

Project Documentation: News Sentiment Analysis

Introduction

In today's fast-paced financial markets, the sentiment conveyed in news articles can significantly influence stock prices and investor decisions. This project aims to analyze sentiment from news descriptions using Natural Language Processing (NLP) and machine learning techniques. By classifying sentiments into categories such as positive, negative, and neutral, this analysis provides insights that can guide investors in making informed decisions.


Project Overview

The project consists of several key steps:

1. Data Loading and Exploration
2. Data Preprocessing
3. Feature Extraction
4. Model Training
5. Model Evaluation
6. Visualization

1. Data Loading and Exploration

The first step involves loading the dataset and conducting exploratory data analysis (EDA) to understand its structure and contents.



```
import pandas as pd

file_path = '/content/news_sentiment_analysis.csv'
data = pd.read_csv(file_path)

data.head()
data.info()
```

	Source	Author	Title	Description	URL	Published At	Sentiment	Type
0	stgnews	Bridger Palmer	Pine View High teacher wins Best in State awar...	ST. GEORGE — Kaitlyn Larson, a first-year teac...	https://www.stgeorgeutah.com/news/archive/2024...	2024-07-12T23:45:25+00:00	positive	Business
1	Zimbabwe Mail	Staff Reporter	Businesses Face Financial Strain Amid Liquidit...	Harare, Zimbabwe – Local businesses are grappl...	https://www.thezimbabwemail.com/business/busin...	2024-07-12T22:59:42+00:00	neutral	Business
2	4-traders	NaN	Musk donates to super pac working to elect Tru...	(marketscreener.com) Billionaire Elon Musk has...	https://www.marketscreener.com/business-leader...	2024-07-12T22:52:55+00:00	positive	Business
3	4-traders	NaN	US FTC issues warning to franchisors over unf...	(marketscreener.com) A U.S. trade regulator on...	https://www.marketscreener.com/quote/stock/MCD...	2024-07-12T22:41:01+00:00	negative	Business
4	PLANET	NaN	Rooftop solar's dark side	4.5 million households in the U.S. have solar ...	https://www.npr.org/2024/07/12/1197961036/roof...	2024-07-12T22:28:19+00:00	positive	Business

Explanation

- **Data Loading:** We utilize the pandas library to load the dataset from a CSV file into a DataFrame. This allows us to easily manipulate and analyze the data.
- **Initial Exploration:** We call methods like `.head()` to view the first few rows of the dataset and `.info()` to get insights about the data types and null values present in each column. This step is crucial for understanding what data we are working with.

2. Data Preprocessing

This step prepares the text data for analysis. Proper preprocessing is essential for improving the accuracy of our model.

```
data['Description'] = data['Description'].astype(str)
texts = data['Description']
labels = data['Sentiment']

from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
encoded_labels = label_encoder.fit_transform(labels)
```

```
encoded_labels

array([2, 1, 2, ..., 2, 1, 2])
```

Explanation

- **Text Conversion:** The 'Description' column is converted to string format to ensure uniformity, as some entries might be in different formats.
- **Label Encoding:** We use the LabelEncoder from sklearn to transform sentiment labels (like positive, negative, neutral) into numerical values. This encoding is necessary because machine learning algorithms require numerical input.

3. Feature Extraction

In this phase, we convert the textual data into numerical features that can be fed into our machine learning model.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(texts, encoded_labels, test_size=0.2,
                                                    random_state=42, stratify=encoded_labels)

from sklearn.feature_extraction.text import TfidfVectorizer

custom_stop_words = ['the'] + [str(i) for i in range(10)]

vectorizer = TfidfVectorizer(max_features=1000, stop_words=custom_stop_words)

X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test.astype(str))
```

Explanation

- **Custom Stop Words:** We define a custom list of stop words that includes common English words (like "the") and numeric values (0-9). This helps in filtering out irrelevant information that does not contribute to the sentiment analysis.
- **TF-IDF Vectorization:** We use the TfidfVectorizer from sklearn to convert the text data into TF-IDF feature vectors. TF-IDF measures the importance of a word in a document relative to a collection of documents (corpus). This transformation captures the essence of the text data in a numerical format suitable for modeling.

4. Model Training

Here, we select and train a machine learning model to classify sentiments based on the extracted features.

