in Spark receives data, it creates an RDD every batch
# We use coalesce(1) to be sure that the final filtered RDD has only one
# so that we have only one resulting part-00000 file in the directory.
# The method saveAsTextFile() should really be re-named saveInDirectory()
# because that is the name of the directory in which the final part-00000
# We use time.time() to make sure there is always a newly created directo
# it will throw an Exception.

lines.foreachRDD( lambda rdd: rdd.filter( filter_tweets ).coalesce(1).sav

# You must start the Spark StreamingContext, and await process terminatio
ssc.start()
ssc.awaitTermination()
```

Run that SparkDemo.py after the first script, TweetRead.py runs… It works!

Conclusion: In a short time, we were able to get Tweets, filter them by a specific language, and save them to the filesystem for later analysis. We worked as a team of students guided by my instruction. Thank you all for your hard work, you rock!

For more information: Laurent Weichberger, Big Data Bear, Hortonworks: lweichberger@hortonworks.com

Report th

30 Likes

Messaging

+20

4 Comments

Add a comment…

Show previous comments

William Gonzalez 7mo
Hadoop Engineer | Big Data Developer

Very cool, Laurent. I need to try it soon. I have never used Twitter dev creds, so I need to figure that part out. Other than that looks very straight-forward. I love Python! (and Spark).

Like Reply | 1 Like · 1 Reply

Laurent Weichberger 7mo
Changing the world one Big Data client at a time ....

Let me know if you need help getting it to work, it is fun!

Like Reply

Leonie Baelen 3w
Master Business Engineering: Data Analytics - Student aan de/het Universiteit Gent

Hi! I am trying your code for a project with Databricks at my University but I doesn't work. Can I contact you fo some questions?

Like Reply | 5 Replies

Load previous replies

Leonie Baelen 2w
Master Business Engineering: Data Analytics - Student aan de/het Universiteit Gent

Laurent Weichberger I succeeded streaming tweets, but due to the timing issues I had not enough time to include in the project. I will definitly further try it myself :) but thank you very much for your help

Like Reply | 1 Like

Laurent Weichberger 2w
Changing the world one Big Data client at a time ....

Leonie Baelen welcome

Like Reply

Laurent Weichberger
Changing the world one Big Data client at a time ....

More from Laurent Weichberger See all 31 articles

Messaging

Summarization Optimization with HDP Apache Spark & Avro.
Laurent Weichberger on LinkedIn

Store Level Forecasting with Apache Spark Machine Learning
Laurent Weichberger on LinkedIn

Use Case Discovery workshop :: GeoLocation Engine with Apache P…
Laurent Weichberger on LinkedIn

Messaging