Programa: Ciencia de Datos
Curso: Big Data
Proyecto Final
Streaming en Spark
Alonso Nuñez Sanchez

El presente documento muestra el resultado de la investigación necesaria para llevar a cabo un ejercicio que pone en práctica conocimientos adquiridos en el curso, específicamente para el tratamiento de datos en formato streaming, en el ámbito del Big Data

El ejercicio consiste en obtener datos desde Twitter, filtrarlos, procesarlos y mostrar estadísticas, específicamente un contador de hashtags

Para lograrlo se cuenta con 2 programas que funcionan juntos
Uno (twitter_app) lee tweets con cierto filtro que veremos más adelante y escribe la salida en un socket, una conexión local tcp
El otro (spark_app) consume el contenido del socket, usando una lectura tipo streaming cada 2 segundos y mostrando en pantalla el top 10 de los hashtags con más apariciones

La demostración del funcionamiento se puede encontrar en: https://youtu.be/YuWHb6QNF3U

API de Twitter
El primer paso consiste en utilizar el API que Twitter pone a disposición de desarrolladores
Para esto, es necesario crear una cuenta de tipo desarollador en https://developer.twitter.com/en.html llenar un formulario explicando con qué fines se usará, y esperar algunas horas/días para que habiliten el usuario y suministren token y key para conectarse y consumir el API

Una vez con los accesos y credenciales, se puede hacer uso de los recursos que Twitter pone a disposición
Para el caso del ejercicio de streaming, mayor documentación se puede encontrar en lo siguientes links:
https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters
https://developer.twitter.com/en/docs/tutorials/consuming-streaming-data
https://developer.twitter.com/en/docs/tweets/filter-realtime/api-reference/post-statuses-filter

Estructura del archivo json recibido

https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json

Es importante conocer la estructura del archivo recibida
Para el ejercicio, se extrae el campo “text” cuando se cumpla la condición de que entities.hastaghs sea distinto de vacío (es decir, si existe hashtag en el tweet, se usa el texto)

```
{
  "created_at" : "Thu Apr 06 15:24:15 +0000 2017" ,
  "id_str" : "850006245121695744" ,
  "text" : "1\/ Today we\u2019re sharing our vision for the future of the Twitter API platform!\nhttps:\/\/t.co\/Xw
  "user" : {
    "id" : 2244994945 ,
    "name" : "Twitter Dev" ,
    "screen_name" : "TwitterDev" ,
    "location" : "Internet" ,
    "url" : "https:\/\/dev.twitter.com\/" ,
    "description" : "Your official source for Twitter Platform news, updates & events. Need technical help? Visit ht
  } ,
  "place" : {
  } ,
  "entities" : {
    "hashtags" : [
    ] ,
    "urls" : [
      {
        "url" : "https:\/\/t.co\/XweGngmxlP" ,
        "unwound" : {
          "url" : "https:\/\/cards.twitter.com\/cards\/18ce53wgo4h\/3xo1c" ,
          "title" : "Building the Future of the Twitter API Platform"
        }
      }
    ] ,
    "user_mentions" : [
    ]
  }
```


Twitter_App

En 4 pasos se conecta al API, se obtienen tweets y se envían al socket que consumirá Spark

1. **Conexión con el API de Twitter**

```
:
1 ACCESS_TOKEN = '92690221-B4imevK0Bo7hhAlA4CwhVlDq4bUvKTJgaIMM5l3IG'
2 ACCESS_SECRET = 'g4gA9UfyuRariMz88D6d8D3UWGzNzYQRp5Rc3zeYrjXas'
3 CONSUMER_KEY = 'anPBf9SWRMF7l2V3SGooxOuEP'
4 CONSUMER_SECRET = 'zkx9LcWcZYkPFPlJl2T8BjfySy5XKqkXwpXsr1qSRCbqpXkcZA'
5 my_auth = requests_oauthlib.OAuth1(CONSUMER_KEY, CONSUMER_SECRET,ACCESS_TOKEN, ACCESS_SECRET)
```

En “my_auth” se guardan las credenciales para conexión

2. Leer Tweets

```
def obtener_tweets():
        url = 'https://stream.twitter.com/1.1/statuses/filter.json'
        query_data = [('language', 'es,en'), ('locations', '-85,8,-30,75')] #Centro y Norteamerica
        #query_data = [('language', 'es,en'), ('locations', '-92.5,7.5,-75,17')] #CentroAmerica
        #query_data = [('language', 'es,en'), ('locations', '-85,8,-83,11')] #Costa Rica
        #query_data = [('language', 'en,es'), ('locations', '-85,8,-30,75'),('track','#')] #Norteamerica
        #query_data = [('language', 'en,es'), ('locations', '-142,-50,-30,75'),('track', '#')] #America

        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
```

En el url del stream, con los parámetros de lenguaje, ubicación y credenciales (obtenidas en el punto anterior) se logra la lectura de los tweets que cumplen las condiciones de los parámetros

3. Filtrar hashtags

```
def enviar_tweets_a_spark(http_resp, tcp_connection):
    for line in http_resp.iter_lines():
        try:
            full_tweet = json.loads(line)
            if full_tweet['entities']['hashtags'] != []: #filtra solo los tweets que contengan hashtags
                tweet_text = full_tweet['text']
                tweet_text_encode = (tweet_text + '\n').encode('utf-8')
                print(tweet_text)
                print ("------------------------------------------")

                tcp_connection.send(tweet_text_encode)
        except:
            e = sys.exc_info()[1]
            print("Error: %s" % e)
```

A partir del conocimiento de la estructura del json recibido, se extrae el campo “text” cuando hasthags es distinto a vacio, se imprime en pantalla y se envía al socket (tcp_connection)

