Chapter 10: Introduction to Spark Streaming

Note on Streaming

Streaming is something that is rapidly advancing and changin fast, there are multipl enew libraries every year, new and different services always popping up, and what is in this notebook may or may not apply to you. Maybe your looking for something specific on Kafka, or maybe you are looking for streaming about twitter, in which case Spark might be overkill for what you really want. Realistically speaking each situation is going to require a customized solution and this course is never going to be able to supply a one size fits all solution. Because of this, I wanted to point out some great resources for Python and Spark StreamingL

- The Official Documentation is great. This should be your first go to. (http://spark.apache.org/docs/latest/streaming-programming-guide.html#spark-streaming-programming-guide)
- Fantastic Guide to Spark Streaming with Kafka (https://www.rittmanmead.com/blog/2017/01/getting-started-with-spark-streaming-with-python-and-kafka/)
- Another Spark Streaming Example with Geo Plotting (http://nbviewer.jupyter.org/github/ibm-cds-labs/spark.samples/blob/master/notebook/DashDB%20Twitter%20Car%202015%20Python%20

Let's discuss SparkStreaming!

Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Data can be ingested from many sources like Kafka, Flume, Kinesis, or TCP sockets, and can be processed using complex algorithms expressed with high-level functions like map, reduce, join and window. Finally, processed data can be pushed out to filesystems, databases, and live dashboards. In fact, you can apply Spark's machine learning and graph processing algorithms on data streams.

Keep in mind that a few of these Streamiing Capabilities are limited when it comes to Python, you'll need to reference the documentation for the most up to date information. Also the streaming contexts tend to follow more along with the older RDD syntax, so a few things might seem different than what we are used to seeing, keep that in mind, you'll definitely want to have a good understanding of lambda expressions before continuing with this!

There are SparkSQL modules for streaming:

http://spark.apache.org/docs/latest/api/python/pyspark.sql.html?highlight=streaming#module-pyspark.sql.streaming (http://spark.apache.org/docs/latest/api/python/pyspark.sql.html?highlight=streaming#module-pyspark.sql.streaming)

But they are all still listed as experimental, so instead of showing you somethign that might break in the future, we'll stick to the RDD methods (which is what the documentation also currently shows for streaming).

Internally, it works as follows. Spark Streaming receives live input data streams and divides the data into batches, which are then processed by the Spark engine to generate the final stream of results in batches.

Twitter Example

In order to use all of this though, we need to setup a Developer API acocunt with Twitter and create an application to get credentials. Review the video for instructions on how to do this or if you are already familiar with it, just get the credentials from:

```
https://apps.twitter.com/
```

Once you have that you also need to install python-twitter, a python library to connect your Python to the twitter dev account.

You probably won't be able to run this example and then previous in the same notebook, you need to restart you kernel.

Let's get started!

Begin by running the TweetRead.py file. Make sure to add your own IP Adress and your credential keys.

In [40]:
```
import findspark
findspark.init()
```

In [41]:
```
import pyspark
```

In [42]:
```
# May cause deprecation warnings, safe to ignore, they aren't errors
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.sql import SQLContext
from pyspark.sql.functions import desc
```

In [43]:
```
# Can only run this once. restart your kernel for any errors.
sc = SparkContext()
```

In [44]:
```
ssc = StreamingContext(sc, 10 )
sqlContext = SQLContext(sc)
```

In [45]:
```
socket_stream = ssc.socketTextStream("127.0.0.1", 5555)
```

In [46]:
```
lines = socket_stream.window( 20 )
```

In [47]:
```
from collections import namedtuple
fields = ("tag", "count" )
Tweet = namedtuple( 'Tweet', fields )
```

In [48]:
```
# Use Parenthesis for multiple lines or use \.
( lines.flatMap( lambda text: text.split( " " ) ) #Splits to a list
  .filter( lambda word: word.lower().startswith("#") ) # Checks for hashtag calls
  .map( lambda word: ( word.lower(), 1 ) ) # Lower cases the word
  .reduceByKey( lambda a, b: a + b ) # Reduces
  .map( lambda rec: Tweet( rec[0], rec[1] ) ) # Stores in a Tweet Object
  .foreachRDD( lambda rdd: rdd.toDF().sort( desc("count") ) # Sorts Them in a DF
  .limit(10).registerTempTable("tweets") ) ) # Registers to a table.
```

