Big Data Programming

Project

Hands-On Big Data Streaming Apache Spark at scale

Vaishali Sharma

TASK

We have learned Spark Stream and Kafka in class. However, we never combine them together for stream data processing. Please follow the tutorial from the link below and use all the components to construct a system that requires Twitter API, Spark, Kafka.

Processing Streaming Twitter Data using Kafka and Spark
Or
Hands-On Big Data Streaming, Apache Spark at scale
The authors provide all the necessary code as well as instructions. You can just follow the steps and make it work.

Submission:
1. A comprehensive report that describes what you have learned in this project and what are the challenges.
2. A screenshots based demonstration.
3. The code that you used to build your system.

DELIVERABLES

PROJECT TITLE: Identifying popular hashtags on Twitter using Apache Spark Streaming at Scale

INTRODUCTION

Spark streaming is an extension of the core Spark API that allows users to process real-time data from various sources including Kafka, Flume and Amazon Kinesis, and pushed the processed data to file systems, databases and live dashboards [1]. As we know that twitter provides the functionality of showing one of the popular hashtags at that current time. In this work, a real-life example is demonstrated that analyze and extracts meaningful insights (in particular top10 popular hashtags) from the Twitter data in real-time using Apache Spark Streaming[2], [3],[4]. A simple application that reads online streams from Twitter is created, that processes tweets using Spark Streaming to identify hashtags, and then finally returns top 10 hashtags to the user in the real-time or interactive mode.

TOOLS: Jupyter Notebook, Linux operating system, Apache Spark

---

[1] *https://databricks.com/glossary/what-is-spark-streaming#:~:text=Spark%20Streaming%20is%20an%20extension%20of%20the%20core,out%20to%20file%20systems%2C%20databases%2C%20and%20live%20dashboards*
[2] *https://towardsdatascience.com/hands-on-big-data-streaming-apache-spark-at-scale-fd89c15fa6b0*
[3] *https://www.toptal.com/apache/apache-spark-streaming-twitter*
[4] *https://medium.com/@varunabhi86/finding-popular-hashtags-on-twitter-using-spark-streaming-35592c1fab4f*

TWITTER-SPARK APPLICATION STEPS

Step 1: Twitter Development Account

Firstly, I created a twitter development account by registering on TwitterApplicationPortal. Then, I clicked on "create new app" and created "TwitterFeed_Pro" standalone app in order to fetch the tweets from Twitter (Fig.1).

After this, I went to my newly created app and opened the "Keys and Access Token" tab and generated my access tokens and keys (Fig.2), given below. I then stored this information in text file.