```
from sklearn.ensemble import RandomForestClassifier

model_forest = RandomForestClassifier(n_estimators=100,
                                     random_state=42)
model_forest.fit(X_train_tfidf, y_train)
```

Explanation

- **Random Forest Classifier:** We employ a Random Forest Classifier, a robust ensemble learning method that combines multiple decision trees to improve classification accuracy. This model is particularly effective in handling the complexity of sentiment classification.
- **Training the Model:** The model is trained on the processed training dataset (X_train_tfidf and y_train). After training, we use the model to predict sentiments on the test dataset (X_test_tfidf).

5. Model Evaluation

Evaluating the model's performance is critical to understanding its effectiveness.

```
from sklearn.metrics import classification_report, accuracy_score

y_pred_forest = model_forest.predict(X_test_tfidf)

accuracy_Forest = accuracy_score(y_test, y_pred_forest)
report_Forest = classification_report(y_test, y_pred_forest, target_names=label_encoder.classes_)

print(f'Accuracy: {accuracy_Forest}')
print('Classification Report:')
print(report_Forest)
```

Accuracy: 0.8714285714285714				
Classification Report:				
	precision	recall	f1-score	support
negative	0.99	0.59	0.74	115
neutral	0.88	0.82	0.85	158
positive	0.85	0.97	0.91	427
accuracy			0.87	700
macro avg	0.91	0.79	0.83	700
weighted avg	0.88	0.87	0.87	700

Explanation

- **Accuracy Score:** We calculate the model's accuracy by comparing the predicted sentiments against the actual sentiments in the test dataset. Accuracy is a simple metric that gives a quick overview of the model's performance.
- **Classification Report:** The classification report provides detailed metrics such as precision, recall, and F1-score for each sentiment category. These metrics help us understand how well the model performs for each class.

Display Actual vs. Predicted Samples

```
# Display actual vs predicted for a few samples
sample_size = 10
sample_indices = np.random.choice(range(len(y_test)), size=sample_size, replace=False)
for i in sample_indices:
    print(f"Text: {X_test.iloc[i]}")
    print(f"Actual: {label_encoder.inverse_transform([y_test[i]])[0]}, Predicted: {label_encoder.inverse_transform([y_pred_forest[i]])[0]}\n")
```

Explanation

1. **Sample Size Definition:**
 - sample_size = 10: This line defines the number of samples to display. In this case, we choose 10 random samples from the test set.
2. **Random Sample Selection:**
 - sample_indices = np.random.choice(range(len(y_test)), size=sample_size, replace=False):
 - Here, we use np.random.choice() to select 10 unique indices randomly from the range of the test dataset (y_test).
 - The replace=False argument ensures that the same index is not chosen more than once, which helps in getting distinct samples.
3. **Looping Through Sample Indices:**
 - for i in sample_indices:: This loop iterates over each randomly selected index to print the corresponding text and its predictions.
4. **Displaying Text and Predictions:**

- `print(f"Text: {X_test.iloc[i]}")`: This line prints the actual text (news description) from the test set corresponding to the randomly selected index `i`.
- `print(f"Actual: {label_encoder.inverse_transform([y_test[i]])[0]}, Predicted: {label_encoder.inverse_transform([y_pred_forest[i]])[0]}")`:
 - **Here, we display both the actual sentiment and the predicted sentiment.**
 - `label_encoder.inverse_transform([y_test[i]])[0]` converts the numerical label back to its original sentiment label (like 'positive', 'negative', etc.) for the actual value.
 - `label_encoder.inverse_transform([y_pred_forest[i]])[0]` does the same for the predicted value.
 - **The 0 indexing is used because `inverse_transform()` returns an array, and we want to access the first (and only) element.**

```
Text: hbo The 'Euphoria' cast is very busy, but they'll find a way to fit in season 3.
Actual: positive, Predicted: neutral

Text: Tribune OnlineInsecurity: COAS charges field commanders on preparednessThe Chief of Army Staff (COAS), Lieutenant General Taoreed Lagbaja, in a bid to address the evolving securi
Actual: positive, Predicted: positive

Text: Todos los países que habían interrumpido su financiación a la Agencia de la ONU para los Refugiados Palestinos la han reanudado a fecha de hoy, con excepción de Estados Unidos y
Actual: positive, Predicted: positive

Text: By McKenzy Parsons Click here for updates on this story &#160;&#160;&#160;&#160;LINCOLN, Nebraska (KETV) &#8212; The Nebraska Attorney General released an opinion concluding LB ;
Actual: positive, Predicted: positive

Text: It will include presentations from the Minnesota Pollution Control Agency, Department of Health and the city of Lake Elmo.
Actual: neutral, Predicted: positive

Text: The man who tried to assassinate Trump had photos on his phone of Trump, Biden and other officials, including Attorney General Merrick Garland and FBI Director Chris Wray.
Actual: negative, Predicted: negative

Text: OF NOTE Fairview Health Services, a Minneapolis-based metro-wide operator of hospitals and clinics, announced the appointment of Tanja Oquendo as chief people officer, effective
Actual: positive, Predicted: positive

Text: « La Déclaration de politique générale (Dpg) est prête depuis au moins le 10 juin » a révélé Dr Abdourahmane Diouf. Il était l&#8217;invité de l&#8217;émission « Grand Jury » de
Actual: positive, Predicted: positive

Text: (marketscreener.com) Canadian Prime Minister Justin Trudeau met with a former Bank of Canada Governor last week about eventually joining his Liberal government, days after the Li
Actual: positive, Predicted: positive

Text: Instrumental in establishing the Sree Chitra Tirunal Institute for Medical Sciences and Technology
Actual: neutral, Predicted: neutral
```

Output:

- **Each iteration outputs the text along with its actual and predicted sentiment, allowing you to compare how well the model performed on those specific samples. This helps in visually assessing the model's accuracy and understanding any misclassifications.**

6. Visualization

Visual aids can enhance understanding of the model's performance and sentiment distribution.

Confusion Matrix

```

from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

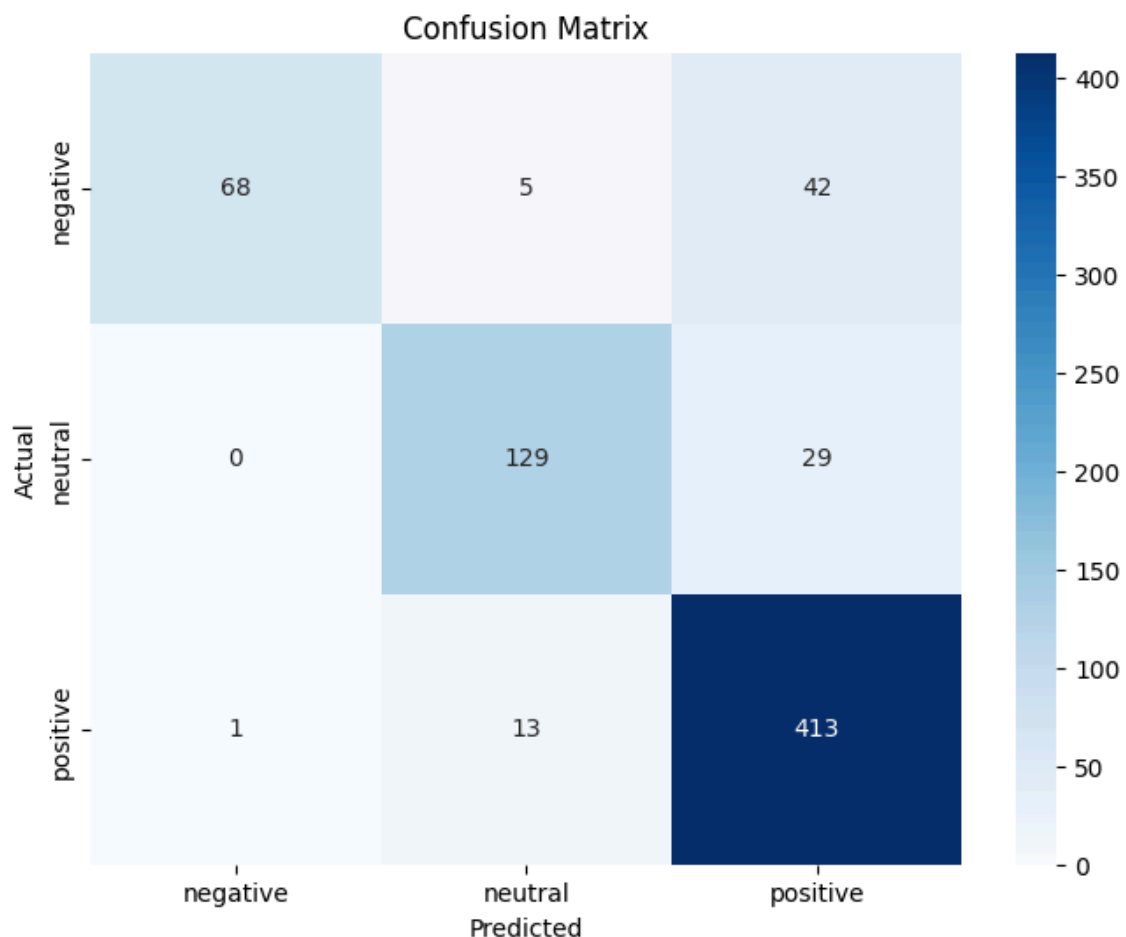
conf_matrix = confusion_matrix(y_test, y_pred_forest)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=label_encoder.classes_,
            yticklabels=label_encoder.classes_)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix')
plt.show()

```

Explanation

- The confusion matrix is a heatmap that visualizes the model's predictions against the actual labels. It highlights where the model is performing well and where it might be misclassifying sentiments.



Bar Chart of Sentiment Distribution

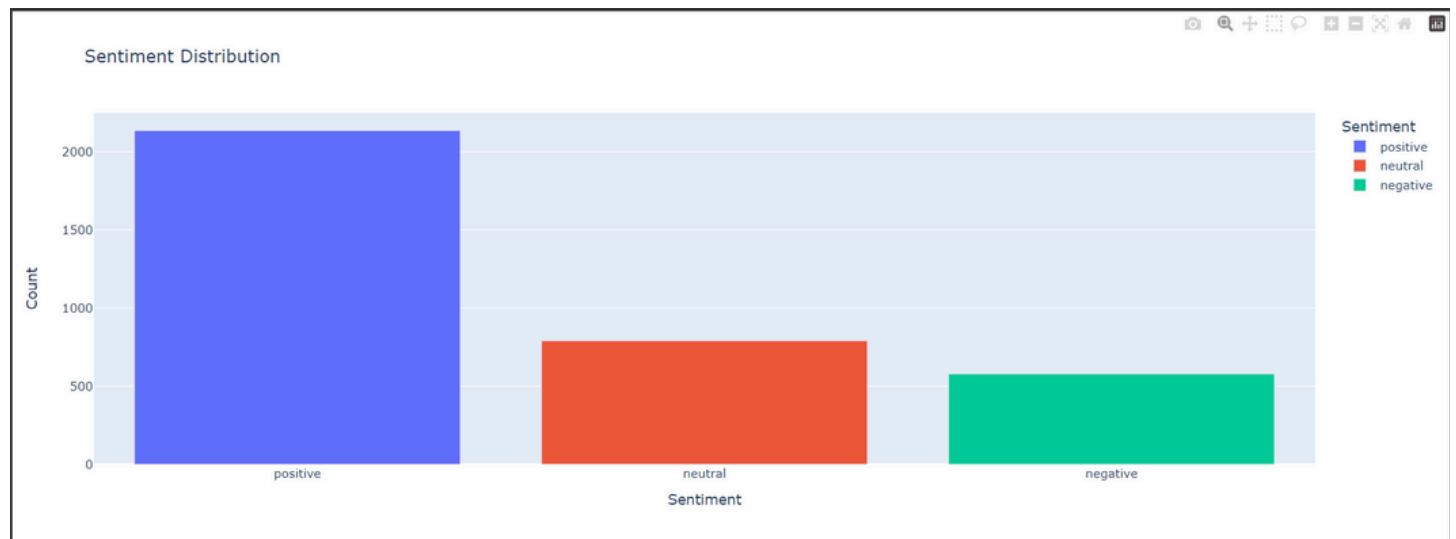
```
sentiment_counts = data['Sentiment'].value_counts()
sentiment_df = sentiment_counts.reset_index()
sentiment_df.columns = ['Sentiment', 'Count']

import plotly.express as px

fig = px.bar(sentiment_df, x='Sentiment', y='Count', color='Sentiment', title='Sentiment Distribution')
fig.show()
```

Explanation

- This bar chart displays the count of each sentiment category (positive, negative, neutral) in the dataset. It provides a quick overview of sentiment distribution, which can be useful for stakeholders.



Pie Chart of Sentiment Distribution


```
import plotly.express as px

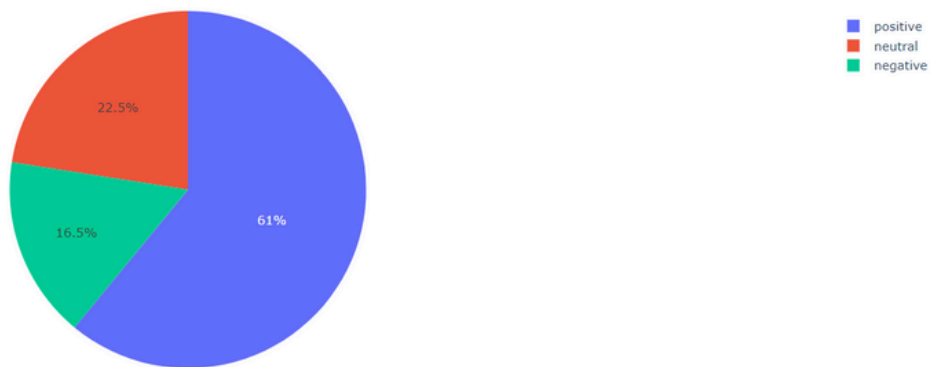
# Create a pie chart with Plotly
fig = px.pie(sentiment_df,
              values='Count',
              names='Sentiment',
              title='Sentiment Distribution')

# Show the figure
fig.show()
```

Explanation

- The pie chart visually represents the proportion of each sentiment category. It helps in understanding the overall sentiment landscape of the news articles.

Sentiment Distribution



Real-Time Stock News Sentiment Analysis - Detailed Documentation

Overview

This project is designed to perform **real-time sentiment analysis** on stock-related news articles using **Kafka** (for data ingestion), **Apache Spark** (for real-time processing), and **Cassandra** (for data storage). The goal is to process live news articles, analyze their sentiment

using a **pre-trained machine learning model**, and store the results for further analysis and visualization.

The architecture allows us to classify the sentiment of incoming news articles into three categories: **positive**, **neutral**, and **negative**. This real-time sentiment analysis can provide valuable insights into how the market and stock prices might be influenced by news events.

Detailed Workflow

1. Setting Up Apache Spark

The process begins by setting up a **SparkSession**, which initializes the environment and enables communication with **Kafka** and **Cassandra**.

[Insert Code Here: SparkSession Setup]

A code editor window with a dark background and light-colored text. It contains Python code for setting up a SparkSession. The code includes imports, session builder configuration with appName, streaming.kafka.useDeprecatedOffsetFetching, and jars.packages, and a call to getOrCreate().

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("KafkaSparkSentimentAnalysis") \
    .config("spark.sql.streaming.kafka.useDeprecatedOffsetFetching", "true") \
    .config("spark.jars.packages",
            "org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2,"
            "com.datastax.spark:spark-cassandra-connector_2.12:3.0.0") \
    .getOrCreate()
```


