Exercise2_streaming TABAAI_AYOUB & BOUTHER_OUMAIMA

January 2, 2022

[1]:
```
# Import all the neccessary packages and
# creating all the spark context and the streaming context

import findspark
findspark.init()
from pyspark.sql import SparkSession, SQLContext, Row
from pyspark.streaming import StreamingContext
# Create a local StreamingContext with two working thread and batch interval of␣
 ↪2 minutes
spark=SparkSession.builder.appName("Twitter-streaming").master("local[8]").
 ↪getOrCreate()
ssc = StreamingContext(spark.sparkContext,120)
# Activate Checkpointing
ssc.checkpoint("checkpoint_TwitterApp")
```

Now, we get our machine name and reserve the port as those used in the first notebook,then we create create a socketTextStream, where we will be expecting a Twitter streaming connection. The socketTextStream is created from a Streaming context instance which allows us to get access to data streaming. Also, we create DStream via a window of 10 minutes and a sliding interval of 2 minutes.

[2]:
```
# Get our machine name and reserved port.
host = "127.0.0.1"
port = 9995
tweets = ssc.socketTextStream(host, port)
long_window = 10 # In minutes
lines = tweets.window(long_window * 60)
```

This function is used as lazy instantiation singleton instance of SQLContext such that SQLContext can be be restarted on driver failures.

[3]:
```
def get_Sql_Context_Instance(sparkContext):
    if ('sqlContextSingletonInstance' not in globals()):
        globals()['sqlContextSingletonInstance'] = SQLContext(sparkContext)
    return globals()['sqlContextSingletonInstance']
```

Now, we have the data we need to the proccessing of the data. This is done by the next two cells. We first split the tweets into words, we map them into key-value pairs, we count each word then

we do the processing by **process** method which creates a dataframe from the rdd already created and save in our local machine as a **csv** file to be used after.

[4]:
```
def process(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
        rows = rdd.map(lambda w: Row(word=w[0], word_count=w[1]))

        # create a DF from the Row RDD
        hashtags_df = sql_context.createDataFrame(rows)

        # 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 word , word_count from␣
 ↪hashtags where word like '#%'order by word_count desc limit 10")

        hashtag_counts_df.show()
        hashtag_counts_df.coalesce(1).write.format('com.databricks.spark.csv').
 ↪mode('overwrite').option("header", "true").csv("hashtag.csv")
    except Exception as e:
        print(e)
        pass
```

[5]:
```
# split each tweet into words
words = lines.flatMap(lambda line: line.split(" ")).map(lambda word:word.
 ↪lower())

# map each word to be a pair of (word,1)
hashtags = words.map(lambda x: (x, 1))

# count of each word
tags_totals = hashtags.reduceByKey(lambda a,b: a+b)

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

[6]:
```
# Start the processing ...
ssc.start()
```

```
----------- 2022-01-02 12:34:00 -----------
+---------------+----------+
|           word|word_count|
```

```
+--------------+----------+
|       #newyear|        43|
|      #socialfi|        24|
|        #web3.0|        20|
|         #covid|         7|
|    #   |         5|
|#mytopfollowers|         5|
|      #covid-19|         4|
|      #airdrops|         4|
|          #web3|         4|
|              #|         2|
+--------------+----------+

----------- 2022-01-02 12:36:00 -----------
+--------------+----------+
|           word|word_count|
+--------------+----------+
|       #newyear|       100|
|      #socialfi|        67|
|        #web3.0|        53|
|         #covid|        19|
|      #airdrops|        14|
|          #web3|        14|
|      #covid-19|         8|
|#mytopfollowers|         7|
|       #covid19|         6|
|   #newyear2022|         6|
+--------------+----------+

----------- 2022-01-02 12:38:00 -----------
+-------------------------+----------+
|                     word|word_count|
+-------------------------+----------+
|                 #newyear|       231|
|                #socialfi|       149|
|                  #web3.0|       116|
|                   #covid|        36|
|                #airdrops|        33|
|                    #web3|        33|
|                 #covid19|        20|
|#       |        12|
|                      #uk|        12|
|          #nomasksinclass|        11|
+-------------------------+----------+

----------- 2022-01-02 12:40:00 -----------
+--------------+----------+
|           word|word_count|
```

```
+--------------+----------+
|       #newyear|       601|
|      #socialfi|       376|
|        #web3.0|       300|
|         #covid|        90|
|      #airdrops|        76|
|          #web3|        76|
|       #covid19|        48|
|   #newyear2022|        33|
|#nomasksinclass|        32|
|            #uk|        32|
+--------------+----------+

----------- 2022-01-02 12:42:00 -----------
+------------------------+----------+
|                    word|word_count|
+------------------------+----------+
|                 #newyear|       786|
|                #socialfi|       502|
|                  #web3.0|       395|
|                   #covid|       113|
|                    #web3|       108|
|                #airdrops|       107|
|                 #covid19|        66|
|             #newyear2022|        43|
|          #nomasksinclass|        40|
|#       |        38|
+------------------------+----------+