ACCESS_TOKEN, ACCESS_SECRET, CONSUMER_KEY, CONSUMER_SECRET

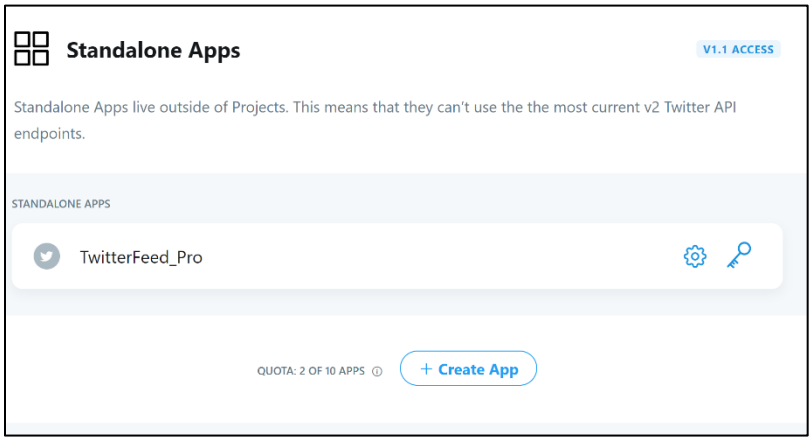


Fig.1 Creation of Standalone Apps

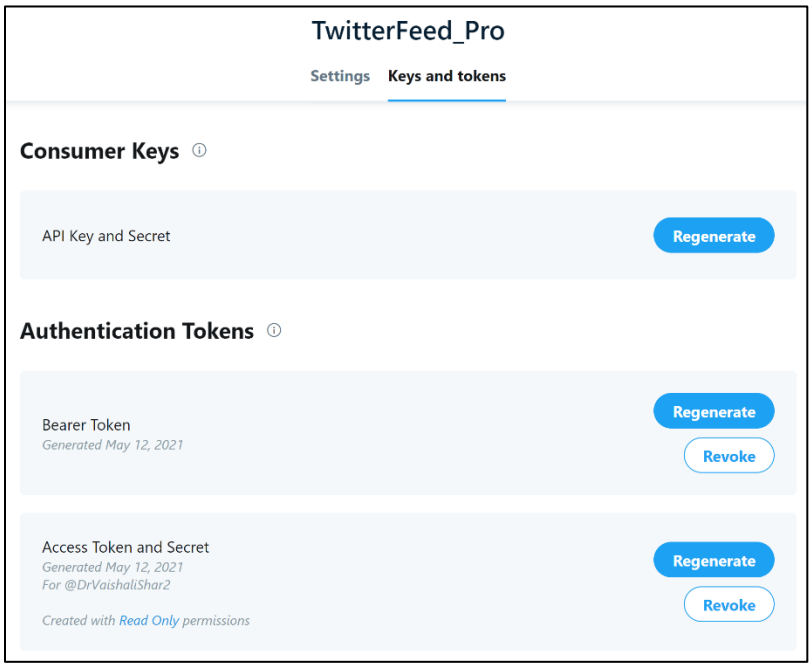


Fig. 2 Generation of Access Tokens

Step 2: Program Design

The streaming process is designed to execute following two files:

receive_tweets.py:

This python file is used to build the Twitter HTTP Client that will receive tweets from TWITTER API using Python and passes them to the Spark Streaming instance. All the libraries are imported first, and then TWITTER KEYS were added in the variables CONSUMER_KEY, CONSUMER_SECRET, ACCESS_TOKEN, ACCESS_SECRET. This will be used OAuth for connecting to Twitter. 'get_tweets' function will call the Twitter API URL and return the response for a stream of tweets. Then a function 'send_tweets_to_spark' is created that taked the response and extract tweets texts and sends every tweet to SparkStreaming instance through a TCP connection. Now localhost configures, get_tweets method is called for getting the tweets from Twitter and pass its response to send_tweets_to_spark for sending the tweets to Spark.

The functions created in this python file are:

1. **def get_tweets()**
2. **def send_tweets_to_spark**(http_resp, tcp_connection):

The full code of this file is given as below:

```
import socket
import sys
import requests
import requests_oauthlib
import json


# Replace the values below with yours
ACCESS_TOKEN = ''
ACCESS_SECRET = ''
CONSUMER_KEY = ''
CONSUMER_SECRET = ''
my_auth = requests_oauthlib.OAuth1(CONSUMER_KEY, CONSUMER_SECRET,ACCESS_TOKEN, AC
CESS_SECRET)

def send_tweets_to_spark(http_resp, tcp_connection):
    for line in http_resp.iter_lines():
        try:
            full_tweet = json.loads(line)
            tweet_text = full_tweet['text']
            print("Tweet Text: " + tweet_text)
            print ("------------------------------------------")
            #tcp_connection.send(tweet_text + '\n')
            tweet_data = bytes(tweet_text + "\n", 'utf-8')
            tcp_connection.send(tweet_data)
        except:
```

```
            e = sys.exc_info()[0]
            print("Error: %s" % e)

def get_tweets():
    url = 'https://stream.twitter.com/1.1/statuses/filter.json'
    query_data = [('language', 'en'), ('locations', '-130,-
20,100,50'),('track','#')]

    query_url = url + '?' + '&'.join([str(t[0]) + '=' + str(t[1]) for t in query_
data])
    response = requests.get(query_url, auth=my_auth, stream=True)
    print(query_url, response)
    return response

TCP_IP = "localhost"
TCP_PORT = 9009
conn = None
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((TCP_IP, TCP_PORT))
s.listen(1)
print("Waiting for TCP connection...")
conn, addr = s.accept()
print("Connected... Starting getting tweets.")
resp = get_tweets()
send_tweets_to_spark(resp,conn)
```

