Université Paris-Saclay

Large-Scale Distributed Data Processing course

Spark Project Report

Maxime Jauroyon
Mahdi Ranjbar
Tonga Junior Cedric

Department of Computer science

Course's teachers :
Dr.Benoît Groz
Dr. Silviu Maniu

May 2, 2023

Table of Contents

1 Introduction 1
2 Streaming 1
2.1 Server 1
3 Spark 2
3.1 Data reception and processing 2
3.2 Clustering 2
3.3 Evaluation of the cluster size in sliding windows 2
4 Visualisation 3
5 Conclusion 4

1 Introduction

The objective of this project is to demonstrate the real-time processing and analysis of data using Spark Streaming and a streaming API. We chose Reddit as our source of data for this project. Our approach involves creating a Python server to connect to the Reddit API and retrieve individual messages and metadata, performing sentiment analysis on the data using TextBlob, and analyzing the data in Spark Streaming using an ML algorithm and window operations. Finally, we visualize the insights generated by our analysis in real-time. In this report, we present the different steps we followed, from connecting to the Reddit server, retrieving the data, analyzing it, and visualizing the results.

2 Streaming

2.1 Server

The `stream.ipynb` implements a server for streaming data from Reddit API. The server is implemented using sockets and multithreading to handle multiple clients simultaneously. The program uses the Python Reddit API Wrapper (PRAW) library to stream submissions from `'all'` subreddit[1] and sends them to connected clients in real-time.

The `RedditAPIServer` class is the main class that encapsulates the functionality of the server. It initializes the necessary variables, creates a server socket, and starts the streaming thread. The class also provides methods to start and stop the server.

The `start()` method initializes the client, creates a server socket, starts the streaming thread, and accepts incoming client connections. The stop method closes the server socket and disconnects all connected clients.

The streaming thread runs in the background and continuously streams data from all subreddits using the PRAW library. It extracts relevant data from each submission (post) in the subreddit and sends it to all connected clients using a socket connection.

The `_accept_clients` method continuously listens for incoming client connections and creates a new thread for each connected client. Each client thread runs

[1] A subreddit is a community on Reddit dedicated to a specific topic where users can share links, images, and text related to that topic and engage in discussions with other users who share their interests.

the `_handle_client` method, which receives data from the client and removes disconnected clients from the list of connected clients.

3 Spark

In this step, we connect to the server made above, we analyze and gather them in cluster.

3.1 Data reception and processing

It is done following the different steps:

- Creation of the Dstream that connects to our server
- Creation of a function that takes in input a message and process it as following:
  - Load the data
  - Apply a sentiment analysis with textblob to extract polarity and subjectivity
  - Transformation of the subreddits from string type to numeric value to return the hash value every time. This will be useful for the clustering algorithm.
  - Return the text (message) and other information like subreddit value, polarity, subjectivity.

3.2 Clustering

We clustered the data according to the polarity, subjectivity and subreddit which plays the role of location in our case. We then used the implementation of K-Means in `MLLIB` which is `streamingKmeans` with 4 clusters. We then predicted the cluster of the data after transforming them into vector.

3.3 Evaluation of the cluster size in sliding windows

We use a window to group messages from the same group and evaluate the group size. We then calculated the size of each group in a sliding window of 30 seconds with a sliding interval of 10 seconds.

We used the *reduceByKeyAndWindow* function of spark streaming which requires a checkpoint, so we created a checkpoint named checkpoint_spark :

```
1 sc.setCheckpointDir("checkpoint_spark")
```

Note: In windows, we had a problem with this because of the dependencies and needs to install spark on windows. But it works fine on Linux and Mac.

4 Visualisation

We wrote a function that will accumulate the size of each cluster in a json format and update it while saving each time as a database under the name **viz.json**.

We then created another notebook **viz.ipynb** for visualization. We started by creating an empty dataframe with two columns cluster and size. Then in an infinite loop, we retrieve each time the whole data saved in the json file viz.json. Then we create another dataframe with the same columns that we add to the first dataframe each time. This allows us to add the cluster and size data to the first dataframe each time. Then we fold the barplots related to the collected data. We have paused for a while to allow the data to be recovered.
The visualization as you can see in figure 1 gives us the size of the clusters in a dynamic way. It allows us to know how the feelings (neutral, positive and negative) are distributed in each cluster.

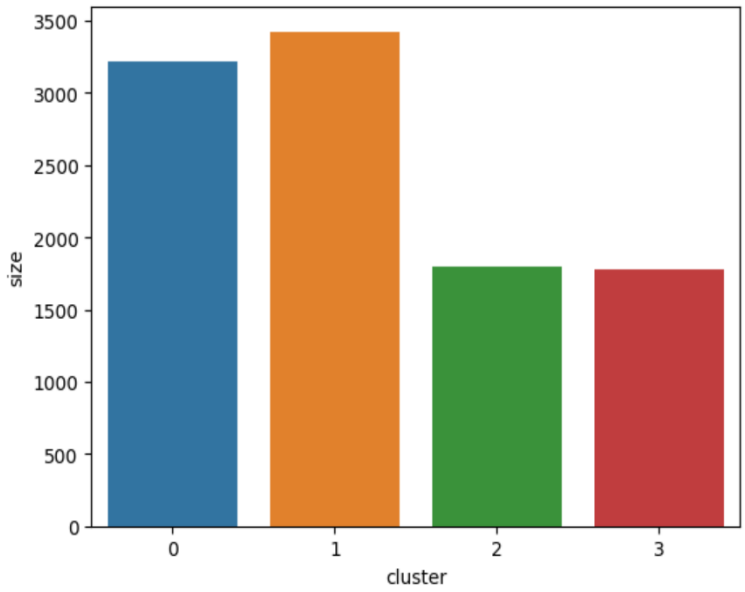

Figure 1: visualization of cluster size in a dynamic way

5 Conclusion

This project has allowed us to apply the concepts seen in class and to understand even more how the spark works. Also it allowed us to see how data streaming works!