الكلية متعددة التخصصات تارودانت
ⵜⴰⵢⵡⵍⵉⵜ ⵜⴰⵣⵜⵣⵡⵔⵜ ⵏ ⵜⵙⵏⴰⵡⵉⵜ
FACULTE POLYDISCIPLINAIRE TAROUDANT

MASTER BIG DATA & ARTIFICIAL INTELLIGENCE

CHALLENGE 1: SENTIMENT ANALYSIS OVER MOROCCAN POLITICAL COMMENTS ON HESPRESS

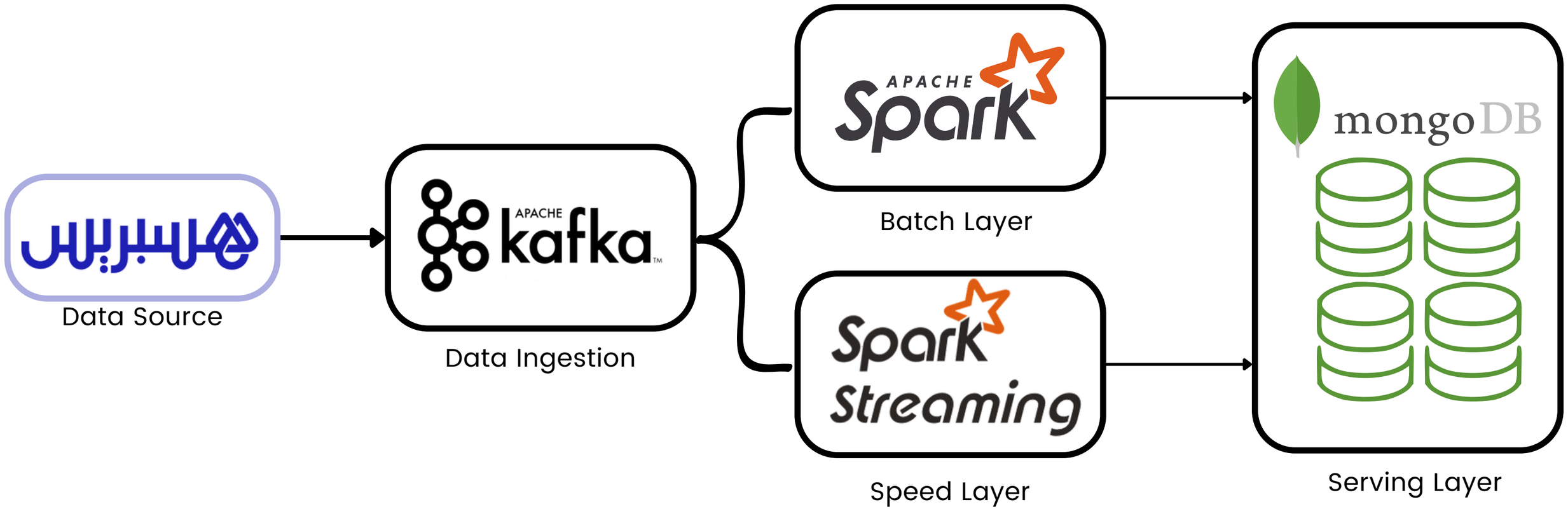
Presented by:
BANANI MOHAMED AMINE

Supervised by:
El HAJJI MOHAMED

1. Architecture
2. Scraping comments from hespress.com
3. Data Ingestion
4. Batch Processing
5. Sentiment Analysis
6. Building Real-Time ETL Pipelines with Apache Kafka
7. Real Time Dashboarding

ARCHITECTURE

Lambda Architecture



SCRAPING COMMENTS FROM HESPRESS WEBSITE USING BEAUTIFULSOUP

Installing Necessary Packages

```
pip install beautifulsoup4
```

```
pip install requests
```

