Streaming Trending Artists on Twitter

ID2221 Data-Intensive Computing Project - KTH Royal Institute of Technology
Authors: Fadhil Mochammad (fadhilm@kth.se), M. Irfan Handarbeni (handa@kth.se)

Twitter, as a social media platform has attracted millions of users to express their thoughts about a wide range of topics. Therefore, huge numbers of tweets are generated resulting in a stream of text data. For this project, we are specifically interested in finding tweets that share Spotify track. We process those tweets and get some information about the track using Spotify API. At the end, our goal is to use that acquired data with a specified time frame to find the trending artists on that particular time frame. We also want to make our project scalable, high-throughput, and fault-tolerance. To achieve this, we use cluster-computing frameworks which are: Kafka, Spark Streaming, and Cassandra.

Here you can see the recorded result: https://youtu.be/eLnKT_aGahk

Dataset

For this project, we are using Twitter data that we obtain from the Twitter Stream API [1]. Since we use Python for our implementation, we use the Tweepy [2] python library to handle the integration of Twitter Stream API with Python. We also need extra information about each track and we get this information by using the Spotipy [3] python library to handle the Spotify Web API [4].

Approach

This project has three main parts. All parts are implemented in Pyspark and Python.

1. **Input Twitter Stream and Kafka**
   In this part, input stream data from Twitter's Streaming API are fed into the Kafka producer. There are two-level filtering in this part. The first filter is calling the Twitter Streaming API using "spotify com" keyword. That filter was used so that the acquired tweets contain a URL to a song in Spotify (open.spotify.com). However, due to limitation in the filter function in Twitter API, we need to add another layer of filtering. Hence, we add a condition so that only JSON objects containing URL are processed. Finally, since we are only interested in the creation time and URL of the tweets, we extracted the value of "created_at" and "expanded_url" and feed it as a JSON object to the Kafka Producer with particular topic.

   Example of processed tweet:
   ```
   {"created_at": "Sun Oc 27 18:50:00 +0000 2019", "expanded_url": "https://open.spotify.com/track/6BcF4tXjinsT9NBYFoxS6t?si=QgyveJpgTbOl3Fbi8MU-JQ"}
   ```

2. **Consume from Kafka, Get Artist Information from Spotify and Storing to Cassandra**
   This part is implemented in Pyspark. A keyspace and table are created in Cassandra (if both don't exist yet), then the spark context and spark streaming context are defined. A Kafka-Spark integration is required so we can consume the data from the broker. We use receiver-less approach for this which directly queries Kafka for the latest offset of the topic. After receiving data from Kafka, we extract the track_id from the "expanded_url" value of each record. This track_id is required for acquiring the name of the artist. We send this track_id as a parameter for Spotify API call. After getting the `artist`, we need to filter the RDD containing the record by ditching record with `None` artist value. This filter is to avoid error when inserting the data into Cassandra. Then, we construct the data for insertion to Cassandra so the RDDs contain records with `artist`, `created_at`, and `count` values. `artist` and `created_at` are from the data we previously acquired and we set the `count` to 1. Finally we write each RDD to Cassandra with saveToCassandra function and `keyspace`, `table` as parameters.

artist	created_at	count
Sheryl Crow	2019-10-27 16:15:27.000000+0000	1
Sheryl Crow	2019-10-27 16:41:40.000000+0000	1
J. Holiday	2019-10-27 15:22:38.000000+0000	1
Chicosci	2019-10-27 14:50:06.000000+0000	1
Jeff Beck	2019-10-27 16:37:09.000000+0000	1
Princ£ Aurelius	2019-10-27 16:12:20.000000+0000	1
The Mad Lads	2019-10-27 13:44:25.000000+0000	1
Rocket Rockers	2019-10-27 15:57:41.000000+0000	1
SEVENTEEN, Ailee	2019-10-27 14:20:36.000000+0000	1
iKON	2019-10-27 10:49:12.000000+0000	1
iKON	2019-10-27 13:28:15.000000+0000	1
iKON	2019-10-27 13:30:36.000000+0000	1

Figure 1. Table stored in Cassandra

3. **Visualization**

   We create the visualization part in Python using Plotpy and Dash library [5]. Using these make us possible to live plot the data in real-time on the web browser. We retrieve the data from the Cassandra table every 5 seconds wherein each query we create a new summary table for all data from the past 60 minutes based on the value in `created_at` column. The summary table contains `artist`, `date`, and `total_count` column. To obtain this summary table, first, we group the data by the `artist` column, then we sum the value on the `count` column and put it in the `total_count` column. The `date` column value is added based on the time when each data is processed. After that, we sort the data by the total_count and pick only the top 20 to be visualized.

