**EXPERIMENT 11**

**AIM :** Case study to perform Sentiment analysis using Spark Streaming.

**THEORY:**

Data is being generated at an unprecedented rate, and by analyzing it correctly and providing valuable and meaningful insights at the right time, it can result in valuable solutions for an array of domains involved with data. Real-time streaming data is widely used across a range of industries, from health care and banking to media and retail. Netflix, for example, provides real-time recommendations tailored to individual preferences. Similarly to every business that streams large amounts of data and relies on various analytics, Amazon tracks its users’ interactions with its products and makes prompt recommendations of related items.

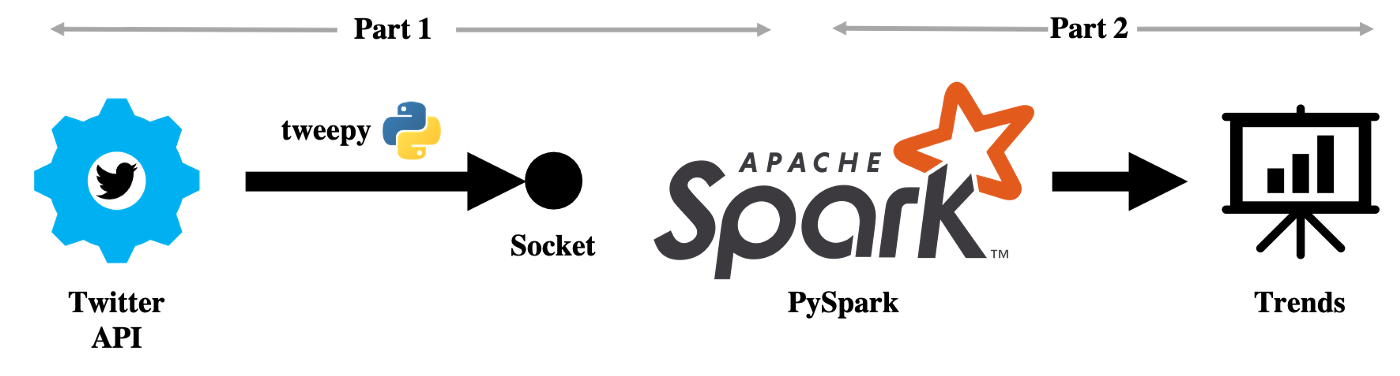
Apache Spark is an efficient framework for analyzing large amounts of data in real-time and performing various analyses on the data. Many resources discuss Spark and its popularity in the big data space, but it is worthwhile to highlight that its core features include real-time big data processing using Resilient Distributed Datasets (RDDs), streaming, and machine learning. In this tutorial, we will demonstrate how Spark Streaming components can be used in conjunction with PySpark to resolve a business problem.

**Motivation**

In today’s society, the importance of social media cannot be denied. Most businesses collect feedback from their Twitter followers in order to gain insight and to gain a better understanding of their customers. Social media feedback changes rapidly, and the ability to analyze that feedback in real time is essential for the success of any business. There are several ways to discover how people react to a new product, brand, or event. For example, the sentiment expressed through tweets about a particular topic, product, brand or event can provide an indication of the level of willingness or trust in a product. Hence, we use this premise in our tutorial and extract trending #tags related to our desired topic every few minutes since hashtagged tweets are more engaging.

We will use Tweepy to access Twitter’s streaming API and the Spark streaming component with TCP socket to receive tweets. We will layer tweets on RDD and then retrieve the most popular hashtags. After that, we use Spark SQL to save the top hashtags to a temporary database. Finally, we visualize the results using Python’s visualization tools.

The following image illustrates the overall architecture of our program.



