Real-Time Data Streaming and NLP Analysis using Kafka, Transformers, and BERTopic

Jean Paul Saba
Caren Dib

January 21, 2025

Abstract

This report details the implementation of a real-time data streaming pipeline leveraging Apache Kafka. The pipeline integrates NLP techniques such as sentiment analysis using Transformers pipeline and topic modeling using BERTopic. It also includes producer-consumer mechanisms and demonstrates training and utilization of a BERTopic model. The goal is to process Reddit posts dynamically, categorize them, and extract insights in real-time.

For more information, visit the GitHub repository: `https://github.com/JPTheKnight/Real-Time-Data-Streaming-using-Kafka`

Contents

1 Introduction

In the era of big data, real-time processing of information is crucial. Platforms like Reddit generate vast amounts of data, presenting opportunities for analyzing public opinion, extracting trends, and categorizing content dynamically.

This project leverages Apache Kafka, a distributed event-streaming platform, to build a pipeline for real-time processing of Reddit posts. We integrate advanced NLP techniques, including:

- Sentiment analysis to gauge public sentiment using Transformers pipeline on a sentiment analysis task.
- Getting data from the stream to train the BERTopic model.
- Topic modeling using BERTopic to identify themes.
- Real-time categorization of Reddit posts.

The pipeline is implemented with Python and demonstrates key functionalities, including producer-consumer architecture, sentiment analysis, topic analysis, and model training.

2 Architecture Overview

The system can follow two architectural versions for real-time data streaming and analysis. Below, we outline the key differences between Version 1 and Version 2.

2.1 Version 1: Single Topic with Partitioned Consumers

In the first version, the architecture consists of a single Kafka topic, **reddit_stream**, which is divided into two partitions. The system components are structured as follows:

- **Producer**: A single Reddit streamer fetches posts and sends them to the `reddit_stream` topic.
- **Consumers for Sentiment Analysis**: A single sentiment analysis consumer script is executed twice, forming two consumers within the same group, **reddit-sentiment-group**. Each consumer reads from one of the two partitions.

- **Consumers for Topic Extraction**: A single topic extraction consumer script is also executed twice, forming two consumers within the **reddit-topic-group**. Each consumer reads from a separate partition of the `reddit_stream` topic.
- **Model Training**: The BERTopic model is trained separately and used for topic extraction.

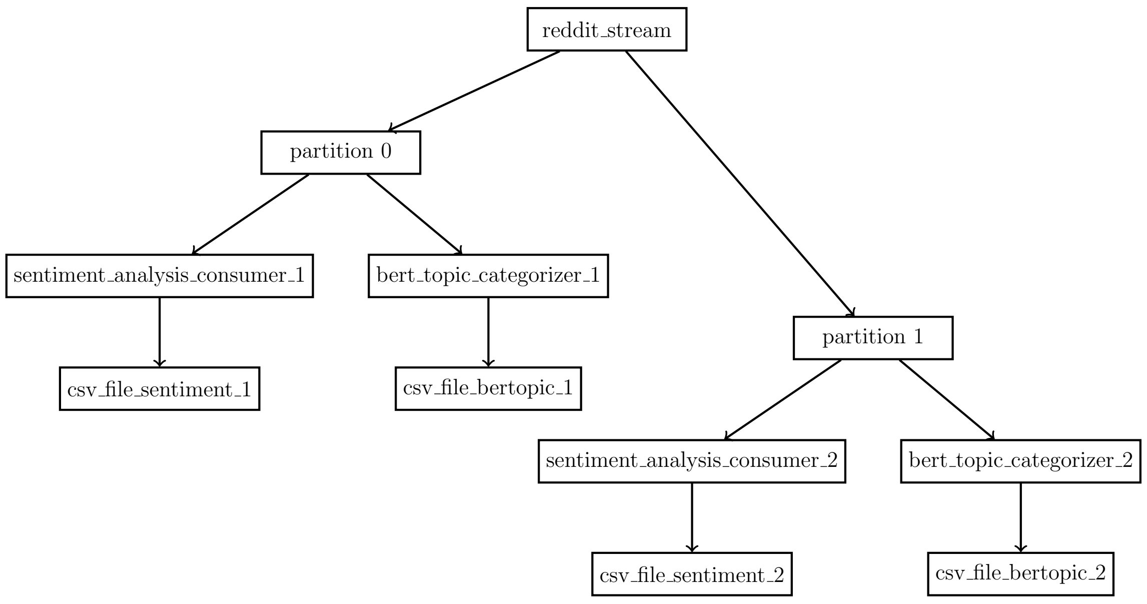
Pros

- **Efficient Partitioning**: By dividing the topic into two partitions, consumers can process data in parallel, improving throughput and performance.
- **Reduced Overhead**: Only one Kafka topic needs to be managed, reducing resource usage.
- **Direct Processing**: Data flows directly from the producer to the consumers, minimizing latency.

Cons

- **Coupling Between Components**: Sentiment analysis and topic extraction are tightly coupled to the same data source, limiting flexibility.
- **Single Pipeline Dependency**: Any issue in the `reddit_stream` topic affects all consumers, potentially disrupting both sentiment analysis and topic extraction simultaneously.
- **Difficult Monitoring**: With both sentiment analysis and topic extraction relying on the same topic, it can be challenging to monitor and debug issues, as all logs and metrics will overlap, making root cause analysis harder.

In this version, the sentiment analysis and topic extraction consumers process data directly from the `reddit_stream` topic. Each consumer in a group reads from one partition, ensuring load balancing and efficient message processing.



2.2 Version 2 (Implemented): Multi-Topic Pipeline with Sentiment-Analyzed Data

The second version introduces an additional Kafka topic, **sentiment_analyzed**, to improve modularity and processing flow. The architecture components are as follows:

- **Producer**: A single Reddit streamer fetches posts and sends them to the `reddit_stream` topic.
- **Consumers for Sentiment Analysis (Also Acting as Producers)**: Two sentiment analysis consumers (within the **reddit-sentiment-group**) read messages from the `reddit_stream` topic, analyze their sentiment, and publish the processed data to a new topic, `sentiment_analyzed`.
- **Consumers for Topic Extraction**: Two topic extraction consumers (within the **reddit-topic-group**) read from the `sentiment_analyzed` topic instead of directly from `reddit_stream`. This ensures that topic extraction occurs only after sentiment analysis is complete.
- **Model Training**: The BERTopic model is trained separately and used for topic extraction.

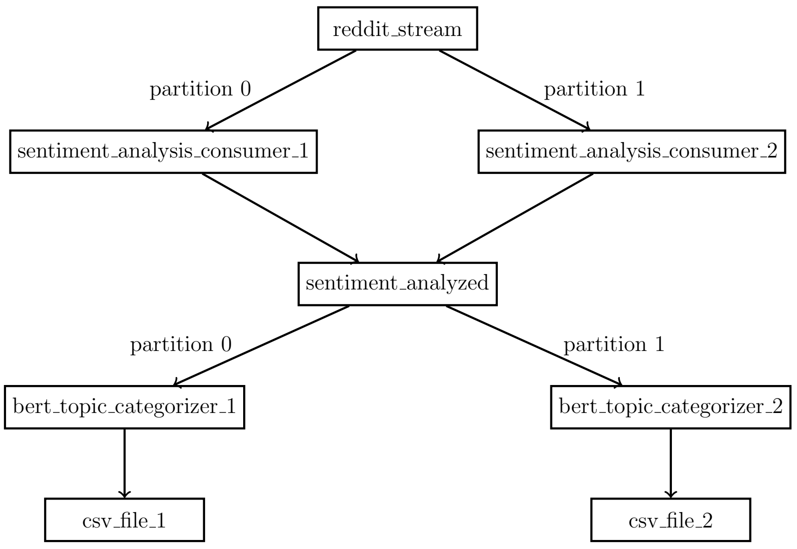
Pros

- **Modularity**: The introduction of `sentiment_analyzed` as an intermediary topic decouples sentiment analysis from topic extraction, enabling more flexibility.
- **Scalability**: Different stages of the pipeline (sentiment analysis and topic extraction) can scale independently based on workload.
- **Data Reusability**: Sentiment-analyzed data can be reused for additional downstream applications without reprocessing the original data.
- **Fault Isolation**: Errors in sentiment analysis or topic extraction do not affect other components, improving fault tolerance.

Cons

- **Increased Complexity**: Managing multiple topics and producer-consumer chains adds complexity to the architecture.
- **Higher Resource Usage**: Additional topics and intermediate storage require more Kafka resources.

This improved version introduces an intermediary topic, `sentiment_analyzed`, ensuring that topic extraction only processes sentiment-enriched data. By separating concerns, the pipeline remains more modular and allows for independent scaling of sentiment analysis and topic extraction components. This is the version that is used in the project.



3 Topic Creation and Partition Assignment

To handle streaming data effectively and ensure parallelism, we created two Kafka topics named **reddit_stream** and **sentiment_analyzed**. These topics were configured with **two partitions** each, allowing for balanced message distribution and parallel consumption by multiple consumers.

3.1 Topic Creation

The topics `reddit_stream` and `sentiment_analyzed` were created using Kafka's command-line tools with the following configuration:

- **Number of Partitions**: 2
- **Replication Factor**: 1 (suitable for development environments)

The topics were created using the following command:

```
.\bin\windows\kafka-topics.bat --create \
    --topic {reddit_stream / sentiment_analyzed} \
    --bootstrap-server localhost:9092 \
    --partitions 2 \
    --replication-factor 1
```

3.2 Partition Usage

Partitions in Kafka allow for parallel processing of data. For the topics:

- **Partition 0** and **Partition 1** divide the messages sent by the producer.
- Messages are assigned to partitions based on a kafka strategy which is round robin (modulo the number of partitions). So the producer will send the first message to the partition 0, then the second message to partition 1, then the third to partition 0...

3.3 Consumers' Role

Two consumers were set up to read messages from each of the topics:

- **Consumer 1**: Reads messages from **Partition 0**.
- **Consumer 2**: Reads messages from **Partition 1**.

Both consumers belong to the same consumer group. Kafka ensures that each partition is consumed by only one consumer within the group, enabling load balancing and efficient parallel processing.

3.4 Benefits of Partitioning

- **Parallelism**: Each consumer processes messages independently, improving overall performance.
- **Scalability**: Adding more partitions allows for increased consumer count and greater throughput.
- **Efficient Load Distribution**: Messages are distributed evenly across partitions, ensuring balanced workload among consumers.

This setup ensures a robust and scalable data streaming pipeline, optimized for real-time processing of Reddit posts.

4 Reddit Streamer Implementation

The Reddit streamer streams Reddit posts to the `reddit_stream` topic. The `reddit_streamer.py` Python script fetches posts using the Reddit API and sends them to Kafka:

4.1 Importing Libraries and Defining Functions

This section introduces the essential libraries and defines helper functions for connecting to Reddit and streaming posts. Key components include:

- **Library Imports**: The required libraries include `kafka`, `praw`, and `json` for Kafka integration, Reddit API access, and JSON serialization.
- **Reddit Streaming Function**: The `reddit_stream` function connects to Reddit using credentials and streams new posts from a specified subreddit. This function outputs a dictionary containing relevant details about each post, including title, author, URL, and score. This ensures structured data for further processing.

Key Features:

- Establishes a connection to Reddit using PRAW.
- Streams posts in real-time using the `subreddit.stream.submissions()` method.
- Prepares structured data for Kafka producers.

4.2 Kafka Producer Creation

This section focuses on creating a Kafka producer and defining how messages are sent to Kafka topics. The primary components include:

- **Kafka Producer Function**: The `kafka_producer` function initializes a Kafka producer connected to the specified broker. It ensures that the data is serialized into JSON format for compatibility with Kafka topics.
- **Message Publishing**: The `publish_to_kafka` function sends messages to Kafka topics. It takes the topic name and message as arguments and flushes the producer buffer after each send to ensure real-time delivery.

Key Features:

- Ensures JSON serialization of messages for compatibility.
- Supports partitioning of messages.
- Enables reliable message delivery with buffer flushing.

4.3 Main Execution

The main execution block orchestrates the entire process of streaming Reddit posts and publishing them to Kafka. Key operations include:

- **Initialization**: Initializes Reddit credentials and Kafka producer settings, ensuring all dependencies are correctly configured.
- **Subreddit Iteration**: Iterates through a predefined list of subreddits. For each subreddit, posts are streamed and processed.
- **Post Limit**: Ensures a maximum of 90 posts per subreddit are processed to be able to get data from multiple subreddits with different topics.
- **Error Handling**: Includes error handling for graceful shutdown on keyboard interruption or unexpected errors.

Key Features:

- Streams posts sequentially from multiple subreddits.
- Maintains a limit on processed posts per subreddit to control data flow.

4.4 Workflow Summary:

1. Posts are fetched from Reddit in real-time using `reddit_stream`.
2. Posts are published to Kafka with partitioning logic.
3. Execution stops after reaching the post limit or receiving a keyboard interrupt.

5 Sentiment Analysis Consumers

The sentiment analysis consumer is responsible for reading messages from the `reddit_stream` topic, performing sentiment analysis on the titles of Reddit posts, and then sending all the analyzed info and the title to the `sentiment_analyzed` topic.

5.1 Initialization and Configuration

The consumer initializes by connecting to the Kafka broker and subscribing to the `reddit_stream` topic.

- **Kafka Configuration**: The consumer connects to the Kafka broker using the `bootstrap_servers` parameter and listens to messages starting from the earliest offset.
- **Sentiment Analysis Pipeline**: A Hugging Face pre-trained sentiment analysis model `(cardiffnlp/twitter-roberta-base-sentiment)` is initialized using the `pipeline` API. This model was trained using social media data.

5.2 Message Processing Workflow

The consumer follows these steps to process messages:

1. Reads a message from the assigned partition of the `reddit_stream` topic.
2. Extracts the `title` field from the message payload.
3. Cleans the title using a custom text cleaning function to remove unwanted links, and extra spaces.
4. Performs sentiment analysis on the cleaned title using the Hugging Face pipeline.

5. Logs the results, including the partition from which the message was read.
6. Sends the result to the `sentiment_analyzed` topic.

5.3 Error Handling and Termination

The consumer is designed to handle errors gracefully:

- **Keyboard Interrupt**: The consumer can be stopped manually, ensuring all processed messages are saved before termination.
- **Exception Handling**: Any unexpected errors are logged, allowing for debugging and resolution without disrupting the overall pipeline.

5.4 Key Features and Benefits

- **Partition Awareness**: The code will be run twice, and each instance will read from a different partition given by kafka.
- **Real-Time Sentiment Analysis**: Sentiments are analyzed in real-time, providing immediate insights into the tone of Reddit posts.

5.5 Example Outputs

Below are example outputs generated by the sentiment analysis consumer:

```
Post: ISRO successfully docks two satellites in space, India fourth
country to achieve feat after US, Russia, China (Partition: 0)
Sentiment: Positive (Confidence: 0.74)

Post: Poppy Playtime Sues Google for Failing to Remove Copyright
Infringing 'Scam' Apps (Partition: 0)
Sentiment: Negative (Confidence: 0.81)

Post: Biden administration launches cybersecurity executive order
(Partition: 0)
Sentiment: Neutral (Confidence: 0.92)
```

5.6 Workflow Summary:

1. Messages are consumed from Kafka in real-time.
2. Sentiment analysis is performed on each message.
3. Processed data is sent to the second topic to continue the streaming pipeline.

This consumer demonstrates an efficient way to integrate real-time data processing with natural language analysis, ensuring scalability and reliability in a distributed environment.

6 Training and Saving the BERTopic Model

The BERTopic model is trained using Reddit post titles collected from the Kafka topic `reddit_stream`. The process involves cleaning and tokenizing text data, training the BERTopic model with dimensionality reduction, and saving the trained model for future use.

6.1 Training the BERTopic Model

To train the BERTopic model, the system follows these steps:

1. **Collecting Titles**: The consumer listens to messages from `reddit_stream` and extracts post titles.
2. **Text Preprocessing**: The titles are cleaned by removing URLs, special characters, and stopwords, ensuring that only relevant words remain.
3. **Dimensionality Reduction**: The UMAP (Uniform Manifold Approximation and Projection) algorithm is used to reduce the dimensionality of embeddings while maintaining meaningful clustering.
4. **Topic Modeling**: BERTopic is applied to the processed titles, identifying patterns and grouping similar posts into topics.

This approach ensures that topics are extracted efficiently, allowing for meaningful categorization of Reddit posts.

6.2 Saving and Utilizing the Trained Model

Once the BERTopic model is trained, it is saved for future use. Saving the model enables:

- **Efficient Deployment**: The model can be loaded and used without retraining, reducing processing time.
- **Consistency**: Using the same trained model ensures consistent topic predictions across different executions.
- **Scalability**: The trained model can be integrated into a larger system for continuous topic extraction from streaming data.

The trained BERTopic model is saved as `bertopic_model` and can be reloaded for real-time topic extraction in the consumer pipeline. This structured approach ensures that topic analysis remains efficient and scalable across different stages of data processing.

7 Topic Extraction Consumers

The topic extraction consumer is responsible for reading messages from the `sentiment_analyzed` topic, performing topic modeling on the titles of Reddit posts, and then categorizing them based on the extracted topics. The processed information is stored in a CSV file and visualized using a word cloud.

7.1 Initialization and Configuration

The consumer initializes by connecting to the Kafka broker and subscribing to the `sentiment_analyzed` topic.

- **Kafka Configuration**: The consumer connects to the Kafka broker using the `bootstrap_servers` parameter and listens to messages starting from the earliest offset.
- **Topic Modeling with BERTopic**: A pre-trained BERTopic model is loaded to categorize posts based on their textual content.
- **Word Cloud Generation**: A word cloud visualization is generated based on the most frequently occurring topics.
- **CSV Output**: The consumer writes the analyzed data, including the title, sentiment, confidence score, and topic description, to a CSV file for storage and further analysis.

7.2 Message Processing Workflow

The consumer follows these steps to process messages:

1. Reads a message from the assigned partition of the `sentiment_analyzed` topic.
2. Extracts the `title`, `sentiment`, and `confidence` fields from the message payload.
3. Cleans and tokenizes the title using a text preprocessing function that removes unwanted links, special characters, and stopwords.
4. Performs topic modeling on the cleaned title using the BERTopic model.
5. Logs the results, including the partition from which the message was read.
6. Updates the topic frequency counter to track the most common topics.
7. Stores the processed data in a CSV file.
8. Upon termination, generates a word cloud to visualize the extracted topics.

7.3 Error Handling and Termination

The consumer is designed to handle errors gracefully:

- **Keyboard Interrupt**: The consumer can be stopped manually, ensuring all processed messages are saved before termination.
- **Exception Handling**: Any unexpected errors are logged, allowing for debugging and resolution without disrupting the overall pipeline.

7.4 Key Features and Benefits

- **Partition Awareness**: The code will be run twice, and each instance will read from a different partition assigned by Kafka.
- **Real-Time Topic Extraction**: Topics are assigned in real-time based on the latest Reddit posts.
- **Data Categorization**: Posts are categorized into meaningful topics, enabling structured analysis.
- **Visualization**: A word cloud is generated after processing to provide an intuitive summary of the most frequently discussed topics.

7.5 Example Outputs

Below are example outputs generated by the topic extraction consumer:

```
Are You Experienced by Jimi Hendrix, Neutral, 0.8792815208435059,
"rock, actions, around, one, korn, sparks, woodstock, dokken,
gilded"

is there a missing godfather 2 love scene?, Neutral, 0.8356384038925171,
"movie, movies, best, poster, films, starring, film, comedy, official,
trailer"

JSA and LGBT+: I'm surprised no one's talking about this, Negative,
0.5650902986526489, Outlier / Uncategorized
```

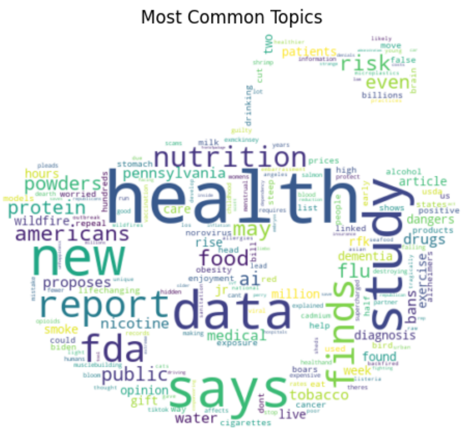


Figure 1: Example Word Cloud Generated from Extracted Topics

7.6 Workflow Summary:

1. Messages are consumed from Kafka in real-time.
2. Titles are processed using NLP techniques.
3. Topics are assigned using BERTopic.
4. Processed data is stored in a CSV file.

5. A word cloud is generated after processing is complete.

This consumer ensures efficient real-time processing of categorized Reddit posts, allowing for better analysis and visualization of trending discussions.

8 Conclusion

This project demonstrates a complete real-time data pipeline using Kafka and Python. It showcases integration of sentiment analysis and topic modeling for dynamic text data, with a focus on efficiency and scalability. The insights obtained can be visualized through word clouds and categorized outputs for deeper analysis.