spark_streaming.py

This python file will do real-time processing for the tweets coming in the real-time, extract the hashtags from these tweets, and calculate how many hashtags have been mentioned in the tweets. Firstly, a SparkContext 'sc' is created and then Streaming Context 'ssc' from 'sc' with a batch interval of 2 seconds is created so that transformation on all streams is received in every two seconds. A checkpoint is also defined here to allow periodic RDD checkpointing. Then main DStream dataStream was defined that will connect to the socket server we created before on port 9009 and read the tweets from that port. Each record in the DStream will be a tweet. In the spark_streaming.py file, firstly all tweets are splitted into words and put in words RDD, then only hashtags from all words are filtered and mapped into pair of (hashtag, 1) and put in hashtags RDD. Next step is calculating the count of mentioned hashtags. Function 'reduceByKey' is used that will calculate how many times the hashtags have been mentioned per each batch, it will reset the count in each batch. Here, we need to count across all the batches, so another function 'updateStatebyKey' is used to maintain the state of RDD while updating it with new data. This function takes a function as a parameter called 'update' function named here as 'aggregate_tags_count' that will sum all the 'new_values' for each hashtag and add them to the 'total_sum' that is the sum of all the batches and save the data into 'tags_totals' RDD. Then processing on tags_toals RDD in every batch is done. This is done by using 'process_rdd' function.

The functions created in this python file are:

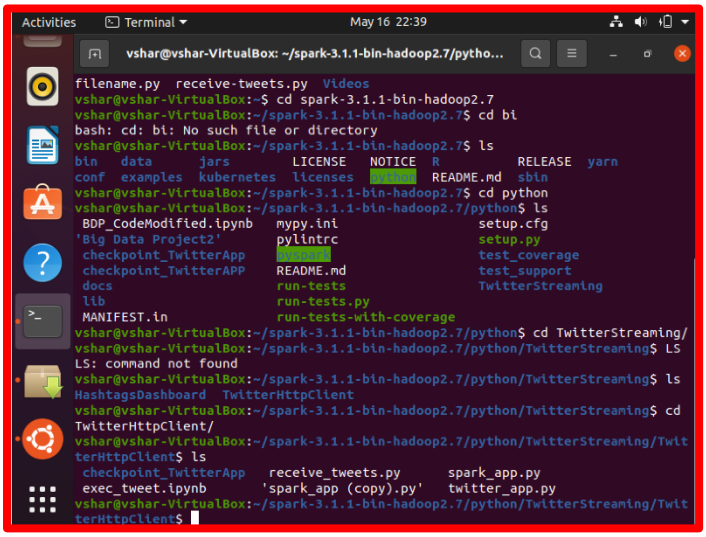
1. **def aggregate_tags_count**(new_values, total_sum)
2. **def get_sql_context_instance**(spark_context)
3. **def get_sql_context_instance**(spark_context)
4. **def process_rdd**

The full code of this file is given as below:

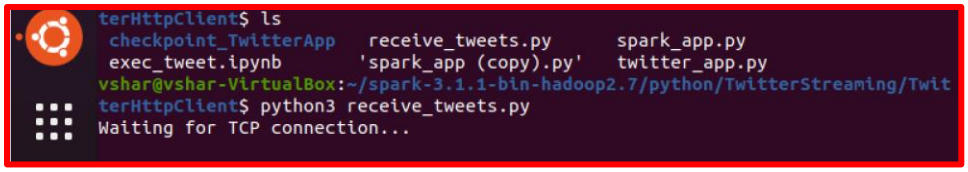
```
import findspark
findspark.init('home/vshar/spark-3.1.1-bin-hadoop2.7')

from pyspark import SparkConf,SparkContext
from pyspark.streaming import StreamingContext
from pyspark.sql import Row,SQLContext
import sys
import requests

# create spark configuration
conf = SparkConf()
conf.setAppName("TwitterStreamApp")
# create spark instance with the above configuration
sc = SparkContext(conf=conf)
sc.setLogLevel("ERROR")
# creat the Streaming Context from the above spark context with window size 2 sec
onds
ssc = StreamingContext(sc, 2)
# setting a checkpoint to allow RDD recovery
ssc.checkpoint("checkpoint_TwitterApp")
# read data from port 9009
dataStream = ssc.socketTextStream("localhost",9009)


def aggregate_tags_count(new_values, total_sum):
    return sum(new_values) + (total_sum or 0)


def get_sql_context_instance(spark_context):
    if ('sqlContextSingletonInstance' not in globals()):
        globals()['sqlContextSingletonInstance'] = SQLContext(spark_context)
    return globals()['sqlContextSingletonInstance']

def process_rdd(time, rdd):
    print("----------- %s -----------" % str(time))
    try:
        # Get spark sql singleton context from the current context
        sql_context = get_sql_context_instance(rdd.context)
        # convert the RDD to Row RDD
```

```
        row_rdd = rdd.map(lambda w: Row(hashtag=w[0].encode("utf-
8"), hashtag_count=w[1]))
        # create a DF from the Row RDD
        hashtags_df = sql_context.createDataFrame(row_rdd)
        # Register the dataframe as table
        hashtags_df.registerTempTable("hashtags")
        # get the top 10 hashtags from the table using SQL and print them
        hashtag_counts_df = sql_context.sql("select hashtag, hashtag_count from h
ashtags order by hashtag_count desc limit 10")
        hashtag_counts_df.show()
    except:
        e = sys.exc_info()[0]
        print("Error: %s" % e)

# split each tweet into words
words = dataStream.flatMap(lambda line: line.split(" "))
# filter the words to get only hashtags, then map each hashtag to be a pair of (h
ashtag,1)
hashtags = words.filter(lambda w: '#' in w).map(lambda x: (x, 1))
# adding the count of each hashtag to its last count
tags_totals = hashtags.updateStateByKey(aggregate_tags_count)

# do processing for each RDD generated in each interval
tags_totals.foreachRDD(process_rdd)

# start the streaming computation
ssc.start()
# wait for the streaming to finish
ssc.awaitTermination()
```

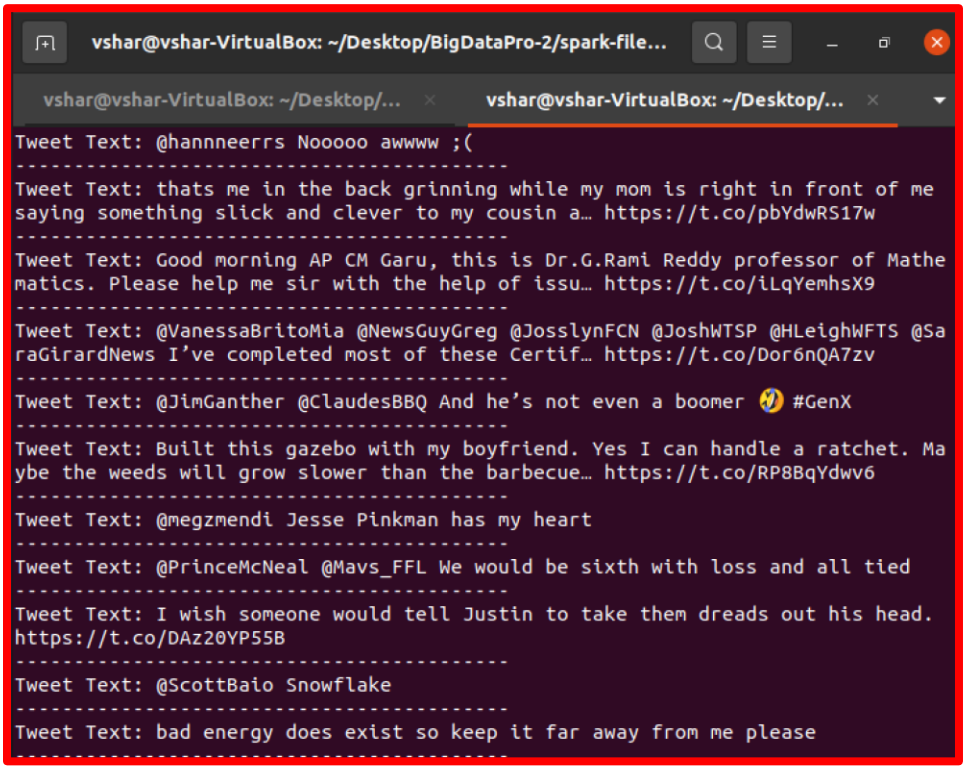
Output is recorded as follows:



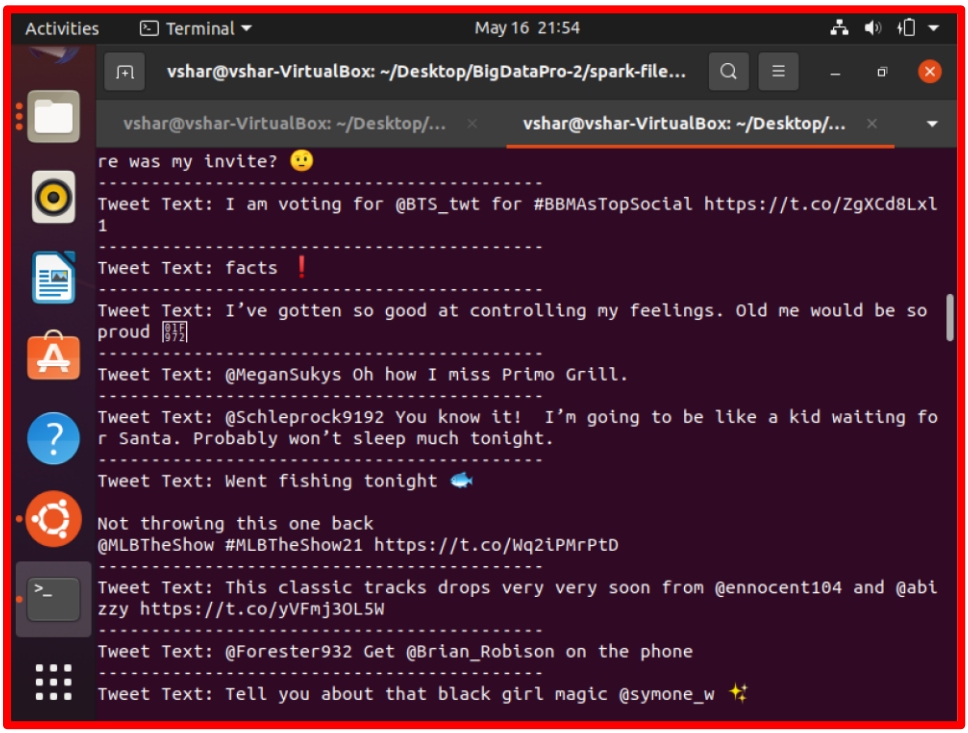
Running receive_tweets.py



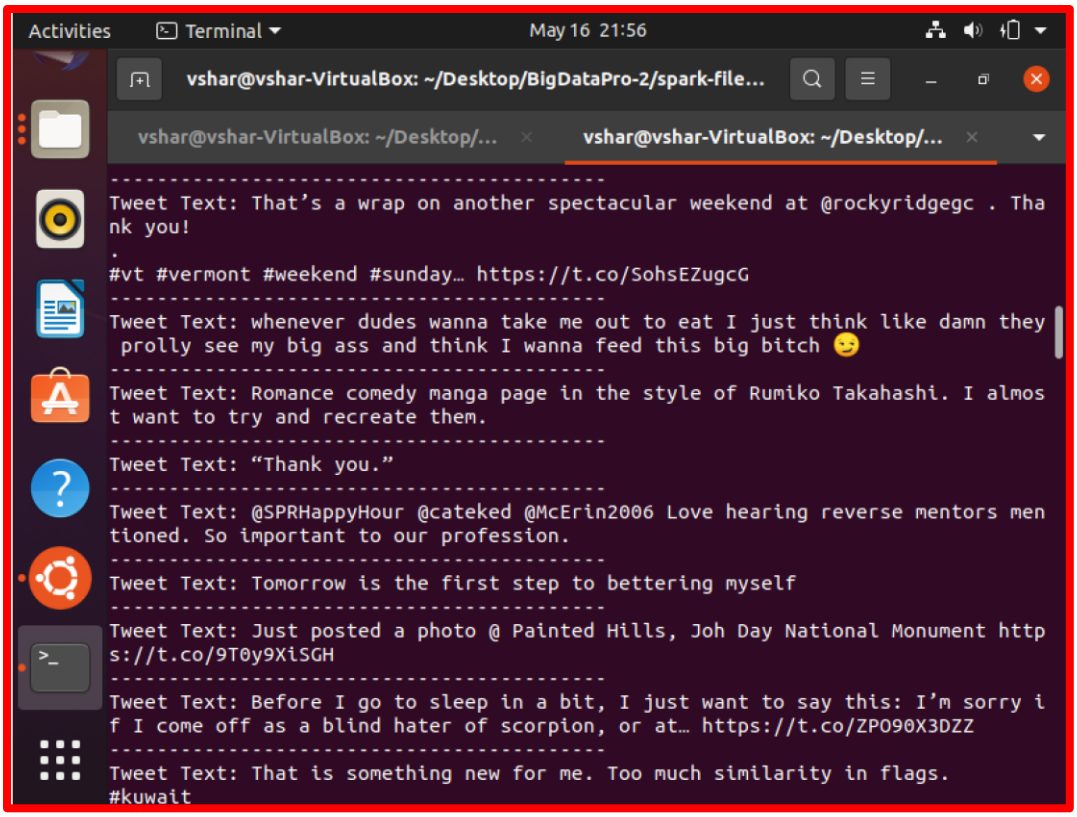
Waiting for TCP Connection



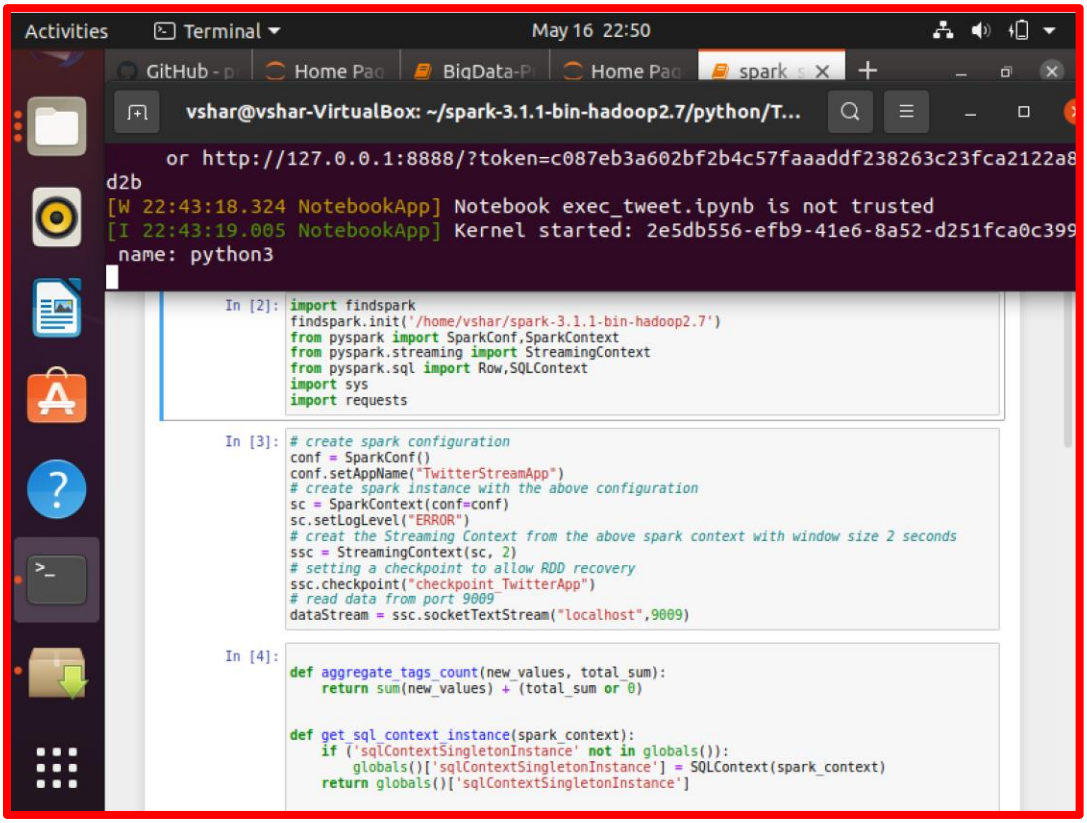
Receiving tweets in real time after running spark_streaming.py



Receiving tweets in real time after running spark_streaming.py



Receiving tweets in real time after running spark_streaming .py



View the top10 hashtags in Jupyter Notebook

LEARNINGS

The project is one of the challenging projects, as it involves data analysis and processing in real time. It was quite interesting but full of challenges. Below are my learnings from this project –

- Learned how to create a Twitter Development account and access the generated keys and tokens which I utilized to create my Twitter HTTP Client application.
- Learned how to access tweets in real time, analyze or process them to derive information about top 10 hashtags in the past few seconds.
