Twitter Tweets Stream with PySpark

To stream tweets with twint and to process them with spark stream and to show real time top 5 hashtags for the given search word

Student:

Kunal PATIL

Teacher:

Mark ANGOUSTURES

Course:

Spark and Python for Big Data

Table of Contents

Introduction

This practical aim to analyze tweets from twitter and to get top hashtags for any searched word within the script from user.

To get tweet information, Twitter official API or Twint can be used. This practical based on getting twitter data from twint,

What is Twint?

Twint is an advanced Twitter scraping tool written in Python that allows for scraping Tweets from Twitter profiles without using Twitter's API.

Twint utilizes Twitter's search operators to let you scrape Tweets from specific users, scrape Tweets relating to certain topics, hashtags & trends, or sort out sensitive information from Tweets like e-mail and phone numbers. I find this very useful, and you can get really creative with it too.

Twint also makes special queries to Twitter allowing you to also scrape a Twitter user's followers, Tweets a user has liked, and who they follow without any authentication, API, Selenium, or browser emulation.

(Source: https://github.com/twintproject/twint)

The execution is done with the help of docker and execute spark application within the container.

Scripts

Creation of Docker File

```
FROM python:3.6-slim

MAINTAINER Kunal PATIL

# Update and Install git
RUN apt-get update && apt-get install git -y

# Install Twint package from official git version
RUN pip3 install --user --upgrade git+https://github.com/twintproject/twint.git@origin/master#egg=twint

# Install PySpark
RUN pip3 install --user pyspark
```

Creation of spark script

```
import twint
from pyspark import SparkContext
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count
from pyspark.sql.types import *

c = twint.Config()
c.Search = "epita"
c.Hide_output = True
c.Pandas = True
c.Limit = 10

tweets = twint.run.Search(c)

columns = twint.storage.panda.Tweets_df.columns

df = twint.storage.panda.Tweets_df[['hashtags', 'tweet']]

sc = SparkContext.getOrCreate()
sc.setLogLevel('WARN')
spark = SparkSession(sc)
htags = sc.parallelize(df['hashtags'])
htags = htags.reduce(lambda x, y: x+y)
twitter_hashtags = spark.createDataFrame(htags, StringType()).createTempView('data')
spark.sql('SELECT value, count(value) As tag_count from data group by value order by tag_count DESC LIMIT 5').show()
sc.stop()
```

Exploring script in detail

Importing necessary libraries

```
import twint
from pyspark import SparkContext
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count
from pyspark.sql.types import *
```

Launch twint with a search word

```
c = twint.Config()
c.Search = "<INPUT WORD>"
c.Hide_output = True
c.Pandas = True
c.Limit = 10

tweets = twint.run.Search(c)
```

(Source: https://github.com/twintproject/twint)

Storing output from twint to dataFrame

```
columns = twint.storage.panda.Tweets_df.columns

df = twint.storage.panda.Tweets_df[['hashtags', 'tweet']]
```

(Source: https://github.com/twintproject/twint/wiki/Pandas-integration)

Process on DataFrame with SparkSession

```
sc = SparkContext.getOrCreate()
sc.setLogLevel('WARN')
spark = SparkSession(sc)
htags = sc.parallelize(df['hashtags'])
htags = htags.reduce(lambda x, y: x+y)
twitter_hashtags = spark.createDataFrame(htags, StringType()).createTempView('data')
spark.sql('SELECT value, count(value) As tag_count from data group by value order by tag_count DESC LIMIT 5').show()
sc.stop()
```

(Source: https://spark.apache.org/docs/latest/api/python/pyspark.sql.html?highlight=sparksession)

Execution

Launch DockerFile

```
kunal@LAPTOP-NJ0MODMM MINGW64 /d/Subjects/PySpark_Final
$ docker build -t hashtags .
Sending build context to Docker daemon  14.85kB
Step 1/6 : FROM python:3.6-slim
3.6-slim: Pulling from library/python
45b42c59be33: Pull complete
f875e16ab19c: Pull complete
d96f1da8de03: Pull complete
7e1476b9f134: Pull complete
278fa8bc613b: Pull complete
Digest: sha256:f8e3a8aa99c425d502c277b8574b107a1d89788299a84e5601cc6e4c5b0b565c
Status: Downloaded newer image for python:3.6-slim
 ---> e89065223480
Step 2/6 : RUN apt-get update && apt-get install git -y
 ---> Running in 2a301bf2f00d
Get:1 http://security.debian.org/debian-security buster/updates InRelease [65.4 kB]
Get:2 http://deb.debian.org/debian buster InRelease [122 kB]
Get:3 http://deb.debian.org/debian buster-updates InRelease [51.9 kB]
Get:4 http://security.debian.org/debian-security buster/updates/main amd64 Packages [267 kB]
Get:5 http://deb.debian.org/debian buster/main amd64 Packages [7907 kB]
Get:6 http://deb.debian.org/debian buster-updates/main amd64 Packages [9504 B]
Fetched 8422 kB in 3s (2908 kB/s)
Reading package lists...
Reading package lists...
Building dependency tree...
Reading state information...
```

Create a container from the docker Image Created

```
kunal@LAPTOP-NJ0MODMM MINGW64 /d/Subjects/PySpark_Final
$ docker run -it -d --name twiiterapp hashtags
35a27e4250626d5d2966a60990b77f5dcf5df8b1b85888735485c1df7132e273
```

Run Shell for container

```
kunal@LAPTOP-NJ0MODMM MINGW64 /d/Subjects/PySpark_Final
$ docker exec -it twiiterapp /bin/bash
root@35a27e425062:/# vim twint_engine.py
root@35a27e425062:/# vim twint_engine.py
root@35a27e425062:/# python twint_engine.py
```

Execute the script

1. Input word: EPITA

```
root@35a27e425062:/# python twint_engine.py
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/root/.local/lib/python3.6/site-packages/py
spark/jars/spark-unsafe_2.12-3.0.1.jar) to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
21/02/17 13:29:30 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
+------------+---------+
|       value|tag_count|
+------------+---------+
|cybersécurité|        8|
|     actuweb|        3|
|         srs|        3|
|  innovations|        3|
|      alumni|        2|
+------------+---------+
```

Below hashtags are popular with epita

```
+------------+---------+
|       value|tag_count|
+------------+---------+
|cybersécurité|        8|
|     actuweb|        3|
|         srs|        3|
|  innovations|        3|
|      alumni|        2|
+------------+---------+
```

2. Input Word: Dassault Systemes

```
+---------------+---------+
|          value|tag_count|
+---------------+---------+
|dassaultsystemes|       10|
|        podcast|        7|
|dassaultsystèmes|        5|
|     technology|        4|
|   3dexperience|        3|
+---------------+---------+
```