EcoStream: Integrating Kafka and FinBERT for Economic News Sentiment Analysis

LAMJOUN Jihane, MERRHEIM Maïssane

21 December 2023

GitHub Repository: https://github.com/maissanemerrheim/Infostream.git

1 Introduction

In today's financial landscape, staying informed and analyzing sentiment in economic news is crucial for informed decision-making. To make wise decisions, you need to understand the market, spot trends, and stay up-to-date with financial news. This applies whether you're an experienced financial expert or someone interested in making smart investments. Having access to current and reliable information is a top priority.

Our project is here to help with a real-time economic news analysis system that provides users with current insights into the sentiment of financial news articles. By using the News API, Kafka, and the FinBERT sentiment analysis model, we've created an efficient and automated solution for tracking and analyzing the sentiment of economic news.

The EcoNews Producer collects news articles based on keywords chosen by the user, while the EcoNews Consumer uses the FinBERT model to analyze sentiment. This project offers a user-friendly tool that helps professionals and enthusiasts make informed decisions in the fast world of finance.

2 Infrastructure and Data Flow

Kafka is a distributed streaming platform designed for building real-time data pipelines and streaming applications. It is often used for ingesting, storing, and processing large volumes of data in real-time. Kafka is known for its durability, fault tolerance, and scalability. Here, we are using Kafka to send and consume news articles.

In this project, we are using the News API to source financial news articles. The News API is a web service that provides access to news articles from various sources. The News API allows us to specify parameters like language and keywords to filter the news articles you want to retrieve.In this project we filtered on English-language articles and on financial or

economic themes.

The code project includes two Python scripts: kafka_eco_news_consumer.py and kafka_eco_news_producer.py These scripts are part of a data pipeline for fetching news articles related to specific keywords, performing sentiment analysis on the articles, and storing the results. We follow PEP 8 coding standards to ensure clean and readable code. Before running the scripts, ensure that you have ZooKeeper server and Kafka server running. To fetch news articles and send them to Kafka, run the producer script with the your chosen keyword as an argument. Then, run the consumer script that subscribes to the Kafka topic you specify in argument, processes messages, and performs sentiment analysis. For a detailed explanation of the commands to execute, please refer to the README, which provides step-by-step instructions for setting up and running the data pipeline.

Thus, the producer script fetches news articles based on keywords, sends them to a Kafka topic, and continuously repeats this process. the consumer script subscribes to the Kafka topic, processes messages, performs sentiment analysis, and stores results in CSV files specific to the topic. Together, these scripts form a data pipeline for collecting, analyzing, and storing news articles related to specific keywords, with Kafka acting as the message broker between them. Below is a detailed explanation of the data architecture and data flow for this project:

Data Flow Overview:

1. Producer (`kafka_eco_news_producer.py`)

- It fetches news english articles related to specific keywords from an external news API and sends them to a Kafka topic.
- The data flow in the producer script can be broken down into the following steps:
    1. **Initialization :** The script is initialized with the Kafka broker address, API key for the news service, and keywords for news search.
    2. **Topic Creation :** The script creates a Kafka topic if it doesn't already exist. The topic name is derived from the provided keywords.
    3. **Fetching and Sending Articles :** The script fetches news articles using the News API, based on the specified keywords. It then sends these articles as messages to the Kafka topic.
    4. **Continuous Execution :** The script runs in a loop, continuously fetching and sending articles every 10 seconds.
- The script uses the `kafka-python` library to interact with Kafka and serializes messages to JSON format before sending them.

2. Consumer (`kafka_eco_news_consumer.py`)

- It subscribes to the Kafka topic created by the producer, processes incoming messages (which are news articles and end-of-batch indicators), and performs sentiment analysis on the articles.

- The data flow in the consumer script can be explained as follows:

    1. **Initialization :** The script is initialized with the Kafka broker address and the input topic to subscribe to.
    2. **Topic Existence Check :** It verifies that the specified Kafka topic exists; otherwise, it raises an exception.
    3. **Message Processing :** The script continuously consumes messages from the Kafka topic.
    4. **Sentiment Analysis :** When it encounters an "end of batch" message, it performs sentiment analysis on the collected news articles using the FinBERT model. Sentiment analysis results are stored in a CSV file specific to the topic.
    5. **Reset for the Next Batch :** After processing a batch of articles and performing sentiment analysis, the script resets and waits for more messages.

