PES University

Department of Computer Science and Engineering

UE22CS343BB3 - Database Technologies

Jan-May 2025

Project Guide

1. Data Collection: Twitter Streaming or Kaggle Dataset

You have two options for collecting the data:

- **Option 1: Stream Twitter Data**

  Use **Twitter API** to fetch real-time tweets and publish them to **Kafka topics**.

  Steps:

  - **Set up a Twitter Developer account** and get your credentials (API keys).
  - Use **Tweepy** (Python library) to connect to the Twitter API and listen to tweets in real-time.
  - Once you fetch the tweets, publish them to **Kafka** using a Kafka Producer.
  - Use **Kafka Topics** to organize the data (e.g., `tweets_topic`, `user_data_topic`).

- **Option 2: Fetch a Dataset**

  Alternatively, you can download a **pre-collected Twitter dataset** and upload it to Kafka for processing.

  Steps:

  - Download a **Twitter dataset** (search for datasets related to Twitter, such as tweets or user data).
  - Use **KafkaProducer** to upload this dataset to Kafka topics for further processing.

  Eg - kaggle datasets

2. Kafka Setup & Data Streaming

- **Set up Kafka:**
  First, you need to install **Kafka** and **Zookeeper** (required by Kafka).

- **Kafka Topics**: Create the necessary Kafka topics where you will publish and subscribe to data.
  - Example topics could include `tweets_topic` for tweet data and `user_data_topic` for user-related data.

Resources:

setting up Kafka

- **Kafka Producers**:
  Write a **Producer script** that connects to Kafka and sends data (tweets or user data) to the topics you created.

- **Kafka Consumers**:
  Write a **Consumer script** that reads data from Kafka topics and processes it (e.g., save to a database, analyze tweets).

  example consumer code :

```
from kafka import KafkaConsumer
consumer = KafkaConsumer('tweets_topic',
bootstrap_servers='localhost:9092', group_id='tweet_group')
for message in consumer:
     print(f"Received: {message.value}")
```

3. Processing Data with Spark Streaming

- **Set up Apache Spark Streaming**:
  Use **Apache Spark** to consume data from Kafka in real-time. Spark can help you process data in **micro-batches**. You'll use the Kafka stream to process tweets and perform analysis like counting hashtags, filtering specific content, etc.

  Steps:

  - Install Apache Spark and set it up for streaming.
  - Use **Spark Streaming** to consume data from Kafka.
  - Perform some basic analysis on the data, such as counting specific words or hashtags.

  Resources:

  - spark streaming documentation

4. Storing Processed Data in PostgreSQL

After processing the data using Spark, you will store the results in a **database**.

- **Set up database** (eg - PostgreSQL):

  Install PostgreSQL or MySQL on your machine or use a hosted version. Create a database and a table to store the processed tweet data.

  Steps:

  - Install **PostgreSQL/MySQL** and create a database (e.g., `twitter_data`).
  - Create a table to store the tweets, user information, and any processed data.

- **Store Data in DB**:

  Write a **script** that reads data from your Kafka Consumer or Spark output and stores it into the database.

  Resources:

  Example queries for inserting data into PostgreSQL.

```
import psycopg2

# Connect to PostgreSQL

conn = psycopg2.connect("dbname=twitter_data user=postgres
password=yourpassword")

cur = conn.cursor()

# Insert a tweet

cur.execute("INSERT INTO tweets (user_name, tweet) VALUES
(%s, %s)", ("Alice", "Hello Kafka!"))

conn.commit()

# Close connection

conn.close()
```

5. Batch Mode Execution (Offline)

- After processing the streaming data, execute **identical queries** on the stored dataset in **batch mode**.
- Compare the performance, speed, and accuracy of the streaming and batch execution.

Steps:

- Use **SQL queries** or **Spark SQL** for processing the stored dataset.

  SELECT tweet, COUNT(*) FROM tweets GROUP BY tweet ORDER BY COUNT(*) DESC LIMIT 10;

- Compare the time taken and resource usage for both streaming and batch modes.

6. Evaluation & Performance Comparison

Once your streaming and batch jobs are running successfully, you need to **evaluate** the system.

- **Measure Performance**:
  Compare the performance of the streaming vs batch execution by tracking metrics like **execution time** and **resource usage**.
- **Accuracy Check**:
  Check if the results from both modes match or if there are discrepancies.