DataComp2 - Project Spark & Elasticsearch

Yacine Mokhtari
Lilia Izri
Alexandre Combeau

May 6, 2022

1 Introduction and Motivations

The goal of our project is to make a short program that processes and analyses 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 How to run the project

Run the three notebooks in this order :
(1) `receive_tweets.ipynb`.
(2) `spark.ipynb`.
(3) `elasticsearch.ipynb`.
Remark: The `requirements.txt` file and `utils.py` have to be in the same directory as the 3 notebooks.

---

3 Processing Twitter Data

3.1 Settings: Defining Twitter developer credentials and Initializing 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 `Depp`.

3.2 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 clustering 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.

3.3 Establish connexion with client

After filtering and formating the received data, we send it to a TCP/Socket through a port (for exemple here 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:

4 Spark Analysis

4.1 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, latitude, longitude, 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 chose to take into account only the polarity returned by the `TextBlob text processing library` and to ignore the subjectivity.

4.2 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).
After training the algorithm and testing it, we compute the number of elements assigned to each cluster using `ReduceByKeyAndWindow()` function and we print the results.

```
-------------------------------------------
Time: 2022-05-06 11:48:30
-------------------------------------------
(0.0, 2)
(1.0, 0)
(0.0, 0)
(0.0, 2)
(0.0, 2)
(0.0, 0)
(0.0, 0)
(0.0, 2)
(1.0, 1)

-------------------------------------------
Time: 2022-05-06 11:48:30
-------------------------------------------
(0, 16)
(1, 1)
(2, 11)
```

Figure 4: Clusters

The first batch of pairs we can see, represents the predictions of the kmeans algorithm. They are of the form (label of tweet, the cluster it was assigned to), we chose as label for a tweet the sentiment. For example, the first pair means the label of the first tweet is 0 and it's assigned to the cluster 2.

The second pairs represent the result of the operation by window: The size of each cluster. Each line is of the form (The index of the cluster, the number of elements assigned to it in the X last seconds) where X is the size of the window. So here, the cluster "0" has 16 tweets.

5 ElasticSearch Indexing

We created a function `tweetToJSON()` (in utils.py) that allows us to transform a tweet to a json then index it in `ElasticSearch`.
We evaluate after that some multi-term queries and some queries with regular expressions.

Remarks:
We have tested different ways to evaluate the queries (terminal with curl, and using `ElasticSearch` directly in the python notebook.

Here are some queries we have tested (we have fixed the number of results to 2, so it's readable...):

- Simple match term ('lawyer') query with boost:

Query where we check for a term with a boost

```
[3]: term2 = "lawyer"
query_with_boost = {
    "span_multi": {
      "match": {
        "prefix": { "tweet": { "value": term2, "boost": 1.08 } }
      }
    }
}
es.search(index=index, query=query_with_boost, size=2)
```


```
[3]: {'took': 5,
 'timed_out': False,
 '_shards': {'total': 1, 'successful': 1, 'skipped': 0, 'failed': 0},
 'hits': {'total': {'value': 11, 'relation': 'eq'},
  'max_score': 20.22163,
  'hits': [{'_index': 'filtered_tweets',
    '_type': '_doc',
    '_id': '590228693',
    '_score': 20.22163,
    '_source': {'user': 'NadinetTr',
     'tweet': '@lawyerschiff that is false https://t.co/K4D50BO2kB https://
t.co/EafsRBuKau',
     'date': 'Fri May 06 14:10:42 +0000 2022',
     'lat': '61.0666922',
     'lon': '-107.991707',
     'hashtags': '',
     'sentiment': '-1',
     'id': '590228693'}},
   {'_index': 'filtered_tweets',
    '_type': '_doc',
    '_id': '1270491551043596289',
    '_score': 16.796377,
    '_source': {'user': 'luminu2020',
     'tweet': 'RT @MissAkuaAfriyie: Johnny Depp's lawyers just need to read
tweets. Fact checkers everywhere and they come with receipts',
     'date': 'Fri May 06 13:29:20 +0000 2022',
     'lat': '44.933143',
     'lon': '7.540121',
     'hashtags': '',
     'sentiment': '0',
     'id': '1270491551043596289'}}]}}
```


Figure 5: Query 1

- Search in different fields (user, tweet) the term ('Kate'):