Scraping Comments

```
import requests
from bs4 import BeautifulSoup

# Define the URL to scrape
url = "https://www.hespress.com/-نواب-البام-يقترحون-معاقبة-محتلي-الأر
.1117725.html"

# Make a GET request to the website
response = requests.get(url)

# Parse the HTML using BeautifulSoup
soup = BeautifulSoup(response.text, "html.parser")

# comments = soup.find_all("div", class_="comment-text")
comments = soup.find_all("div", class_="comments")

for comment in comments:
    print(comment.text)
```

Storing Comments in a json File

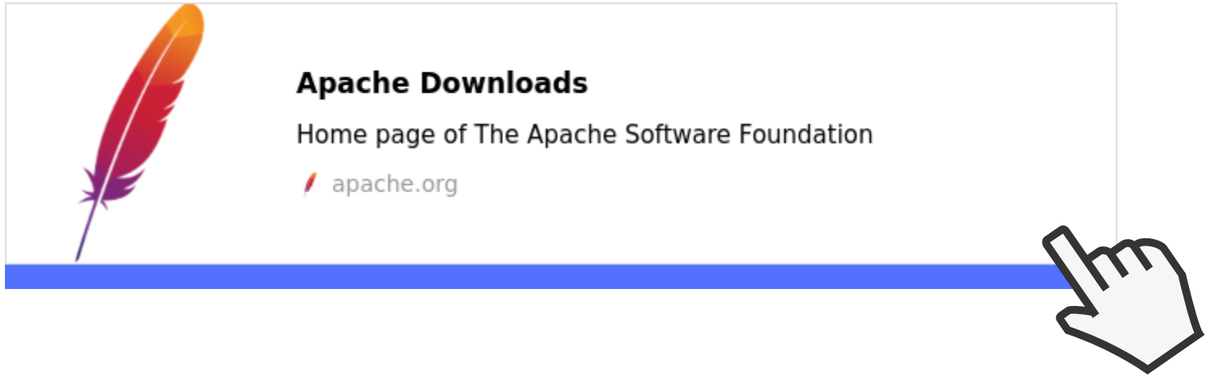
```
comments_to_insert = []
# Iterate through the comments and extract the text of the two div elements
for comment in comments:
    for ul in comment.find_all('ul'):
        for li in ul.find_all('li'):
            div_container = li.find('div', {'class': 'comment-body'})
            div1 = div_container.find('div', {'class': 'comment-head'})
            div2 = div_container.find('div', {'class': 'comment-text'})
            comment_obj = {"author": div1.text.split("\n")[2],
             "date" :div1.text.split("\n")[4] ,
             "text": div2.text
                          }
            comments_to_insert.append(comment_obj)

print(comments_to_insert)

# insert all comments
comments_collection.insert_many(comments_to_insert)

import json

with open("comments.json", "w") as file:
    json.dump(comments_to_insert, file, indent=4)
```

OUTPUT

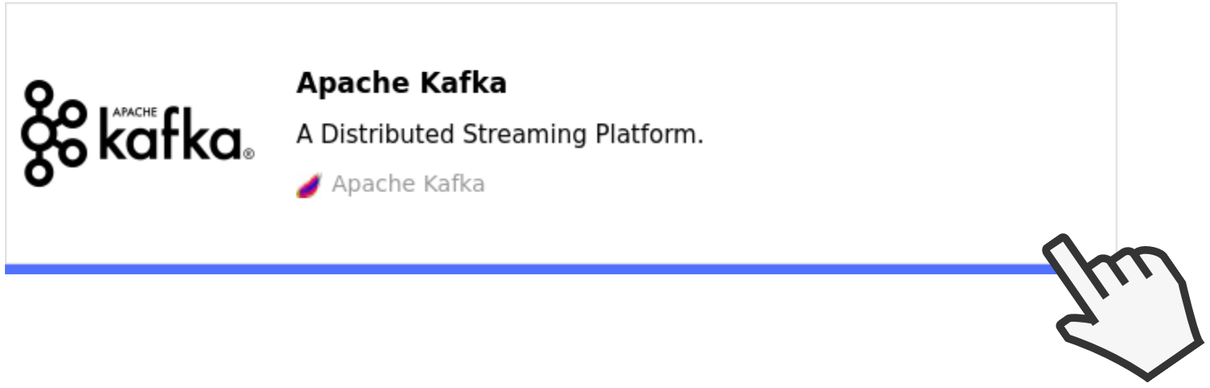

```
{
        'author': 'نورالدين ',
        'date': 'السبت 4 فبراير 2023 - 12:04 ',
        'text': 'عاقبوا غير أمينكم العام عاد أجيو شوفو الباقي...اللهم اضرب الظالمين بالظالمين و أخرجنا من بينهم سالمين\n'
},

{
        'author': 'bidaoui ',
        'date': 'السبت 4 فبراير 2023 - 12:06 ',
        'text': 'الفراشة سرطان صامت يقتل المجتمع ببطء\n'
},
.............
```