4. Apertura del socket

```
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 = obtener_tweets()
enviar_tweets_a_spark(resp, conn)
```

Se abre un socket tcp en local_host, con el puerto 9009 y se esperan los tweets que envía el API y que filtra la función vista en el punto anterior

Spark_App

La segunda aplicación consume el contenido del socket abierto anteriormente usando las librerías de Spark para manejo de streaming

1. Creación de la configuración Spark y el StreamingContext

```
from pyspark import SparkConf,SparkContext
from pyspark.streaming import StreamingContext
from pyspark.sql import Row,SQLContext
import sys
import requests
import findspark
findspark.init('c:\spark')
# crea una configuración spark
conf = SparkConf()
conf.setAppName("TwitterStreamApp")
# crea un contexto spark con la configuración anterior
sc = SparkContext(conf=conf)
sc.setLogLevel("ERROR")
# crea el Contexto Streaming desde el contexto spark visto arriba con intervalo de 2 segundos
ssc = StreamingContext(sc, 2)
# establece un punto de control para permitir la recuperación de RDD
ssc.checkpoint("checkpoint_TwitterApp")
# lee data del puerto 9009
dataStream = ssc.socketTextStream("localhost",9009)
```

2. Procesar streaming
    a. Leer cada entrada
    b. Separar palabras
    c. Filtrar palabras que comiencen con #, es decir los hashtags
    d. Pasar esta palabra a la función “aggregate_tags_count” para llevar el contador de cada hashtag

```
# divide cada Tweet en palabras
words = dataStream.flatMap(lambda line: line.split(" "))
# filtra las palabras para obtener solo hashtags, luego mapea cada hashtag para que sea un par de (hashtag,1)
hashtags = words.filter(lambda w: '#' in w).map(lambda x: (x, 1))
# agrega la cuenta de cada hashtag a su última cuenta
tags_totals = hashtags.updateStateByKey(aggregate_tags_count)
# procesa cada RDD generado en cada intervalo
tags_totals.foreachRDD(process_rdd)
# comienza la computación de streaming
ssc.start()
# espera que la transmisión termine
ssc.awaitTermination()
```

3. Después de leer el streaming, escribe en un dataframe (una tabla temporal con los campos hashtag y contador) y muestra en pantalla lo resultados del top10

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

process_rdd(time, rdd):
    ——→print("----------- %s -----------" % str(time))
    ——→try:
    ——→# obtén el contexto spark sql singleton desde el contexto actual
                sql_context = get_sql_context_instance(rdd.context)
    ——→# convierte el RDD a Row RDD
                row_rdd = rdd.map(lambda w: Row(hashtag=w[0], contador=w[1]))
                #
    ——→# crea un DF desde el Row RDD
                hashtags_df = sql_context.createDataFrame(row_rdd)
    ——→# Registra el marco de data como tabla
                hashtags_df.registerTempTable("hashtags")
    ——→# obtén los 10 mejores hashtags de la tabla utilizando SQL e imprímelos
                hashtag_counts_df = sql_context.sql("select hashtag, contador from hashtags order by contador desc limit 10"
                #
                hashtag_counts_df.show()
    ——→# llama a este método para preparar los 10 mejores hashtags DF y envíalos

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

Algunos resultados

A continuación, algunos resultados de lecturas obtenidas en diferentes momentos

26 Enero por la tarde

```
----------- 2020-01-26 14:17:48 -----------
+----------------+-------------+
|         hashtag|hashtag_count|
+----------------+-------------+
|#HardWorkPaysOff|            1|
|        #RIPKobe|            1|
|   #ElkhartCheer|            1|
|     #KobeByrant|            1|
|            #psa|            1|
+----------------+-------------+

----------- 2020-01-26 14:19:54 -----------
+--------------+-------------+
|       hashtag|hashtag_count|
+--------------+-------------+
|   #KobeBryant|           62|
|     #RIPMAMBA|           59|
|#RIPKobeBryant|           57|
|           #24|           38|
|         #Kobe|           36|
|          #RIP|           26|
|      #RIPKobe|           25|
|            #8|           23|
|   #BlackMamba|           17|
|   #KobeByrant|           17|
+--------------+-------------+
```

26 Enero por la noche

```
----------- 2020-01-26 18:26:28 -----------
+-------------------+--------+
|            hashtag|contador|
+-------------------+--------+
|#bushfiresaustralia|       1|
|         #CraftBeer|       1|
|        #blackmagic|       1|
|        #gosubmusic|       1|
|    #AstrosCheating|       1|
|       #hittacastro|       1|
|              #Beer|       1|
|             #voodoo|       1|
|         #australia|       1|
|   #tellmethefuture|       1|
+-------------------+--------+
```

```
----------- 2020-01-26 18:27:58 -----------
+-------------------+--------+
|            hashtag|contador|
+-------------------+--------+
|       #RoyalRumble|      16|
|        #KobeBryant|       9|
|           #GRAMMYs|       4|
|          #RIPMamba|       4|
|             #4Line|       3|
|                  #|       2|
|#bushfiresaustralia|       1|
|         #CraftBeer|       1|
|        #blackmagic|       1|
|        #gosubmusic|       1|
+-------------------+--------+
```

```
----------- 2020-01-26 18:28:12 -----------
+-------------------+--------+
|            hashtag|contador|
+-------------------+--------+
|       #RoyalRumble|      25|
|        #KobeBryant|      10|
|          #RIPMamba|       6|
|           #GRAMMYs|       5|
|             #4Line|       3|
|                  #|       2|
|               #WWE|       2|
|#bushfiresaustralia|       1|
|         #CraftBeer|       1|
|        #blackmagic|       1|
+-------------------+--------+
```