Check for a term in the text of the tweet or the user

```
[4]: term3 = "Kate"
query_body = {
    "multi_match": {
        "query": term3,
        "type": "most_fields",
        "fields": ["tweet", "user"]
    }
}

es.search(index=index, query=query_body, size=2)
```


```
[4]: {'took': 4,
 'timed_out': False,
 '_shards': {'total': 1, 'successful': 1, 'skipped': 0, 'failed': 0},
 'hits': {'total': {'value': 17, 'relation': 'eq'},
  'max_score': 3.6380336,
  'hits': [{'_index': 'filtered_tweets',
    '_type': '_doc',
    '_id': '15165186372925112 32',
    '_score': 3.6380336,
    '_source': {'user': 'moviezclubs2',
     'tweet': 'What Did Amber Heard Assistant Kate James Say About Johnny De
pp? Explained\xa0– https://t.co/tM1rvUh3wO',
     'date': 'Fri May 06 14:11:56 +0000 2022',
     'lat': '44.933143',
     'lon': '7.540121',
     'hashtags': '',
     'sentiment': '0',
     'id': '1516518637292511232'}},
   {'_index': 'filtered_tweets',
    '_type': '_doc',
    '_id': '3059665419',
    '_score': 3.3363068,
    '_source': {'user': 'iffycanfly',
     'tweet': "RT @IceQueenCherie: @stfnysnddn Yes, because Kate Moss is Dep
p's ex gf.  Amber's testimony made relevant previous relationships, hence J
D's…",
     'date': 'Fri May 06 13:28:56 +0000 2022',
     'lat': '39.7837304',
     'lon': '-100.445882',
     'hashtags': '',
     'sentiment': '1',
     'id': '3059665419'}}]}
```


Figure 6: Query 2

- Using regexp (search tweets with no in user's name):

Query with a regexp and highlight

```
[5]: field = "user"
regexp = "no.*"

body_query = {
  "query": {
    "regexp": {
      field: {
        "value": regexp,
        "flags": "ALL",
        "case_insensitive": False,
        "max_determinized_states": 10000,
        "rewrite": "constant_score"
      }
    }
  },
  "highlight" : {
        "pre_tags" : ["<em>"],
        "post_tags" : ["</em>"],
        "fields" : {
            "user" : {}
        }
    }
}
es.search(index=index, body=body_query, size=2)
```

Figure 7: Query 3 - body

```
[5]: {'took': 8,
 'timed_out': False,
 '_shards': {'total': 1, 'successful': 1, 'skipped': 0, 'failed': 0},
 'hits': {'total': {'value': 3, 'relation': 'eq'},
  'max_score': 1.0,
  'hits': [{'_index': 'filtered_tweets',
    '_type': '_doc',
    '_id': '197204625',
    '_score': 1.0,
    '_source': {'user': 'NoBombardeenUIO',
     'tweet': 'Johnny Depp:  ~"Amber era tan mala que me hizo acompañarle a
presentar una demanda ante la FIFA para que a Ecuador… https://t.co/TBIFW32q
gc',
     'date': 'Fri May 06 14:11:29 +0000 2022',
     'lat': '25.029422',
     'lon': '-77.36195598496681',
     'hashtags': '',
     'sentiment': '0',
     'id': '197204625'},
    'highlight': {'user': ['<em>NoBombardeenUIO</em>']}},
   {'_index': 'filtered_tweets',
    '_type': '_doc',
    '_id': '1309527365509427206',
    '_score': 1.0,
    '_source': {'user': 'nocturnabelle',
     'tweet': 'RT @hleneeh: the way people jump to use ableist language when
discussing depp v. heard is something else',
     'date': 'Fri May 06 14:10:38 +0000 2022',
     'lat': '40.7127281',
     'lon': '-74.0060152',
     'hashtags': '',
     'sentiment': '0',
     'id': '1309527365509427206'},
    'highlight': {'user': ['<em>nocturnabelle</em>']}}]}}
```

Figure 8: Query 3 - result

- Match query using elasticsearch_dsl:

```
[6]: from elasticsearch_dsl import Search

# Search engine
s = Search(using=es, index=index)
```

Query where "heard" appears in the tweet text

```
[7]: term = "heard"
result = s.query('match', tweet=term).execute()
for hit in result:
    print(hit.tweet)
```

```
RT @GellertDeppBR: Amber Heard: Na Austrália fui espancada, torturada e est*
prada por Johnny Depp.  Amber Heard voltando da Austrália:  #Ju…
RT @samjrakoh: amber heard listening to kate james' testimony of amber's abu
se of her/amber heard listening to audio of herself abusing joh…
I would LOVE to see the responses of the Amber Heard sympathisers when they
get shown the audio of Heard literally… https://t.co/dZSeoSy6Co
RT @KinelRyan: So strange that Amber Heard is suddenly telling stories that
no one has ever heard before about Johnny Depp, that no one can…
RT @SystemicDunking: I feel like I'm watching Amber heard playing the role o
f Amber heard in a made for tv movie about Johnny Depp. Its wil…
RT @ellisgreg: Believe all women? Even Amber Heard? https://t.co/7OCYTx6qqV
RT @ellisgreg: Believe all women? Even Amber Heard? https://t.co/7OCYTx6qqV
Rayuan Maut Johnny Depp saat Lamar Amber Heard https://t.co/bUmDE5WdTq
Sinestas perdudo enbeo casobde Depp Vs Heard, acá te enteras
RT @dougnappa: Amber Heard testemunhando contra o Johnny Depp: https://t.co/
pgknkoItfu
```

Figure 9: Query 4

We can see that it's simpler to evaluate queries using the high-level library.

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.