DATA INGESTION

Install Apache Zookeeper



Install Apache KAFKA



Run zookeper server :

```
.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
```

```
[2023-02-10 01:18:57,504] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-10 01:18:57,504] INFO   ______                   _                                                (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-10 01:18:57,504] INFO  |___  /                  | |                                               (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-10 01:18:57,505] INFO     / /    ___     ___    | | __   ___    ___   _ __     ___   _ __        (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-10 01:18:57,505] INFO    / /    / _ \   / _ \   | |/ /  / _ \  / _ \ | '_ \   / _ \ | '__| (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-10 01:18:57,505] INFO   / /__  | (_) | | (_) |  |   <  |  __/ |  __/ | |_) | |  __/ | |     (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-10 01:18:57,505] INFO  /_____|  \___/   \___/   |_|\_\  \___|  \___| | .__/   \___| |_| (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-10 01:18:57,506] INFO                                                | |                    (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-10 01:18:57,506] INFO                                                |_|                    (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-10 01:18:57,506] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-10 01:18:57,513] INFO Server environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, built on 04/08/2021 16:35 GMT
(org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-10 01:18:57,513] INFO Server environment:host.name=host.docker.internal (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-10 01:18:57,514] INFO Server environment:java.version=1.8.0_351 (org.apache.zookeeper.server.ZooKeeperServer)
```

Run KAFKA server:

```
.\bin\windows\kafka-server-start.bat .\config\server.properties
```

Create a topic called " comments " :

```
kafka-topics.bat --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic comments
```

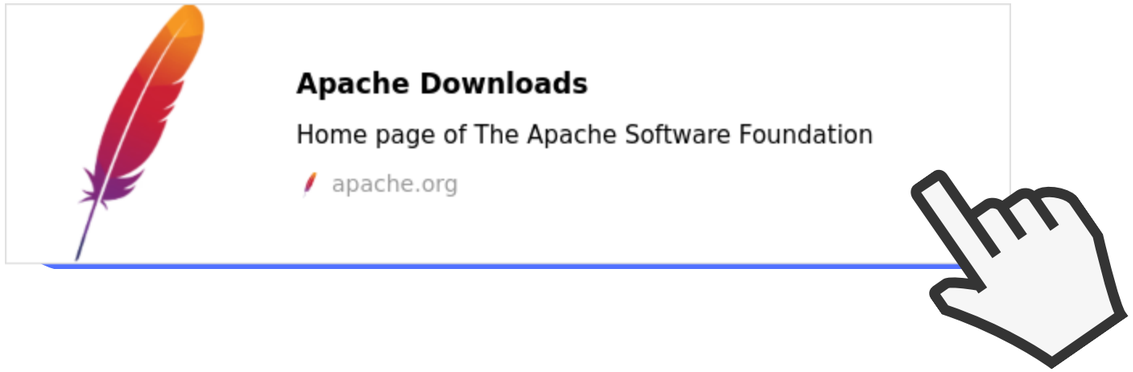
- --create flag indicates that you want to create a new topic.
- --zookeeper flag specifies the ZooKeeper ensemble that Apache Kafka should use to coordinate the brokers in the cluster.
- --replication-factor flag specifies the number of replicas of each partition that should be stored in the cluster. In this case, we are setting the replication factor to 1, meaning that there will be one copy of each partition stored in the cluster.
- --partitions flag specifies the number of partitions that the topic should have. In this case, we are setting the number of partitions to 1.
- --topic flag specifies the name of the topic, which is "comments" in this case.