- Learned how to integrate pyspark with Jupyter notebook and run two programs in parallel.
- Created spark configuration and created another instance from this configuration.
- Learned to create Streaming Context from the above spark context with window size 2 seconds.
- Learned how to create a TCP connection and read data from port.
- Created DataFrame from the row RDD
- Fetched the top 10 hashtags from the table using SQL
- Splitted each tweet into words
- filtered the words to get only hashtags, then map each hashtag to be a pair of (hashtag,1)
- Added the count of each hashtag to its last count
- Did processing for each RDD generated in each interval
- Learned how to start and stop streaming.

Overall, this project was the wonderful learning experience for me where I created a simple application that reads online streams from Twitter is created, and processes tweets using Spark Streaming to identify hashtags, and then finally returns top 10 hashtags to the user in the real-time or interactive mode

CHALLENGES

One of the greatest challenge was to converting the streaming data into a DataFrame. For this task, several important functions were created and used in the main function. Another challenge was connecting port and local address, after multiple trials, they got connected and helped in fetching the tweets in the real-time mode.

WEB REFERENCES / SOURCES

[1] https://databricks.com/glossary/what-is-spark-streaming#:~:text=Spark%20Streaming%20is%20an%20extension%20of%20the%20core,out%20to%20file%20systems%2C%20databases%2C%20and%20live%20dashboards
[2] https://towardsdatascience.com/hands-on-big-data-streaming-apache-spark-at-scale-fd89c15fa6b0
[3] https://www.toptal.com/apache/apache-spark-streaming-twitter
[4] https://medium.com/@varunabhi86/finding-popular-hashtags-on-twitter-using-spark-streaming-35592c1fab4f
[5] https://github.com/priyaa-t/TwitterStreaming