----------- 2022-01-02 12:44:00 -----------
+------------------------+----------+
|                    word|word_count|
+------------------------+----------+
|                 #newyear|       921|
|                #socialfi|       605|
|                  #web3.0|       474|
|                   #covid|       134|
|                    #web3|       132|
|                #airdrops|       131|
|                 #covid19|        77|
|#       |        51|
|          #nomasksinclass|        51|
|             #newyear2022|        51|
+------------------------+----------+

----------- 2022-01-02 12:46:00 -----------
+------------------------+----------+
|                    word|word_count|
```

```
+-------------------------+----------+
|                 #newyear|      1060|
|                #socialfi|       684|
|                  #web3.0|       534|
|                   #covid|       154|
|                    #web3|       152|
|                #airdrops|       150|
|                 #covid19|        90|
|          #nomasksinclass|        58|
|#       |        57|
|             #newyear2022|        56|
+-------------------------+----------+

----------- 2022-01-02 12:48:00 -----------
```

[8]:
```
ssc.stop()
```

[10]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

path_hashtag = r'hashtag.csv/
 ↪part-00000-c1893e80-1a96-410e-9e18-b3393db65c0d-c000.csv'
df = pd.read_csv(path_hashtag)
df.describe()
```

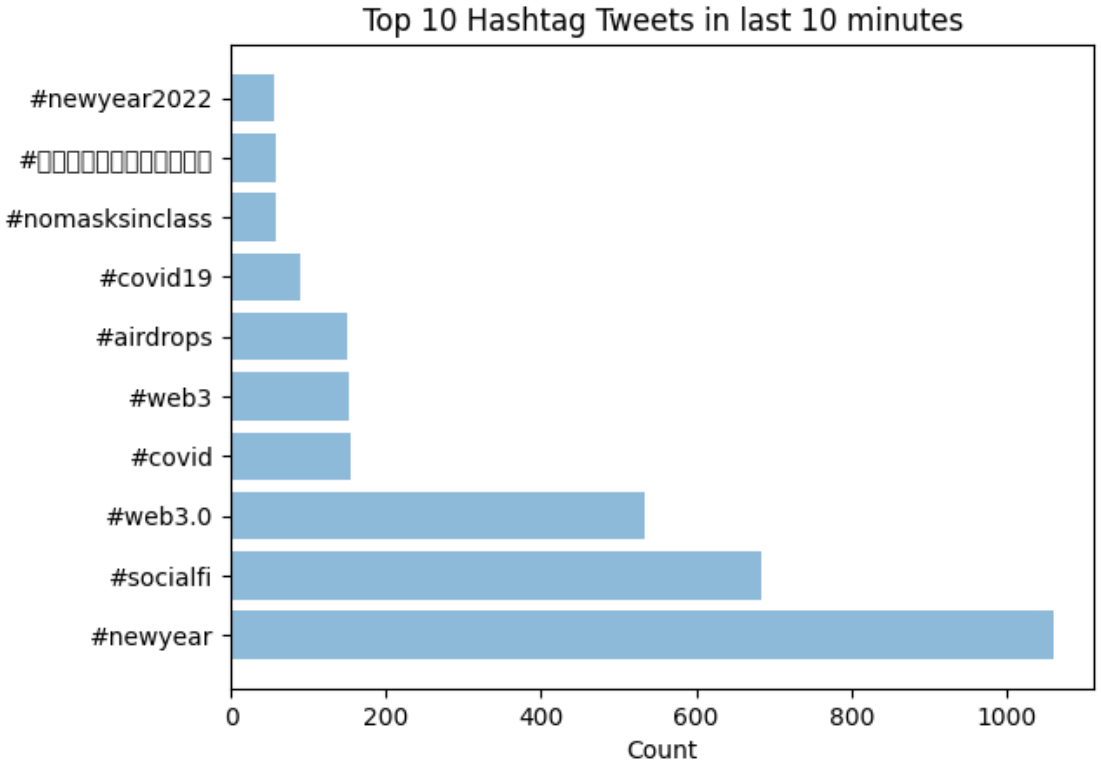
[10]:
```
        word_count
count    10.000000
mean    299.500000
std     344.283233
min      56.000000
25%      66.000000
50%     151.000000
75%     439.000000
max    1060.000000
```

[14]:
```
import matplotlib.pyplot as plt
plt.rcdefaults()
import numpy as np
import matplotlib.pyplot as plt

objects = df.word
y_pos = np.arange(len(objects))
count = df.word_count

plt.barh(y_pos, count, align='center', alpha=0.5)
plt.yticks(y_pos, objects)
plt.xlabel('Count')
```

```
plt.title('Top 10 Hashtag Tweets in last 10 minutes')
plt.show()
```



[ ]: