WORKING WITH STREAMING TWITTER DATA USING KAFKA

Here we show how to read messages streaming from Twitter and store them in Kafka. In Part 2 we will show how to retrieve those messages from Kafka and read them into Spark Streaming.

Overview

People use **Twitter** data for all kinds of business purposes, like monitoring brand awareness. Twitter, unlike Facebook, provides this data freely. So you can use that and store it in a big data database so that you can run analytics over it. You could, for example, make a graph of currently trending topics.

Since this data coming is as a stream, it makes sense to process it with a streaming product, like **Apache Spark Streaming**. That keeps data in memory without writing it to storage, unless you want to. But streaming data has value when it is live, i.e., streaming. So there would not be much reason to store that data permanently to some place like Hadoop.

Kafka is the tool most people use to read streaming data like this. It follows a publish-subscribe model where you write messages (publish) and read them (subscribe). Messages are grouped into **topics**. As messages are consumed, they are removed from Kafka.

Now, here is our example.

Prerequisites

- Python
- Kafka
- Twitter API credentials

Steps

1. Create an App on the Twitter API website. Basically that will give you keys that you need to use the Twitter API.
2. Then install Kafka. It's as simple as downloading and unzipping it.
3. Install Install kafka-python and twitter-python:

```
pip install kafka-python
pip install python-twitter
pip install tweepy
```

4. Start Zooper and Kafka from the Kafka install directory:

```
bin/zookeeper-server-start.sh config/zookeeper.properties
bin/kafka-server-start.sh config/server.properties
```

5. Create a topic. We will create the topic "trump" as obviously there are a lot of Tweets about the President.

```
 bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-
factor 1 --partitions 1 --topic trump
```

6. Fill in the access keys you got from your Twitter API account and add them to this code below.

```
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
from kafka import SimpleProducer, KafkaClient

access_token = "(get your own)"
access_token_secret =  "(get your own)"
consumer_key =  "(get your own)"
consumer_secret =  "(get your own)"

class StdOutListener(StreamListener):
    def on_data(self, data):
        producer.send_messages("trump", data.encode('utf-8'))
        print (data)
        return True
    def on_error(self, status):
        print (status)

kafka = KafkaClient("localhost:9092")
```

```
producer = SimpleProducer(kafka)
l = StdOutListener()
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
stream = Stream(auth, l)
stream.filter(track="trump")
```

7. Now run it. Not only will it write messages to the screen, it will publish them to Kafka. The messages are in JSON format and will look something like:

```
l,"in_reply_to_status_id_str":null,"in_reply_to_user_id"
:null,"in_reply_to_user_id_str":null,"in_reply_to_screen
_name":null,"user":{"id":32871086,"id_str":"32871086","name":"Kyle
Griffin","screen_name":"kylegriffin1","location":
"Manhattan,
NY","url":"http:\/\/www.msnbc.com\/the-last-word","description":"Producer
. MSNBC's @TheLastWord. Honorary Aussie. Opinions mine. Please
clap.","protected":false,"verified":true,"followers
_count":148893,"friends_count":1212,"listed_count":
2646,"favourites_count":38473,"statuses_count":30483,
"created_at":"Sat Apr 18 12:45:48 +0000
2009","utc_offset":-14400,"time_zone":"Eastern Time (US &
Canada)","geo_enabled":true,"lang":"en",
"contributors_enabled":false,"is_translator":false,
"profile_background_color":"C0DEED","profile_background
_image_url":"http:\/\/pbs.twimg.com\
```

8. You can test that topics are getting published in Kafka by using:

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic
trump --from-beginning
```

9. It should echo the same output.

In a subsequent post we will show how to retrieve these messages from Kafka and read them into Apache Spark.