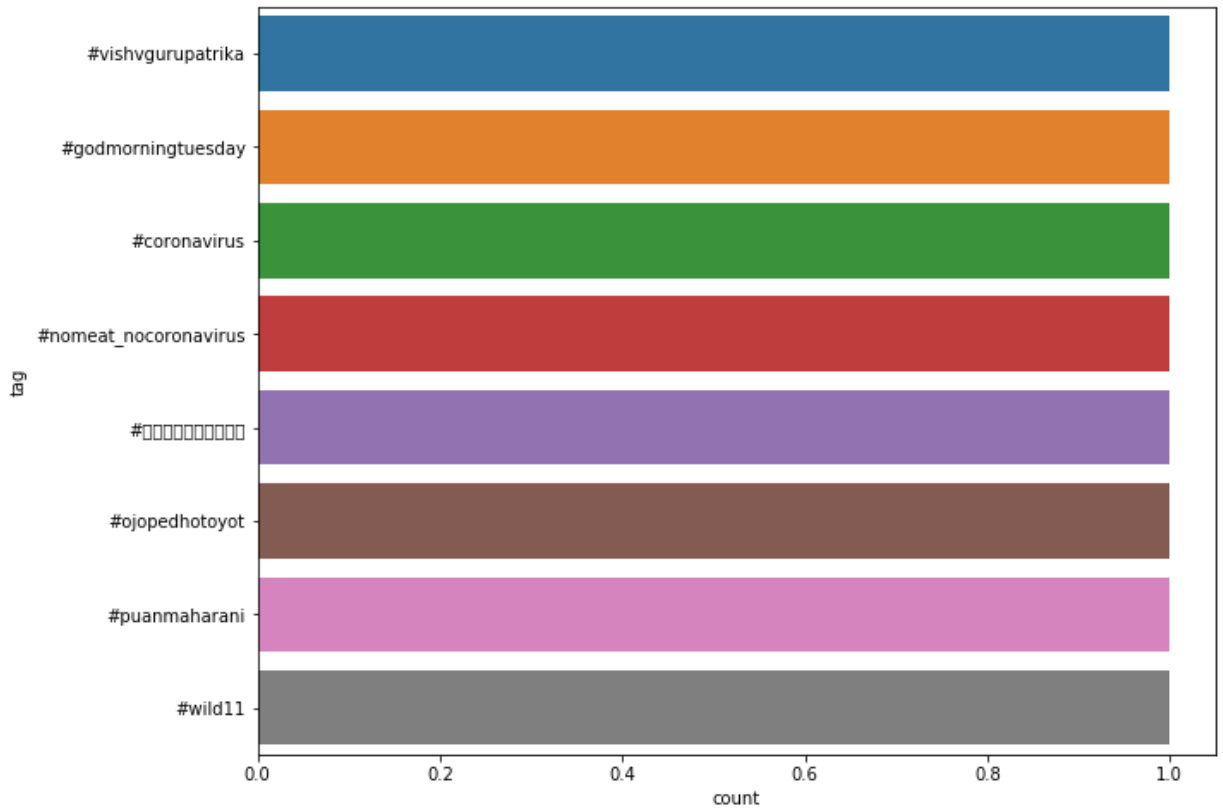
Now run TweetRead_new.py

- ..\Spark_Streaming>python TweetRead_new.py >> tweets_vCov.txt

In [50]:
```
ssc.start()
```

In [51]:
```
import time
from IPython import display
import matplotlib.pyplot as plt
import seaborn as sns
# Only works for Jupyter Notebooks!
%matplotlib inline
```

In [55]:
```
try:
    count = 0
    while count < 10:
        time.sleep( 3 )
        top_10_tweets = sqlContext.sql( 'Select tag, count from tweets' )
        top_10_df = top_10_tweets.toPandas()
        display.clear_output(wait=True)
        plt.figure( figsize = ( 10, 8 ) )
        sns.barplot( x="count", y="tag", data=top_10_df)
        plt.show()
        count = count + 1
except:
    print("No tweets now")
```



In [56]:
```
ssc.stop()
```