Creating a Producer :

```
from kafka import KafkaProducer
import pandas as pd
import json
# Set up the Kafka producer
producer = KafkaProducer(bootstrap_servers='localhost:9092')

comments = pd.read_json('comments_analysis.json')

# Iterate through the comments and send each one to the Kafka topic
comments_to_ingest = []
# Iterate over the rows of the dataframe
for index, row in comments.iterrows():
    comment_obj = {"author": row['author'], "date" :row['date'] , "text": row['text'], "sentiment":row['sentiment']}
    comments_to_ingest.append(comment_obj)

for comment in comments_to_ingest:
    # Serialize the comment object to a JSON string
    comment_json = json.dumps(comment)
    # Send the comment to the Kafka topic
    producer.send('comments', key=None, value=comment_json.encode())

# Wait for any outstanding messages to be delivered and delivery report confirmations to be received
producer.flush()
```

BATCH PROCESSING

Download and Install Apache Spark



Starting spark using :

```
Spark-shell
```

Download and Install Apache Spark

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
23/02/10 00:10:40 WARN NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://host.docker.internal:4040
Spark context available as 'sc' (master = local[*], app id = local-
1675984243531).
Spark session available as 'spark'.
Welcome to

      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.3.1
      /_/

Using Scala version 2.12.15 (Java HotSpot(TM) 64-Bit Server VM, Java
1.8.0_351)
```

Install PySpark and findspark

```
pip install pyspark
```

```
pip install findspark
```

Processing Comments

```
import findspark
findspark.init()
from pyspark.sql import SparkSession

# Create Spark session
spark = SparkSession.builder.appName("BatchProcessing").getOrCreate()

# Load data from the local JSON file into a DataFrame
df = spark.read.option("multiline","true").json(r"comments.json")

# Perform batch processing to count the number of comments per author
result = df.groupBy("author").count().collect()

# Show result
for row in result:
    print(f"Author: {row[0]}\nNumber of comments: {row[1]}\n")

# Stop Spark session
spark.stop()
```

Output

```
Author:  مواطن
Number of comments: 4

Author:  كمال بيه
Number of comments: 1

Author:  الاحتلال أنواع
Number of comments: 1

Author: ahmed
Number of comments: 4

Author:  ابوجهل
Number of comments: 1

Author:  احمد
Number of comments: 1
```

```
Author:  خريبگي
Number of comments: 1

Author:  عكاشة
Number of comments: 1

Author:  العريشي
Number of comments: 1

Author:  نواب اخر زمان
Number of comments: 1

Author:  مواطن معاق
Number of comments: 1

Author: Me again
Number of comments: 1
```

```
Author:  عبد الوافي.
Number of comments: 1

Author:  السيّارين
 والسيّارات..
Number of comments: 1
```

Processing Comments

```
# Create an RDD from the "text" column
text_rdd = df.rdd.flatMap(lambda x: re.split("\W+", x.text))

# Count the frequency of each word
word_counts = text_rdd.map(lambda word: (word, 1)).reduceByKey(add)

# Sort the words by frequency
sorted_word_counts = word_counts.sortBy(lambda x: x[1], ascending=False)

# Show the top 10 most used words
top_10 = sorted_word_counts.take(10)
for word, count in top_10:
    print(f"{word}: {count}")
```

Output