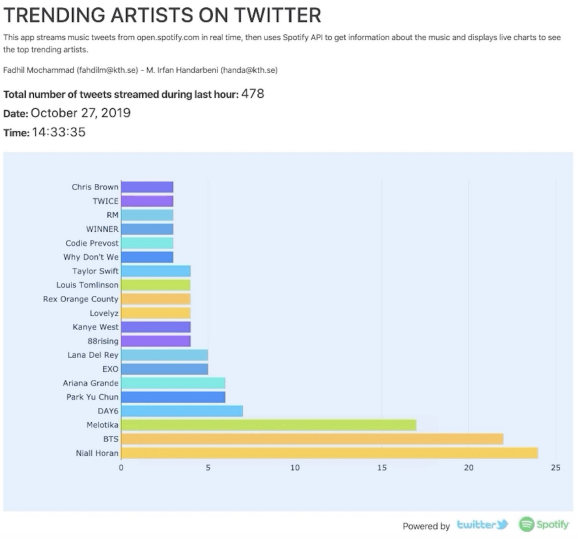

Figure 2. Visualization

Run the Code

The first thing to do, start the Kafka Server, Zookeeper Server, start Cassandra, and create the topic on Kafka :

```
$KAFKA_HOME/bin/zookeeper-server-start.sh $KAFKA_HOME/config/zookeeper.properties
$KAFKA_HOME/bin/kafka-server-start.sh $KAFKA_HOME/config/server.properties
$KAFKA_HOME/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1
--partitions 1 --topic <kafka_topic>
$CASSANDRA_HOME/bin/cassandra -f
```

`$KAFKA_HOME` and `$CASSANDRA_HOME` are the environment variables which contain the path of Kafka and Cassandra on your computer. Put your Kafka topic on the `<kafka_topic>`.

Make sure that all the python libraries that are needed for this project are already installed. Run the command below on your terminal.

```
pip install -r requirements.txt
```

After all the libraries already installed, create a file name “credential.json” and put the twitter API credential and Spotify API credential in JSON format.

```
{
    "twitter_api_key": <twitter_api_key>,
    "twitter_api_secret" : <twitter_api_secret>,
    "twitter_token": <twitter_token>,
    "twitter_token_secret" : <twitter_token_secret>,
    "SPOTIFY_CLIENT_ID" : <SPOTIFY_CLIENT_ID>,
    "SPOTIFY_CLIENT_SECRET" : <SPOTIFY_CLIENT_SECRET?
}
```

Run the TwitterStreaming.py and put the Kafka topic as the argument. This program will stream data from Twitter and produce data to a specified topic in Kafka.

```
python3 TwitterStreaming <kafka_topic>
```

Run the kafkaStreaming.py by using `spark-submit` because you need to add `spark-streaming-kafka` and `pyspark-cassandra`[6] library to be able to connect the Spark to Cassandra. Put also the Kafka topic as the argument. This program will consume data for a specified topic from Kafka, do some processing on Spark, and store the result on Cassandra.

```
spark-submit --packages
anguenot/pyspark-cassandra:2.4.0,org.apache.spark:spark-streaming-kafka-0-8_2.11:2.1.1
--conf spark.cassandra.connection.host=127.0.0.1 kafkaStreaming.py <kafka_topic>
```

Start the visualization by running the visualize.py file and go to 127.0.0.1:8050 in the web browser.

```
python visualize.py
```

You can also see the result data stored in database using Cassandra CQL shell.

```
$CASSANDRA_HOME/bin/cqlsh
cqlsh> SELECT * FROM <keyspace_name>.<table_name>;
```

Note: To be able to run this project, make sure that you use the same libraries and frameworks version because not all versions of the libraries and frameworks are compatible with each other. The python libraries version can be seen on the requirement.txt file. The frameworks versions were used for this project:

- Spark: 2.4.4
- Python: 3.7.4
- Cassandra: 3.11.4
- Kafka: 2.3.0

Result

We successfully stream data in real-time fashion from twitter and use Kafka as a message broker before consuming it for the next process. The information from the acquired tweets is extracted and processed to produce information about the most shared music artist on Twitter. After that, we visualize the result so we can see who are the most popular artists on Twitter within the given interval time. From the visualized result of our experiments, we can see that Pop artists are dominating as the most shared on Twitter. That is as expected as most people listen to Pop music. However, there is also an interesting insight from the visualization of the top 20 artists. There are a lot of K-Pop artists inside the list, which might indicate that South Korea has successfully spread its influence in the entertainment industry. Nonetheless, this project had been an exciting experience for us to be able to explore various distributed-computing platforms and APIs for acquiring, processing and storing data. This project could be extended by incorporating sentiment analysis to the text of the tweets and generate the sentiment of twitter user's on the shared songs/artists. But we did not implement it in this project due to limited time, and it would be out of the scope of the project and course.

Reference

[1] Twitter API: https://developer.twitter.com/en.html
[2] Tweepy Python Library: https://www.tweepy.org/
[3] Spotipy Python Library: https://spotipy.readthedocs.io/en/latest/
[4] Spotify Web API: https://developer.spotify.com/documentation/web-api/
[5] Plotpy and Dash Python Library: https://plot.ly/dash/
[6] pyspark-cassandra Library: https://github.com/anguenot/pyspark-cassandra