DataComp2 - Project Spark & Elasticsearch

Yacine Mokhtari
Lilia Izri
Alexandre Combeau

May 5, 2022

1 Introduction and Motivations

The goal of our project is to make a short program to process and analyze real-time stream data (in this case, tweets) using Sparklab Streaming[4], index some of this data with ElasticSearch[1] and evaluate some queries with it. Our project will be devided in three main parts that we will detail in this report.

Figure 1: Schema

2 Processing Twitter Data

2.1 Settings: Defining Twitter developer credentials and Authenticating the connection

The first step was to create a Twitter Application[3] so as we can get both the Consumer Keys and the Authentication Tokens. Using the `tweepy` library[5], we can retrieve Tweets from the Twitter API and store them into sockets before making any analysis on them as illustrated in figure1.
For this project, we focused on tweets containing the keyword `netflix`.

2.2 Stream Listener Class

In order to get the tweets in real-time, we have to implement a new class that inherits from the `Stream` module of `Tweepy`. The constructor takes a `Socket` object and calls the super constructor, pretty straightforward. Then, we overwrite the `on_data`[6] method to adapt it to our needs.

```
#  We combine data and metadata to send them
#  We add a key ###:field:### so we can split the fields easily
#  We remove '\n' from a tweet and put one '\n' between tweets
tweet_info = (" ###:field:### user: " + user + " ###:field:### tweet: " + text +\
              " ###:field:### date: " + date + " ###:field:### location: " + location + \
              " ###:field:### hashtags: " + hashtags
              ).replace('\n', ' ') + '\n'

print(tweet['user']['location'])
print(tweet_info)

# Send to socket : We convert this tweet into a bite code (since spark takes easily this kind of data)
self.client_socket.send(tweet_info.encode('utf-8'))
```

Figure 2:

The `on_data` method will format the received JSON file (a tweet) into a string. We only keep some features such as the name, the location, hashtags (if any) and the date. These different fields are are delimited by the tag `"###field### field_name:"`.

The location is obtained using the geopy library[2] with the captured location of the tweet. Since the location can be something the user created, we handle these cases by returning an unknown location. The geopy library gives us the latitude and longitude which we will use when classifying those tweets with the machine learning component of Spark.

> Note that the \n (linebreaks) that appear in the tweet text are replaced by a simple space " ".
> Doing this makes sure that each tweet corresponds to just one line and it's correctly read when creating the DStream in spark with the `socketTextStream()` function.

2.3 Establish connexion with client

After filtering and formating the received data, we send it to a TCP/Socket through the port 5552. As we said earlier, this will be the entry point of our Spark Streaming listener :

```
new_skt = socket.socket()         # initiate a socket object
host = "127.0.0.1"                # address host
port = 5552                       # specifie port
new_skt.bind((host, port))        # Binding host and port
```

```
print(f"Now listening on port: {port}")
new_skt.listen(5)                 # waiting for client connection
c, addr = new_skt.accept()        # Establish connection with client

print(f"Received request from: {addr}")
send_tweets(c)                    # send tweets to client socket
```

Figure 3:

3 Spark Analysis

3.1 Process data

We process the data we receive by first, spliting the incoming string so each row in our datastream `'tweets'` be a tuple where each element will correspond to a field of the tweet received. And using `textblob`, we perform sentiment analysis on the text of the tweet and we add this sentiment to the previous tuple. So each tweet will be represented by a tuple of the form (user, text, date, location, hashtags, sentiment) where sentiment equals -1 if the tweet is more likely negative. It will equal 1 if it's more positive, and 0 otherwise.

> **Remark:** We chosed to take into account the polarity returned by the `TextBlob text processing library` and to ignore the subjectivity.

3.2 Training a Machine Learning Algorithm

For our case, we decided to go with the `Streaming k-means` algorithm. This is a simple, yet interesting algorithm to get started with the ML component of Spark. We will be classifying tweets using their sentimental analysis (returned by TextBlob) and their location (both latitude and longitude).

4 ElasticSearch Indexing

References

[1] E. Elastichsearch. Official documentation. https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html.

[2] geopy. Tutorial. https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.mllib.clustering.StreamingKMeans.html.

[3] T. Inc. Twitter developer portal. https://dev.twitter.com/apps/new.

[4] S. Streaming. Official documentation. https://spark.apache.org/docs/latest/streaming-programming-guide.html.

[5] Tweepy. Index - official documentation. https://docs.tweepy.org/en/stable/index.html.

[6] Tweepy. Streaming. https://docs.tweepy.org/en/stable/streaming.html.