```
122 :
80 :في
73 :و
71 :من
41 :على
36 :لا
23 :ان
22 :او
21 :هو
20 :أن
```

SENTIMENT ANALYSIS

Sentiment Analysis

https://www.kaggle.com/datasets/tariqmassaoudi/hespress

author	date	text	sentiment
نورالدين	السبت 4 فبراير 2023 - 12:04	عاقبوا غير أمينكم العام عاد أجيو شوفو الباقي...اللهم اضرب الظالمين بالظالمين و أخرجنا من بينهم سالمين	positive
bidaoui	السبت 4 فبراير 2023 - 12:06	الفراشة سرطان صامت يقتل المجتمع ببطء	positive
Kamal	السبت 4 فبراير 2023 - 12:06	هذا يعطي شرعية لاحتلال الملك العمومي، لتفادي هذا النوع من الحوادث يجب إيقاف كل أشكال احتلال الملك العمومي و التعبئة من أجل هذا الهدف، و ليس الإعتراف بالهزيمة بمثل هاته النصوص.	negative

Installing the Necessary Packages

```
pip install pandas
```

```
pip install sklearn
```

```
# Load the existing dataset into a pandas dataframe
df = pd.read_csv("comments_politique.csv")

# Select a smaller subset of the data
subset = df.sample(frac=0.1)

# Preprocess the text data
vectorizer = CountVectorizer()
features = vectorizer.fit_transform(subset['comment'])

# Train the model
model = DecisionTreeClassifier()
model.fit(features, subset['score'])

# Load the scraped data into a pandas dataframe
scraped_data = pd.read_json("./../Data/comments.json")

# Extract the text field from the scraped data
scraped_comments = scraped_data['text'].tolist()

# Preprocess the scraped comments
scraped_features = vectorizer.transform(scraped_comments)

# Make predictions on the scraped comments
scraped_scores = model.predict(scraped_features)
print(scraped_scores)
```

#Output of predicted sco

```
[ 1  2 -17 131  22  52  27  18   3   2  48  17  14   0 -13   1 -17   4
  30 305  14   8  24 -13   0  30 136   2  27   6  11  18   3  13   3   0
   4   2 -22   0  -3   6  13   2 136  69   3   7  12   7   5   0   2  35
 -24   6   1   2   3   7   1]
```

BUILDING REAL-TIME ETL PIPLINE WITH APACHE KAFKA AND APACHE SPARK

SBT (Scala Build Tool)

sbt is an open-source build tool for Scala and Java projects, similar to Apache's Maven and Gradle. Its main features are: Native support for compiling Scala code and integrating with many Scala test frameworks. Continuous compilation, testing, and deployment.

Loading Dependacies using SBT

```
scalaVersion := "2.12.15"
sparkVersion = "3.3.1"
```

```
libraryDependencies += "org.apache.spark" %% "spark-core" % sparkVersion
libraryDependencies += "org.apache.spark" %% "spark-sql-kafka-0-10" % sparkVersion
libraryDependencies += "org.apache.spark" %% "spark-streaming" % sparkVersion
libraryDependencies += "org.apache.kafka" %% "kafka" % "3.3.2"
libraryDependencies += "org.mongodb.spark" %% "mongo-spark-connector" % "10.1.1"
libraryDependencies += "org.apache.spark" %% "spark-sql" % sparkVersion
libraryDependencies += "org.apache.spark" % "spark-streaming-kafka-0-10_2.12" % sparkVersion
libraryDependencies += "org.mongodb.scala" %% "mongo-scala-driver" % sparkVersion
```

Import The Necessary Pacakges

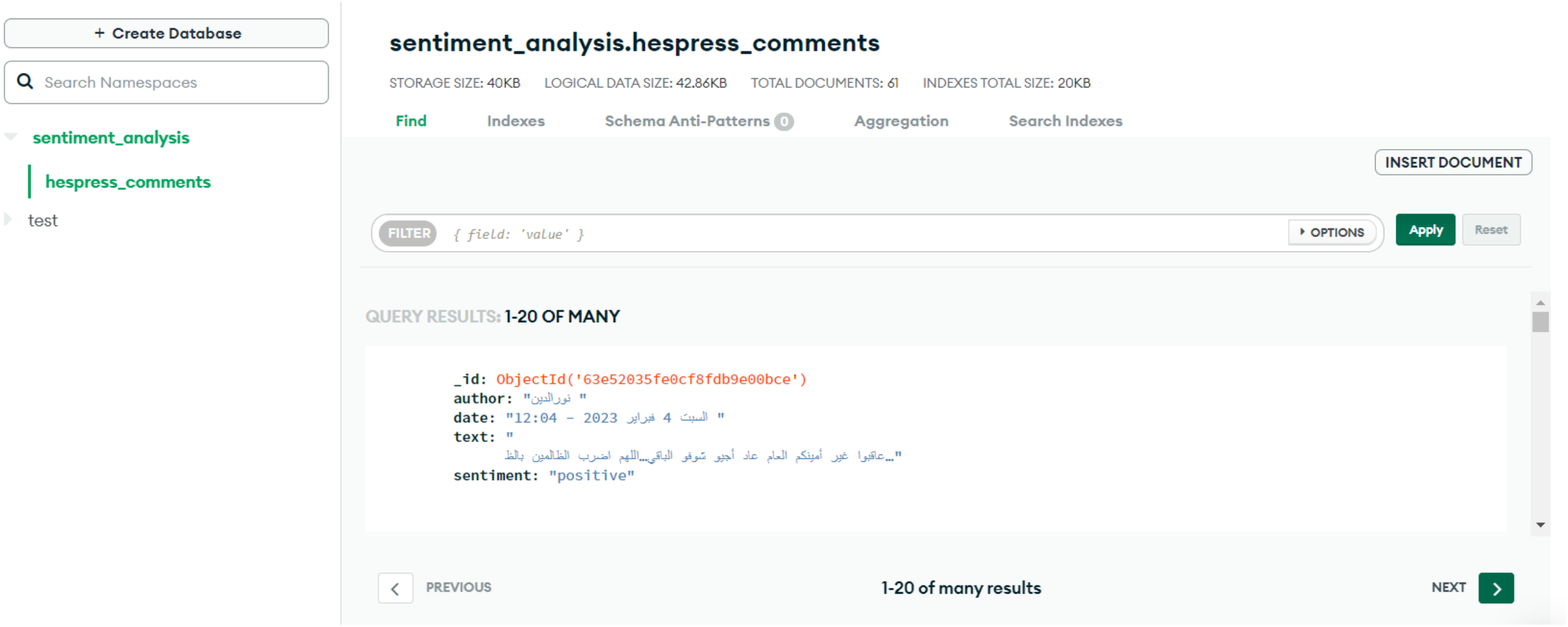
```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._
```

```
// Create Spark session
    val spark =
SparkSession.builder().appName("KafkaToSparkStreaming").master("spark.master").getOrCreate()
```

```
    // Define the schema for the JSON data
    val schema = new StructType()
      .add("author", StringType)
      .add("date", StringType)
      .add("text", StringType)

    // Read data from Kafka
    val df = spark
      .readStream
      .format("kafka")
      .option("kafka.bootstrap.servers", "localhost:9092")
      .option("subscribe", "comments")
      .option("startingOffsets", "earliest")
      .load()
      .select(from_json(col("value").cast("string"), schema).as("data"))
      .select("data.*")
```

```
// Write the data frame to MongoDB
    df.write
      .format("com.mongodb.spark.sql.DefaultSource")
      .option("uri", "mongodb://localhost:27017/db_name.collection_name")
      .mode("overwrite")
      .save()

    // Stop the Spark session
    spark.stop()
```

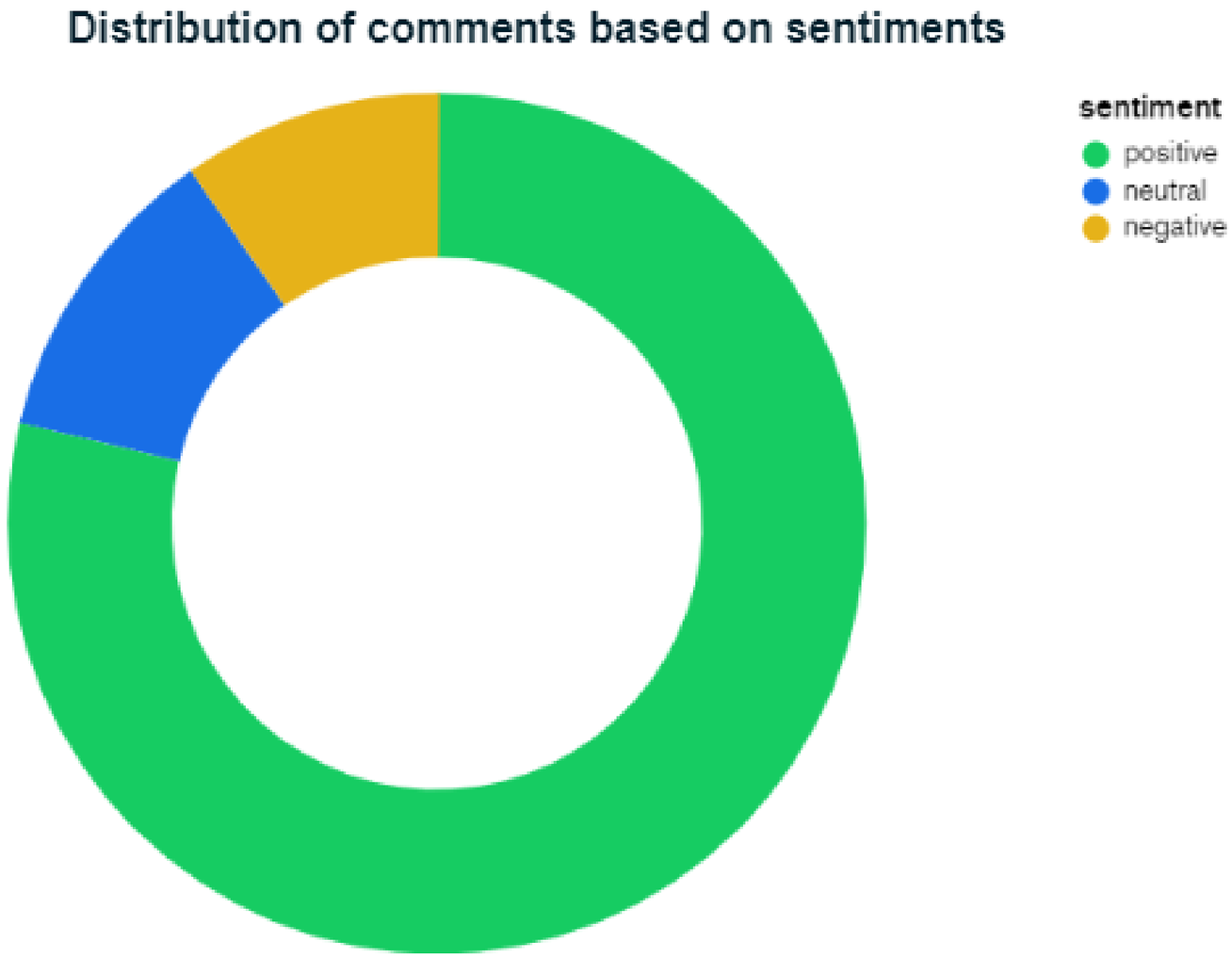
the comments has been successfully streamed to mongo db

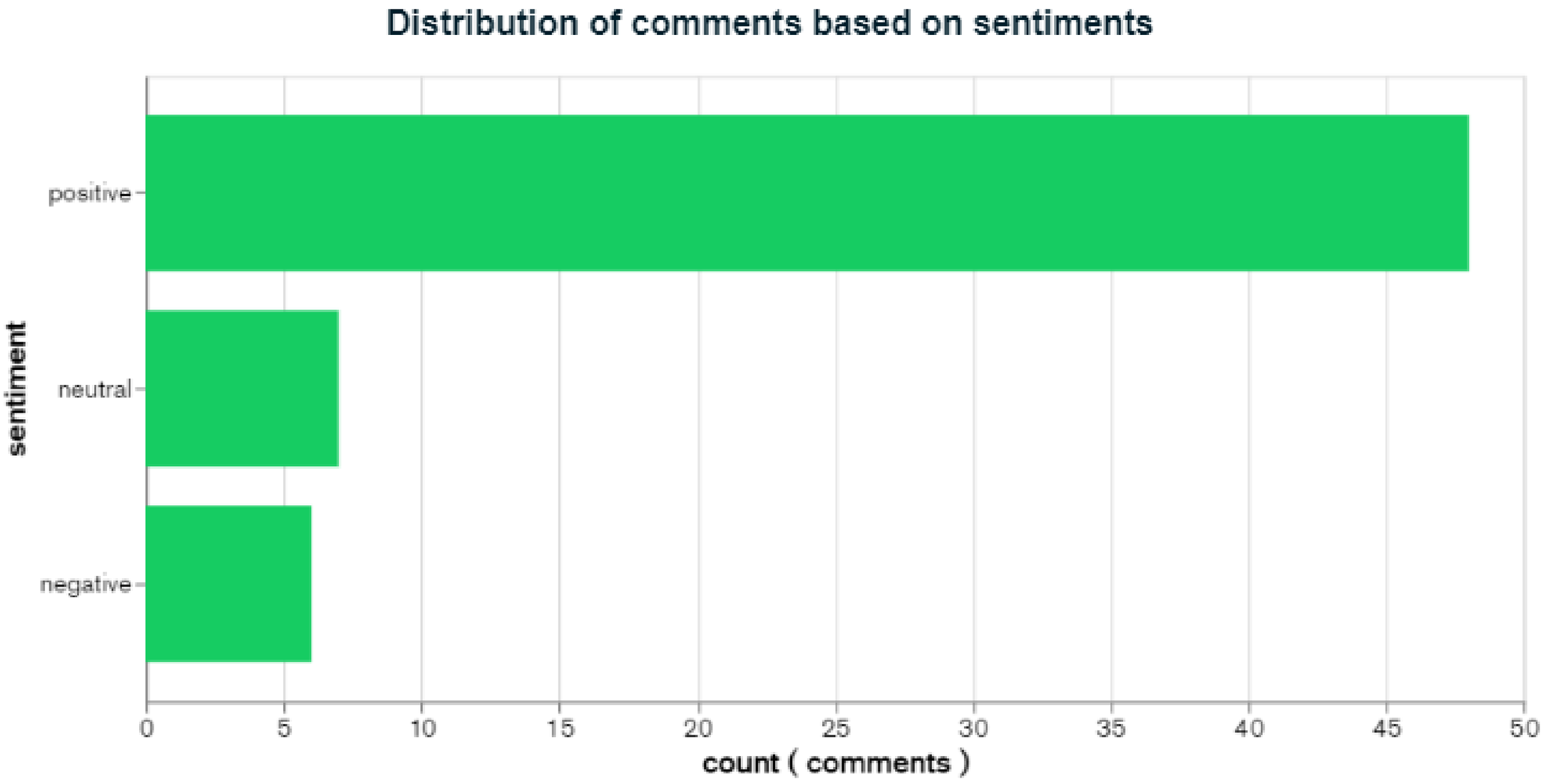


REAL-TIME DASHBORDING

Real time dashboarding in Mongo DB

10 seconds refresh




Distribution of comments based on sentiments
positive
neutral
negative
sentiment
0
5
10
15
20
25
30
35
40
45
50
count (comments)

github repo :

https://github.com/mamen15/Hespress_Comments_Sentiment_Analysis

Dependencies :

https://mvnrepository.com/

THANK YOU FOR YOUR ATTENTION