Lab 04: Streaming Data Processing with Spark

CSC14118 Introduction to Big Data 20KHMT1

SmallData

2023-05-22

Contents

1 Lab 04: Streaming Data Processing with Spark

1.1 List of team members

ID	Full Name	Completed
20120366	Pham Phu Hoang Son	100%
20120391	Ha Xuan Truong	100%
20120393	Huynh Minh Tu	100%
20120468	Nguyen Van Hai	100%

1.2 Team's result

Section	Point
Get Twitter tweets	1.5
Stream tweets to Apache Spark	4
Perform sentiment analysis on tweets	2
Visualize the analytic results	2.5

1.3 Team reflection

Does your journey to the deadline have any bugs? How have you overcome it?

While working on lab04, "Streaming Data Processing with Spark," I came across the following issues:

- Initially, I tried to run Streaming Data with Spark on Google Colab, but unfortunately, it didn't work as expected. As a result, our team made the decision to switch to running the Python file locally.

- Another challenge we faced was the slow performance and frequent errors during the streaming process. Whenever we ran the code, it only processed a single batch of data before encountering issues.

What have you learned after this process?

From the experience of working on lab04 and encountering the mentioned issues, several valuable lessons can be learned:

- Platform compatibility: It is important to consider the compatibility of the tools and platforms being used for the project. In this case, attempting to run Streaming Data with Spark on Google Colab resulted in compatibility issues. Therefore, it is crucial to thoroughly test and ensure that the chosen platform supports the required dependencies and functionalities.
- Local execution as an alternative: When facing challenges with a specific platform, having a backup plan or alternative approach is essential. In this situation, switching to running the Python file locally proved to be a viable solution. It is crucial to be adaptable and prepared to modify the execution environment to overcome obstacles and ensure progress.
- Performance optimization: The slow speed and frequent errors during the streaming process highlight the importance of performance optimization. Stream processing can be resource-intensive, and it is essential to analyze and fine-tune the code to improve efficiency. This may involve optimizing Spark configurations, leveraging appropriate data structures, or implementing parallel processing techniques.
- Error handling and resilience: When working with streaming data, it is crucial to anticipate and handle errors effectively. The fact that the code processed only a single batch of data before encountering issues suggests the need for robust error handling mechanisms. This may involve implementing error monitoring, graceful failure handling, and appropriate error recovery strategies to ensure the stability and reliability of the streaming process.
- Continuous testing and debugging: Streaming data processing can be complex, and continuous testing and debugging are vital. It is important to regularly validate the code, test various scenarios, and closely monitor the system's behavior. Prompt identification and resolution of bugs and issues contribute to smoother and more reliable data processing.

1.4 Get Twitter tweets

1.4.1 Step 1: Download data from https://huggingface.co/datasets/deberain/ChatGPT-Tweets.

1.4.2 Step 2: Download mongoDB in ubuntu.

```
sudo apt-get install gnupg

curl -fsSL https://pgp.mongodb.com/server-6.0.asc | \
sudo gpg -o /usr/share/keyrings/mongodb-server-6.0.gpg \
--dearmor

echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-6.0.gpg ] https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list

sudo apt-get update

sudo apt-get install -y mongodb-org
```

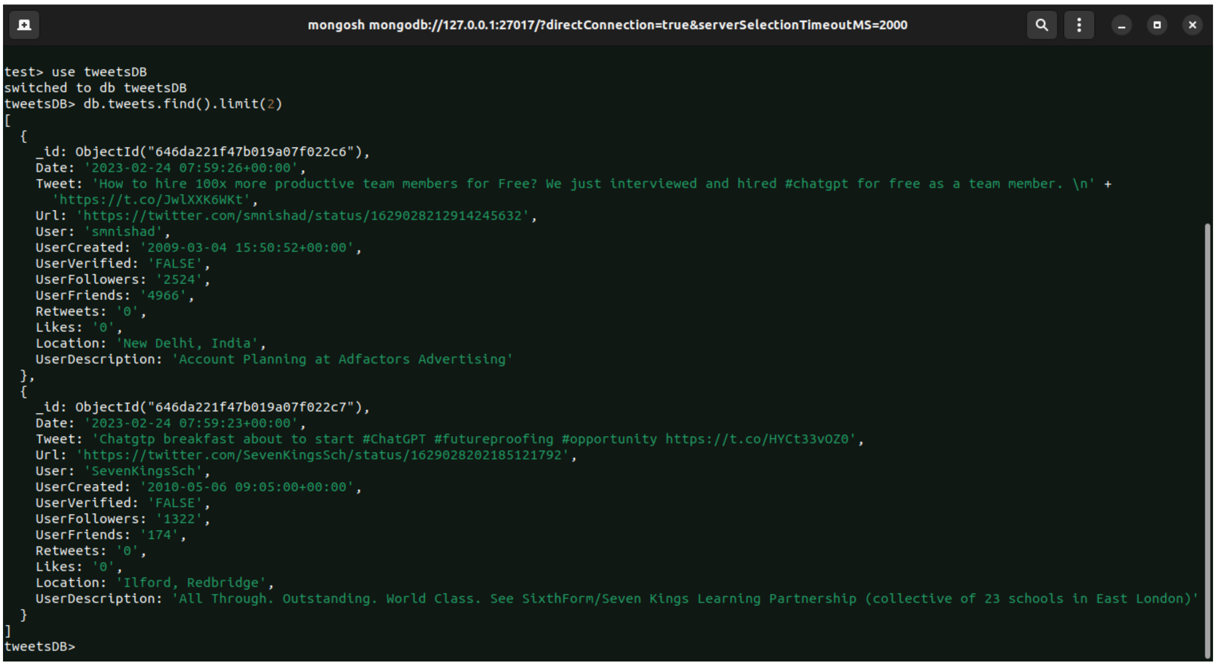
Run mongoDB in terminal:

```
sudo systemctl start mongod

sudo systemctl status mongod
```

1.4.3 Step 3: Save tweets data to mongoDB by pymongo

Run file data_to_mongo.py to save tweets data in mongoDB



mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
test> use tweetsDB
switched to db tweetsDB
tweetsDB> db.tweets.find().limit(2)
[
  {
    _id: ObjectId("646da221f47b019a07f022c6"),
    Date: '2023-02-24 07:59:26+00:00',
    Tweet: 'How to hire 100x more productive team members for Free? We just interviewed and hired #chatgpt for free as a team member. \n' +
      'https://t.co/JwlXXK6WKt',
    Url: 'https://twitter.com/smnishad/status/1629028212914245632',
    User: 'smnishad',
    UserCreated: '2009-03-04 15:50:52+00:00',
    UserVerified: 'FALSE',
    UserFollowers: '2524',
    UserFriends: '4966',
    Retweets: '0',
    Likes: '0',
    Location: 'New Delhi, India',
    UserDescription: 'Account Planning at Adfactors Advertising'
  },
  {
    _id: ObjectId("646da221f47b019a07f022c7"),
    Date: '2023-02-24 07:59:23+00:00',
    Tweet: 'Chatgtp breakfast about to start #ChatGPT #futureproofing #opportunity https://t.co/HYCt33vOZ0',
    Url: 'https://twitter.com/SevenKingsSch/status/1629028202185121792',
    User: 'SevenKingsSch',
    UserCreated: '2010-05-06 09:05:00+00:00',
    UserVerified: 'FALSE',
    UserFollowers: '1322',
    UserFriends: '174',
    Retweets: '0',
    Likes: '0',
    Location: 'Ilford, Redbridge',
    UserDescription: 'All Through. Outstanding. World Class. See SixthForm/Seven Kings Learning Partnership (collective of 23 schools in East London)'
  }
]
tweetsDB>
```

Figure 1.1: Tweets to MongoDB

1.5 Stream tweets to Apache Spark

We using kafka to stream tweets from mongoDB to apache Spark.

1.5.1 Step 1: Install Apache Spark.

Download Spark in this link: https://www.apache.org/dyn/closer.lua/spark/spark-3.4.0/spark-3.4.0-bin-hadoop3.tgz

Unzip and save to './Home'. Rename to Spark. Open terminal and run this command:

```
nano ~/.bashrc
```

Add and replace username to user ubuntu

```
export SPARK_HOME=/home/{username}/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

Return Terminal and run:

```
source ~/.bashrc
```

Add jars:

```
cd $SPARK_HOME/jars && wget https://repo1.maven.org/maven2/org/mongodb/spark/mongo
spark-connector_2.12/10.1.1/mongo-spark-connector_2.12-10.1.1.jar
```

```
cd $SPARK_HOME/jars && wget https://repo1.maven.org/maven2/org/mongodb/mongodb-
driver/3.12.13/mongodb-driver-3.12.13.jar
```

```
cd $SPARK_HOME/jars && wget https://repo1.maven.org/maven2/org/mongodb/mongo-
java-driver/3.12.13/mongo-java-driver-3.12.13.jar
```

```
cd $SPARK_HOME/jars && wget https://repo1.maven.org/maven2/org/mongodb/bson/4.9.1/
4.9.1.jar
```

1.5.2 Step 2: Install kafka

Download Kafka in this link: https://downloads.apache.org/kafka/3.4.0/kafka_2.12-3.4.0.tgz

Unzip and add to '/Home'

Rename to Kafka

Open terminal and run:

```
cd /home/{username}/Kafka
```

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

Open new terminal and run:

```
cd /home/{username}/Kafka
```

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

Open new terminal and run this command to create a topic of Kafka:

```
bin/kafka-topics.sh --create --topic twitterstream --bootstrap-
server localhost:9092

# verify
bin/kafka-topics.sh --describe --topic twitterstream --bootstrap-
server localhost:9092
```

1.5.3 Step 3: Install some Python libraries

Open terminal and install some libraries

```
pip install datasets
pip install pymongo
pip install textblob
pip install findspark
```

1.5.4 Step 4: Write data from mongoDB to kafka

- Firstly, we convert all columns to a struct column.
- Secondly,convert struct column to JSON and rename to "value"
- Lastly, We save tweets to kafka each 15 seconds with 10000 rows

You can find source code in file mongo_to_kafka.py to see our step by step and get a deep understanding of how to write data to kafka.

Run file mongo_to_kafka.py by command:

```
spark -submit --packages org.apache.spark:spark-streaming-kafka-0-
12_2:3.4.0,org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.0 mongo_to_kafka.py
```

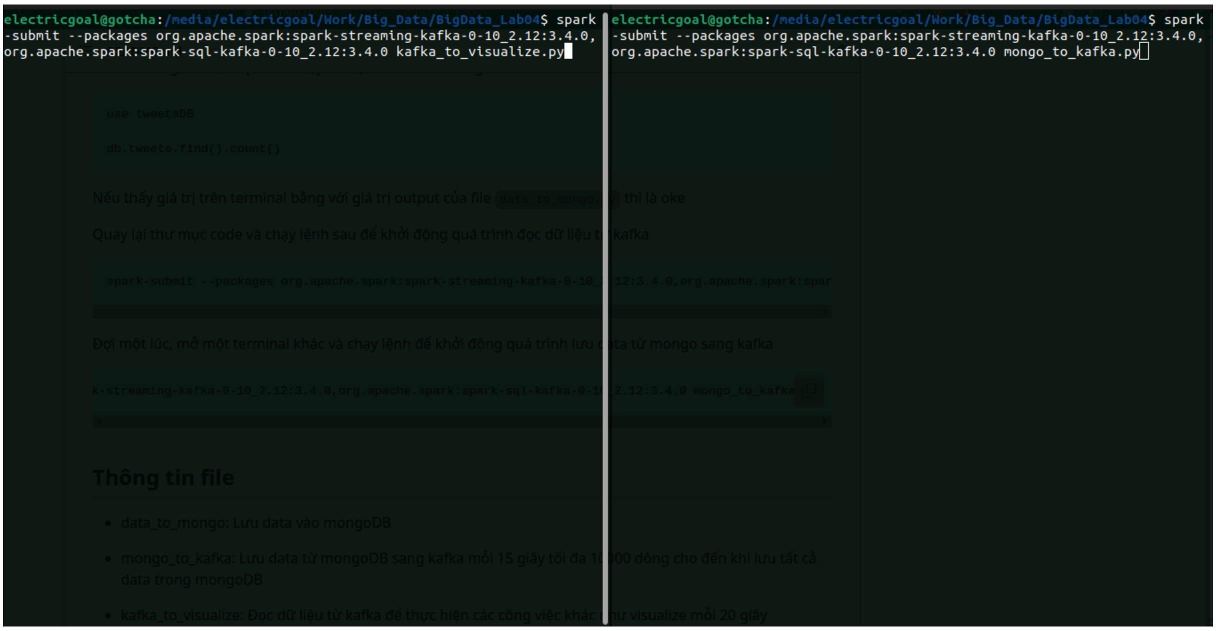
1.5.5 Step 5: Streaming data

ou can find source code in file kafka_to_visualize.py to see our step by step and get a deep understanding of how to streaming data by spark.

Run file kafka_to_visualize.py by command:

```
spark -submit --packages org.apache.spark:spark-streaming-kafka-0-
12_2:3.4.0,org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.0 kafka_to_visualize.py
```

Command to run two file mongo_to_kafka.py and kafka_to_visualize.py:

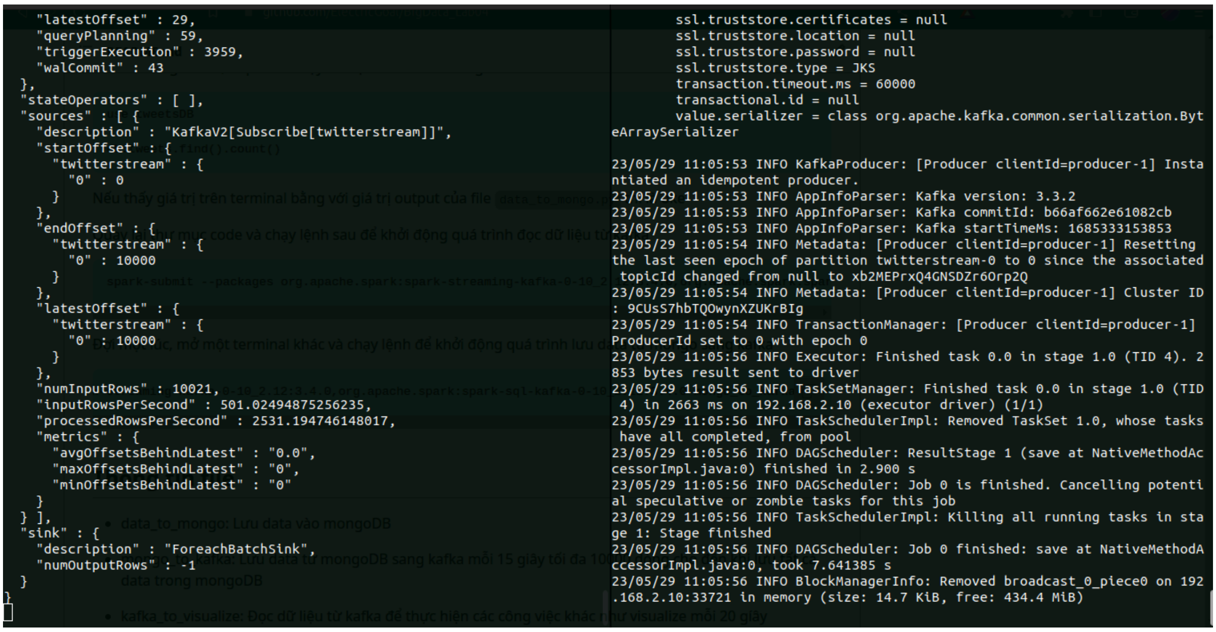


```
electricgoal@gotcha:/media/electricgoal/Work/Big_Data/BigData_Lab04$ spark
-submit --packages org.apache.spark:spark-streaming-kafka-0-10_2.12:3.4.0,
org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.0 kafka_to_visualize.py
```

```
electricgoal@gotcha:/media/electricgoal/Work/Big_Data/BigData_Lab04$ spark
-submit --packages org.apache.spark:spark-streaming-kafka-0-10_2.12:3.4.0,
org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.0 mongo_to_kafka.py
```

Figure 1.2: Command to run

Results after running:



```
    "latestOffset" : 29,
    "queryPlanning" : 59,
    "triggerExecution" : 3959,
    "walCommit" : 43
  },
  "stateOperators" : [ ],
  "sources" : [ {
    "description" : "KafkaV2[Subscribe[twitterstream]]",
    "startOffset" : {
      "twitterstream" : {
        "0" : 0
      }
    },
    "endOffset" : {
      "twitterstream" : {
        "0" : 10000
      }
    },
    "latestOffset" : {
      "twitterstream" : {
        "0" : 10000
      }
    },
    "numInputRows" : 10021,
    "inputRowsPerSecond" : 501.02494875256235,
    "processedRowsPerSecond" : 2531.194746148017,
    "metrics" : {
      "avgOffsetsBehindLatest" : "0.0",
      "maxOffsetsBehindLatest" : "0",
      "minOffsetsBehindLatest" : "0"
    }
  } ],
  "sink" : {
    "description" : "ForeachBatchSink",
    "numOutputRows" : -1
  }
}
```

```
        ssl.truststore.certificates = null
        ssl.truststore.location = null
        ssl.truststore.password = null
        ssl.truststore.type = JKS
        transaction.timeout.ms = 60000
        transactional.id = null
        value.serializer = class org.apache.kafka.common.serialization.Byt
eArraySerializer

23/05/29 11:05:53 INFO KafkaProducer: [Producer clientId=producer-1] Insta
ntiated an idempotent producer.
23/05/29 11:05:53 INFO AppInfoParser: Kafka version: 3.3.2
23/05/29 11:05:53 INFO AppInfoParser: Kafka commitId: b66af662e61082cb
23/05/29 11:05:53 INFO AppInfoParser: Kafka startTimeMs: 1685333153853
23/05/29 11:05:54 INFO Metadata: [Producer clientId=producer-1] Resetting
the last seen epoch of partition twitterstream-0 to 0 since the associated
 topicId changed from null to xb2MEPrxQ4GNSDZr6Orp2Q
23/05/29 11:05:54 INFO Metadata: [Producer clientId=producer-1] Cluster ID
: 9CUsS7hbTQOwynXZUKrBIg
23/05/29 11:05:54 INFO TransactionManager: [Producer clientId=producer-1]
ProducerId set to 0 with epoch 0
23/05/29 11:05:56 INFO Executor: Finished task 0.0 in stage 1.0 (TID 4). 2
853 bytes result sent to driver
23/05/29 11:05:56 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID
 4) in 2663 ms on 192.168.2.10 (executor driver) (1/1)
23/05/29 11:05:56 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks
 have all completed, from pool
23/05/29 11:05:56 INFO DAGScheduler: ResultStage 1 (save at NativeMethodAc
cessorImpl.java:0) finished in 2.900 s
23/05/29 11:05:56 INFO DAGScheduler: Job 0 is finished. Cancelling potenti
al speculative or zombie tasks for this job
23/05/29 11:05:56 INFO TaskSchedulerImpl: Killing all running tasks in sta
ge 1: Stage finished
23/05/29 11:05:56 INFO DAGScheduler: Job 0 finished: save at NativeMethodA
ccessorImpl.java:0, took 7.641385 s
23/05/29 11:05:56 INFO BlockManagerInfo: Removed broadcast_0_piece0 on 192
.168.2.10:33721 in memory (size: 14.7 KiB, free: 434.4 MiB)
```

Figure 1.3: Result

1.6 Perform sentiment analysis on tweets

The `sentiment(res_df)` function takes a DataFrame, performs text cleaning on the "Tweet" column, performs sentiment analysis, classifies the sentiments based on defined thresholds, and returns the updated DataFrame with sentiment labels and the original DataFrame for visualization purposes.

Sets the positive and negative thresholds for sentiment classification. Tweets with a sentiment score greater than `positive_threshold = 0.2` are classified as "positive", and tweets with a sentiment score less than `negative_threshold = -0.2` are classified as "negative". Tweets within the threshold range are classified as "neutral".

1.7 Visualize the analytic results

The `visualize_batch()` function takes a batch DataFrame, performs sentiment analysis on it using the sentiment function, aggregates the sentiment values over time windows, creates a line plot of the average sentiment values, and saves the plot as an image file. This function can be used iteratively for each batch of data to visualize and save sentiment analysis results over time.

Print out the average sentiment tweets over a period (in this case is 1 minute)

Results after running:

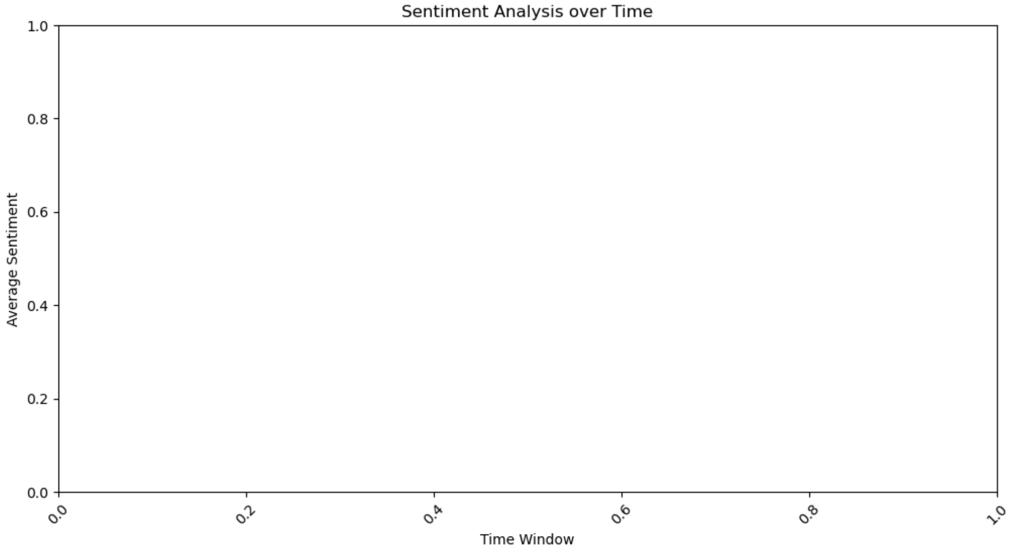


Figure 1.4: Batch 0

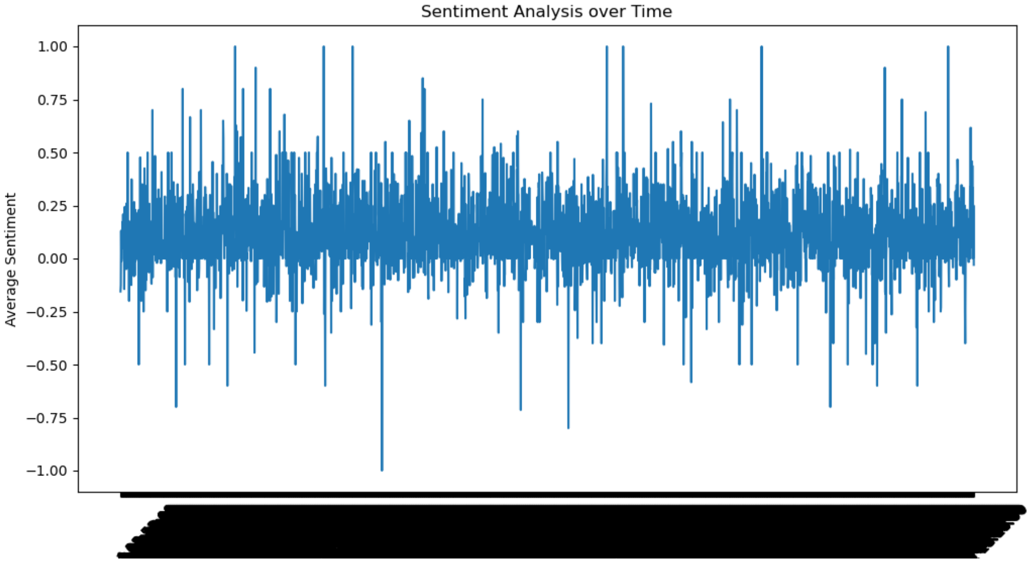


Figure 1.5: Batch 1

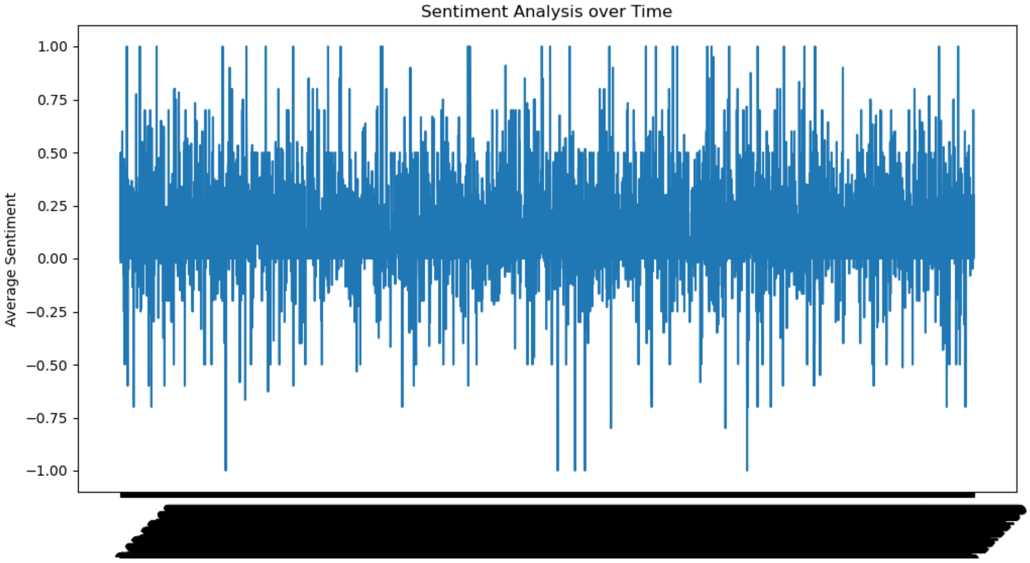


Figure 1.6: Batch 2

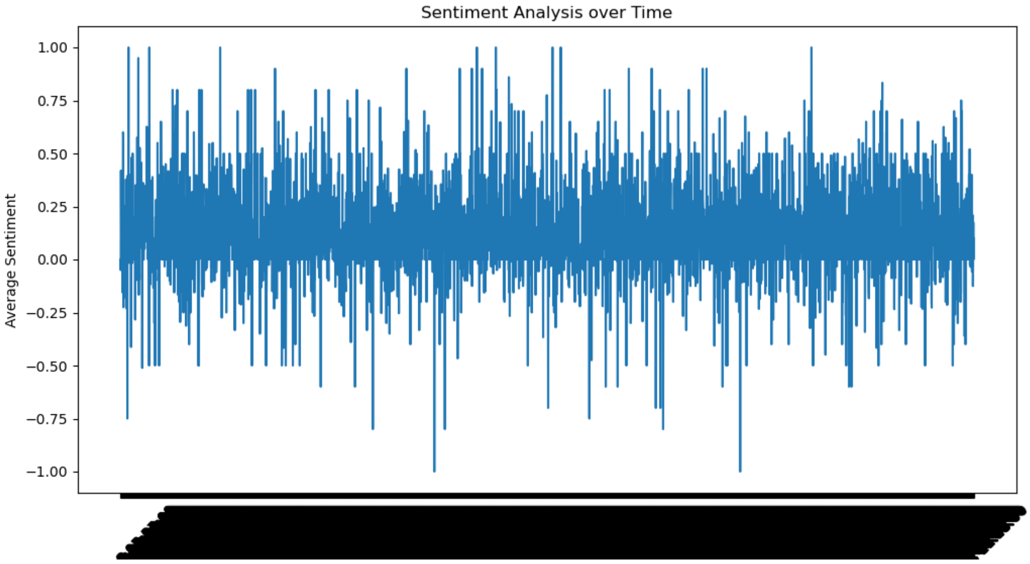


Figure 1.7: Batch 3

