Summary

The term paper's central goal is to gain a comprehensive understanding of Apache Kafka and its real-world applications. It leverages various technologies, including Apache Kafka, PySpark, and MySQL. The primary workflow contains continuous data retrieval from the News API, preprocessing, sentiment analysis, and the seamless storage of the analyzed data.

1. Data Ingestion with Kafka Producer:
   I began the project by running Apache Zookeeper. It maintains essential status information on Zookeeper servers, ensuring seamless coordination among Kafka brokers. Next, I ran Apache Kafka, which is the core messaging system that handles the collection and distribution of data from various sources. With Kafka infrastructure in place, I created a Kafka topic named 'nyt-demo-7'. This topic is the entry point for data ingestion. Then, I set up a Kafka Producer through Python script, which was responsible for fetching real-time news articles related to 'technology' from various sources through the News API. This producer was configured to communicate with the Kafka cluster, specifically the 'localhost:9092' Kafka broker, to send the retrieved articles as JSON objects to the Kafka topic 'nyt-demo-7.'

2. Data Consumption with Kafka Consumer:
   To consume the data ingested by the producer, a Kafka consumer was employed. The consumer was set to subscribe to the 'nyt-demo-7.' topic. This consumer, through Python script, received the articles from Kafka and printed the titles of the received articles as proof of successful data ingestion.

3. Data Preprocessing and Sentiment Analysis:
   I utilized PySpark streaming capabilities to perform sentiment analysis. It was used to retrieve data from the Kafka topic and apply data preprocessing and sentiment analysis. A schema was defined to structure the incoming JSON data. By selecting relevant fields from the articles, the data was prepared for analysis. The process of text cleaning included steps such as converting text to lowercase and removing non-alphabetic characters, numbers, and punctuations. These preprocessing steps ensured that the data was in a suitable format for sentiment analysis. Sentiment analysis was carried out through a user-defined function (UDF) employing the Textblob library to calculate the polarity of the article's content. The polarity score determined whether an article's sentiment was negative, positive, or neutral. Another UDF was employed to classify the sentiment based on polarity scores, providing a comprehensive understanding of the article's sentiment.

4. Data Storage in MySQL:
   The PySpark script was configured to save the cleaned and analyzed data into a MySQL database named 'nyt_demo.' In the PySpark script, I made use of the JDBC connector to write the processed news articles to the MySQL database. This step ensures that the valuable insights gained from sentiment analysis are retained for future analysis or retrieval, marking the end of the data pipeline.

5. Visualization using Tableau
   I leveraged Tableau to create insightful visualizations. With roughly 200 negative articles, 450 neutral articles, and 325 positive pieces, the first visualization showed the sentiment distribution. The second visualization delves deeper into the distribution of sentiment, with

'Technology' standing out as the most prevalent category across all sentiment types. The third visualization, based on news desks, revealed the 'Business' news desk as the dominant category across all sentiment categories, providing valuable insights into the dataset's composition.