- The script uses various libraries, such as `kafka-python`, `transformers` (for FinBERT sentiment analysis), and `pandas`.

Overall, this data architecture and flow enable real-time ingestion and sentiment analysis of news articles based on specified keywords, with results stored in separate CSV files for each topic. It allows for scalability and extensibility, making it easy to add more topics and keywords for analysis.

3 Sentiment Analysis with FinBERT

As a part of the project, we decided to focus ourselves on the retrieved real-time business data, and to apply sentiment analysis to it. To do so, we used the FinBERT model.
FinBERT is a pre-trained NLP model that is used to apply sentiment analysis on financial texts. This model was built from the BERT language model and fine-tuned for financial sentiment classification. It gives 3 different outputs : "Positive", "Negative", "Neutral". FinBERT analyzes the data by capturing the relationships between the words, and the sentences and understanding the context on financial text data. Therefore, if a stock price index or a cryptocurrency often appears alongside "positive" words, the model will predict that the article holds a "positive" sentiment, assigning a higher probability to this classification. We decided to focus our study on the stock market indices and cryptocurrencies, as these particular indices are the ones that are the most used by professionals for making wise and informed financial decisions.
The results we obtained on stock price indices are the following :

- **Nikkei 225** :14% are positive articles, 20% are neutral, and 61% hold a negative sentiment.

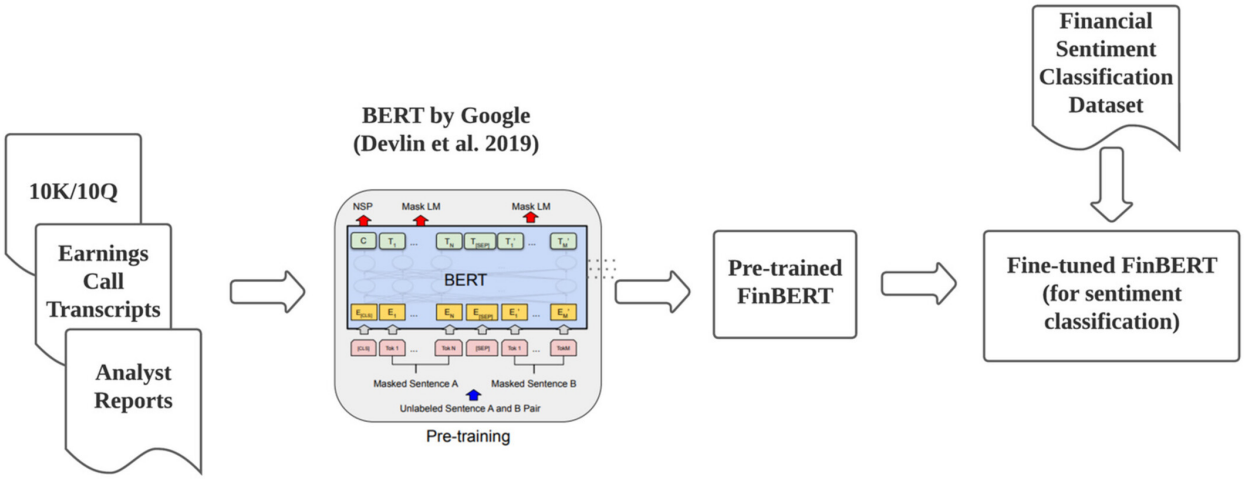

Figure 1: FinBERT

- **DJIA** : 18% are positive articles, 34% are neutral, and 48% hold a negative sentiment.
- **CAC40** : 34% are positive articles, 23% are neutral, and 41% hold a negative sentiment.
- **NASDAQ** : 15% are positive articles, 43% are neutral, and 42% hold a negative sentiment.
- **FTSE 100** : 14% are positive articles, 48% are neutral, and 38% hold a negative sentiment.
- **DAX** : 11% are positive articles, 69% are neutral, and 20% hold a negative sentiment.

Figure 2: Stock Market Indices

These results were obtained on the 21st of December in the morning. Therefore, at this time, according to our sentiment analysis and the data we extracted, it doesn't seem to be a great idea to invest on Nikkei 225, or DJIA companies. For the rest of the stock market indices, the results we obtained are more nuanced since a large part of the articles hold a neutral sentiment analysis, specially for the DAX, FTSE 100 and NASDAQ companies. Professionals or enthusiasts might need to extract other type of real-time data than news articles, to make an informed decision on where to invest, or to wait a little bit to see if the sentiment analysis result might change from on hour to another.
The results we obtained on cryptocurrencies on the 21st of December in the morning are the following :

- **Chainlink**: 17% are positive articles, 74% are neutral, and 9% are negative.
- **Bitcoin (BTC)** : 13% are positive articles, 64% are neutral, and 23% are negative.
- **Ethereum (ETH)** : 13% are positive articles, 18% are neutral, and 69% hold a negative sentiment.
- **Solana** : 29% are positive articles, 19% are neutral, and 52% hold a negative sentiment.
- **Polygon** : 3% are positive articles, 90% are neutral, and 3% hold a negative sentiment.

After retrieving this real-time data, it is always intresting to explain the sentiment analysis results that were obtained. According to these pie charts, Solana and Chainlink can be interesting to invest on, since most of the articles related to these cryptocurrencies, are

either positive or neutral. However, investing on Bitcoin or Polygon or even Ethereum cryptocurrencies might be dangerous, since they are more often related to articles that were classified as negative or neutral.

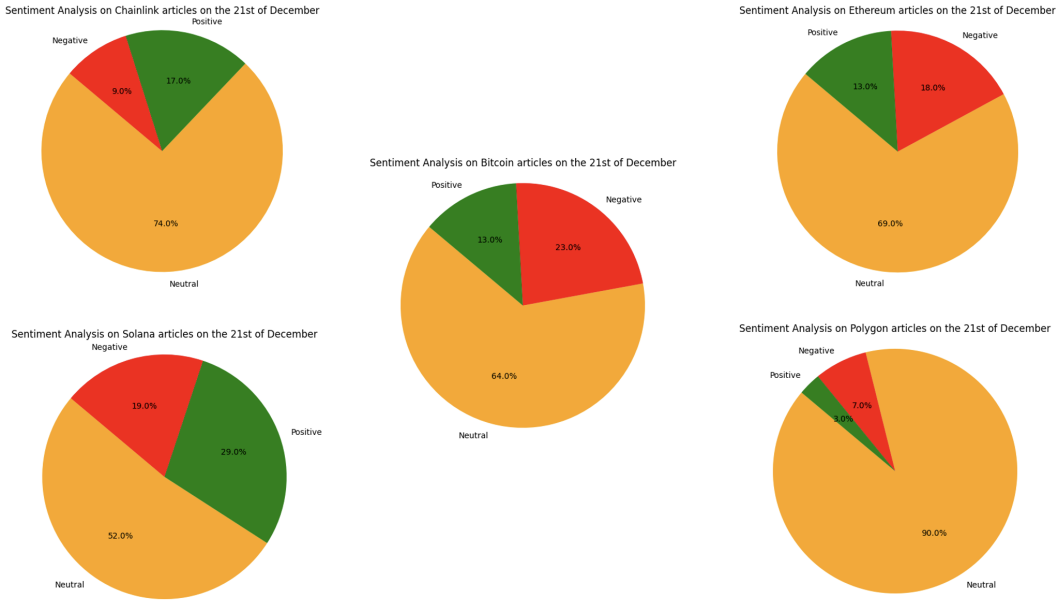

Figure 3: Crypto-currencies

4 Conclusion : Challenges and Solution

Kafka and FinBert are great tools to extract real-time data and apply sentiment analysis on financial text data. We obtained interesting results that could help professionals and enthusiasts to make an wise and informed choice when choosing on what companies or cryptocurrency to invest on. We could have pursue our work on other stcok market indices, or other cryptocurrencies. It would have also been interesting to extract other type of real-time data (videos, graphs etc) and compare the sentiment analysis applied on these data with the one that we retrieved thanks to News API. One can also notice that our work was done a free version of News API. Therefore, the amount of retrieved data (maximum 100 articles), is not always sufficient to apply sentiment analysis on, even more, when it comes to financial data. Thus, it would have been beneficial to collect a larger volume of data for conducting sentiment analysis, which could lead to more accurate results.