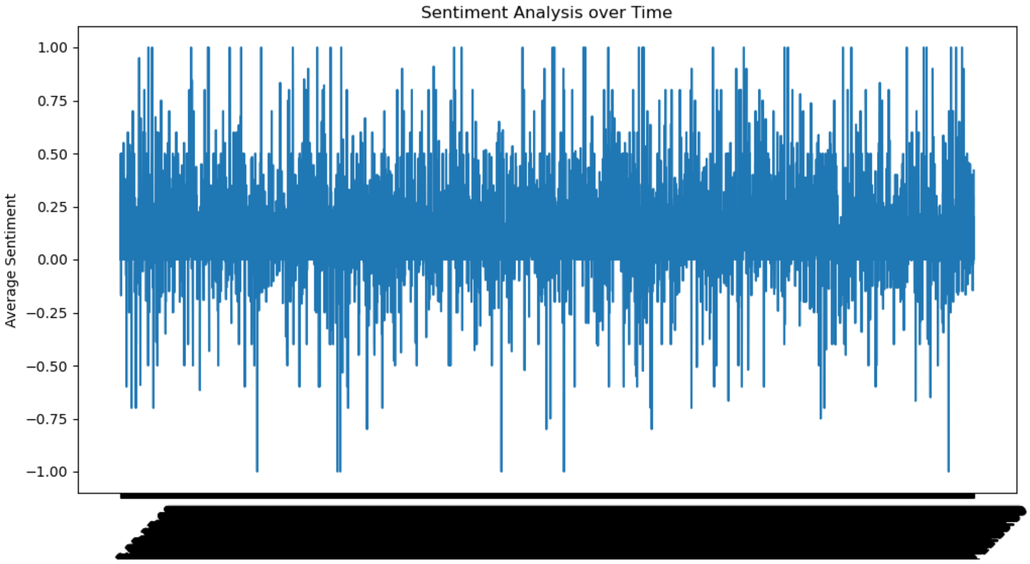


Figure 1.8: Batch 4

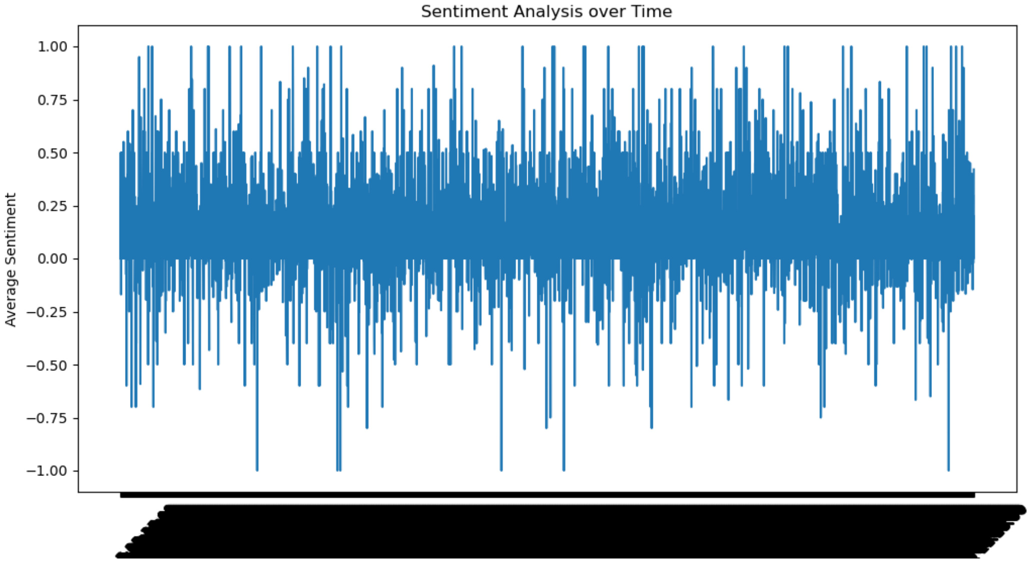


Figure 1.9: Batch 5

1.8 References

- https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html
- https://kafka.apache.org/documentation/
- https://sparkbyexamples.com/spark/spark-streaming-with-kafka/?utm_content=expand_article
- https://matplotlib.org/stable/gallery/index.html