**CODE:**

**Receive\_tweets:**

from tweepy.auth import OAuthHandler

from tweepy import Stream

from tweepy.streaming import StreamListener

import socket

import json

consumer\_key    = '<consumer\_key>'

consumer\_secret = '<consumer\_secret>'

access\_token    = '<access\_token>'

access\_secret   = '<access\_secret>'

class TweetsListener(StreamListener):

    def \_\_init\_\_(self, csocket):

        self.client\_socket = csocket

    def on\_data(self, data):

        try:

            message = json.loads( data )

            print( message['text'].encode('utf-8') )

            self.client\_socket.send( message['text'].encode('utf-8') )

            return True

        except BaseException as e:

            print("Error on\_data: %s" % str(e))

        return True

    def if\_error(self, status):

        print(status)

        return True

def send\_tweets(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=['football'])

new\_skt = socket.socket()

host = "127.0.0.1"

port = 5555

new\_skt.bind((host, port))

print("Now listening on port: %s" % str(port))

new\_skt.listen(5)

c, addr = new\_skt.accept()

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

send\_tweets(c)

**read\_tweets.py**

import findspark

findspark.init()

from pyspark import SparkContext

from pyspark.streaming import StreamingContext

from pyspark.sql import SQLContext

from pyspark.sql.functions import desc

sc = SparkContext()

ssc = StreamingContext(sc, 10)

sqlContext = SQLContext(sc)

socket\_stream = ssc.socketTextStream("127.0.0.1", 5555)

lines = socket\_stream.window(60)

from collections import namedtuple

fields = ("hashtag", "count" )

Tweet = namedtuple( 'Tweet', fields )

( lines.flatMap( lambda text: text.split( " " ) )

  .filter( lambda word: word.lower().startswith("#") )

  .map( lambda word: ( word.lower(), 1 ) )

  .reduceByKey( lambda a, b: a + b )

  .map( lambda rec: Tweet( rec[0], rec[1] ) )

  .foreachRDD( lambda rdd: rdd.toDF().sort( desc("count") )

  .limit(10).registerTempTable("tweets") ) )

ssc.start()

import time

from IPython import display

import matplotlib.pyplot as plt

import seaborn as sns

get\_ipython().run\_line\_magic('matplotlib', 'inline')

count = 0

while count < 5:

    time.sleep(5)

    top\_10\_tags = sqlContext.sql( 'Select hashtag, count from tweets' )

    top\_10\_df = top\_10\_tags.toPandas()

    display.clear\_output(wait=True)

    plt.figure( figsize = ( 10, 8 ) )

    sns.barplot( x="count", y="hashtag", data=top\_10\_df)

    plt.show()

    count = count + 1

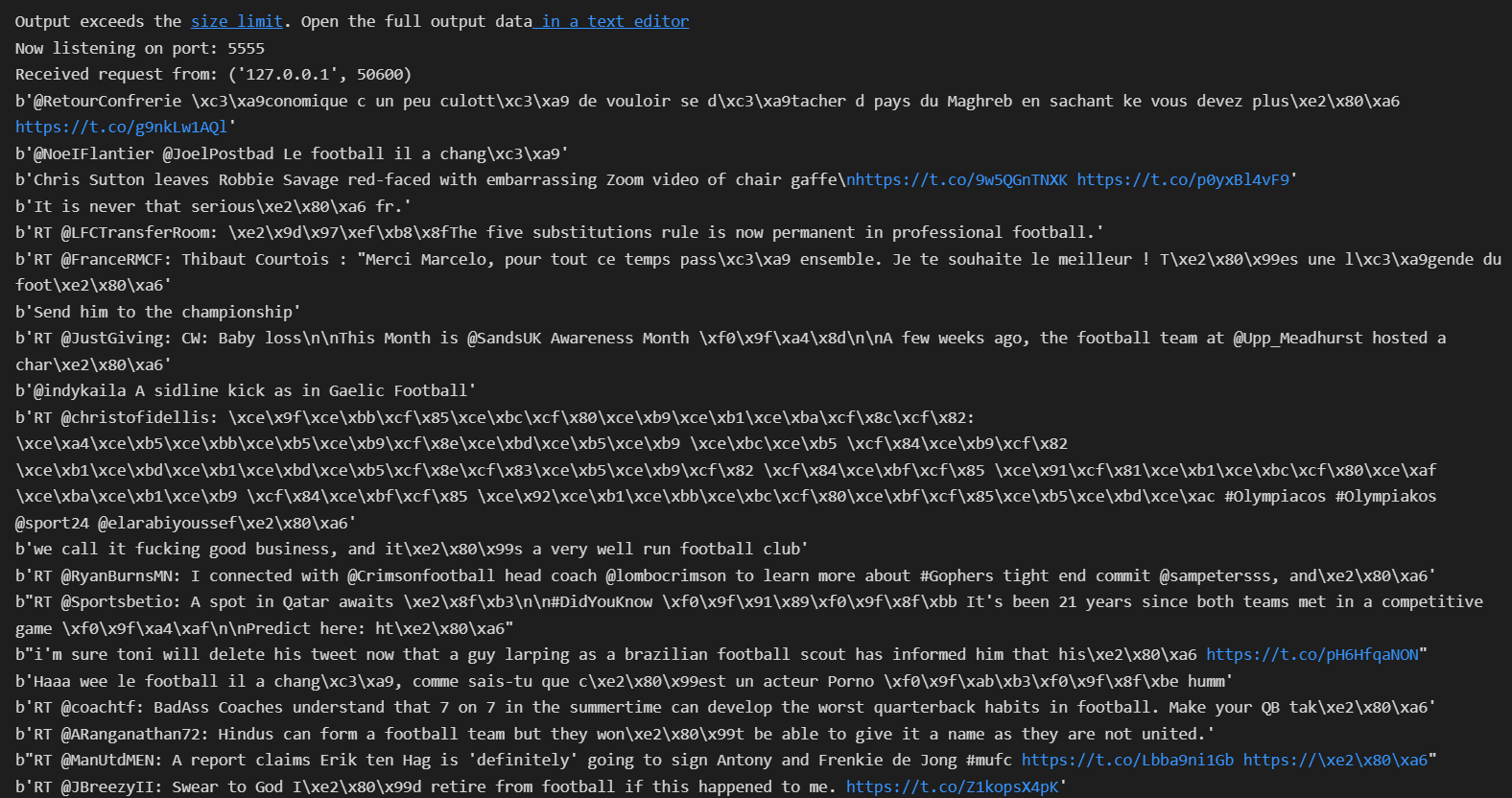
    print(count)

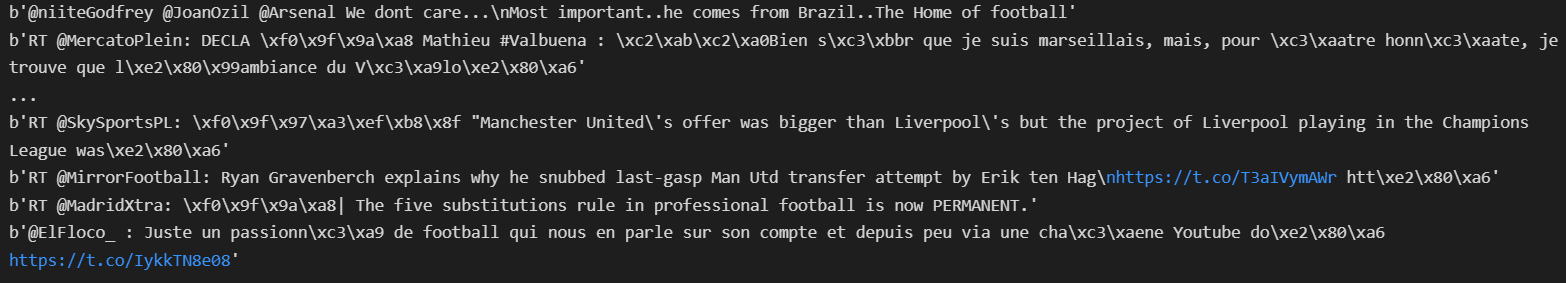
ssc.stop()

Output:

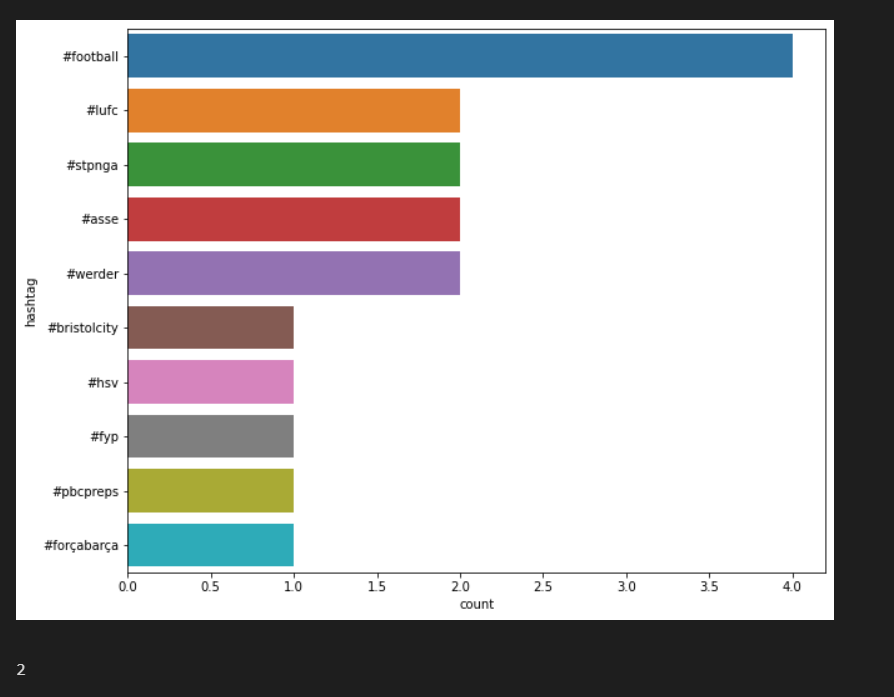
**receive\_tweet.py:**

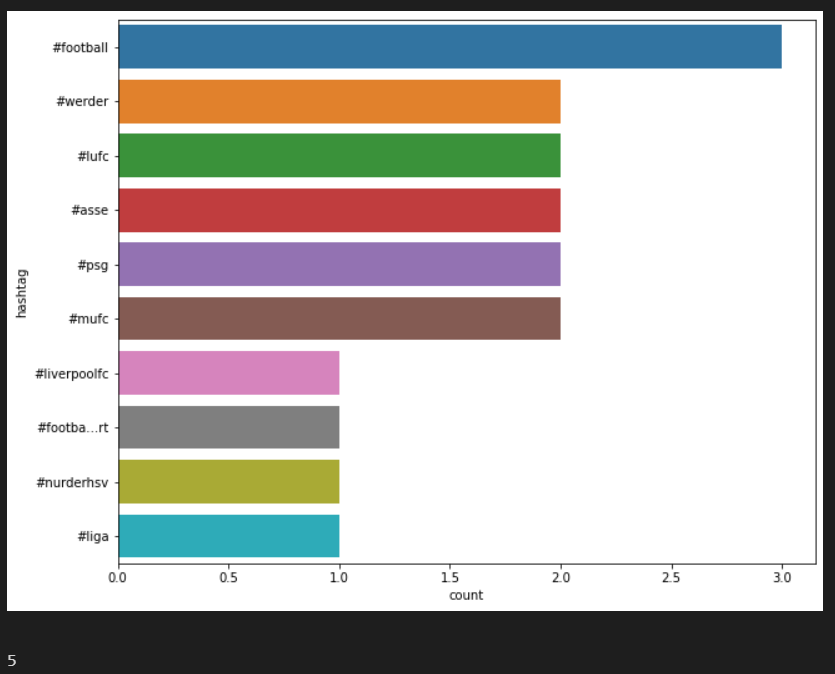
The tweets related to football are obtained as shown below.





The next images contain the 2nd and 5th iteration of the analysis using Spark Streaming respectively; and we can see the hashtags that are being used the most with the word football.





**CONCLUSION :**

In this experiment, I explored and learnt about Spark Streaming for real-time streaming data processing. For the streaming data we used tweetpy library and performed Sentimental Analysis on the same. Initially, #football, #ufc, #stpnga were the top hashtags, later on, #football, #werder , #lufc become the top hashtags -- clearly displaying how we processed the trend with time. The output for the experiment were observed and attached. Thus in this experiment, we implemented Sentimental Analysis on twitter streaming data.