Explanation:

- **SparkSession** is the entry point for any Spark application, configured with the necessary settings to connect to Kafka and Cassandra.

2. Reading Data from Kafka

Apache Kafka serves as the message broker that streams stock news articles. In this step, Spark listens to a specified Kafka topic (news_data), where news articles are constantly being published.

[Insert Code Here: Kafka Data Reading]



```
# Kafka broker information and topic name
kafka_brokers = 'localhost:9092'
kafka_topic = 'news_data'

# Read stream from Kafka
df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", kafka_brokers) \
    .option("subscribe", kafka_topic) \
    .load()
```


Explanation:

- Kafka allows for real-time ingestion of stock news data by reading messages from a topic where news articles are continuously streamed.

3. Processing the Kafka Data

The data coming from Kafka is in a binary format. We need to cast it into a string format and define a schema to parse the incoming JSON data.

[Insert Code Here: Kafka Data Processing and Schema Definition]



```
# Define schema for the incoming data
schema = "category STRING, datetime BIGINT, headline STRING, id INT, image STRING, related STRING, source STRING, summary STRING, url STRING"

# Convert the Kafka value column from bytes to JSON format
df = df.selectExpr("CAST(value AS STRING)")

# Parse the JSON data and extract fields based on the schema
df = df.selectExpr("value as json").selectExpr("from_json(json, '{}') as data".format(schema)).select("data.*")
```

Explanation:

- **Schema:** Defines the structure of the incoming data, allowing us to extract specific fields such as the summary, headline, and more.
- **CAST to STRING:** Converts the raw binary data into a readable string format.
- **JSON Parsing:** Parses the string data to a structured format, enabling us to manipulate fields like the summary for further processing.

4. Loading Pre-trained Machine Learning Model

We load a **pre-trained machine learning model** (a Random Forest classifier) and a **TF-IDF vectorizer**. These are used to classify the sentiment of each news article's summary.

```
import joblib

# Load the pre-trained sentiment model and TF-IDF vectorizer
model = joblib.load('random_forest_model.pkl')
vectorizer = joblib.load('tfidf_vectorizer.pkl')
```

Explanation:

- The **model** is trained to predict sentiment based on news text.
- The **TF-IDF vectorizer** converts the raw text into numerical features that the model can use to make predictions.

5. Sentiment Analysis Function

A **User-Defined Function (UDF)** is created to predict the sentiment of each news article by cleaning the text, vectorizing it using the pre-trained TF-IDF model, and then applying the Random Forest model for prediction.

```
import re
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType

# Mapping of numerical sentiment prediction to sentiment categories
sentiment_mapping = {
    0: "negative",
    1: "neutral",
    2: "positive"
}

# Define a UDF for sentiment analysis
def predict_sentiment(summary):
    if summary is None:
        return "unknown"
    try:
        # Clean the text (lowercase and remove punctuation)
        summary_cleaned = re.sub(r'[^\w\s]', '', summary.lower())
        # Transform the cleaned text into a TF-IDF vector
        summary_vectorized = vectorizer.transform([summary_cleaned])
        # Predict sentiment using the pre-trained model
        prediction = model.predict(summary_vectorized)
        # Map numerical prediction to sentiment category
        return sentiment_mapping.get(prediction[0], "unknown")
    except Exception as e:
        return "unknown"

# Register the UDF
sentiment_udf = udf(predict_sentiment, StringType())
```

Explanation:

- **Text Cleaning:** Removes unnecessary characters and normalizes the text (e.g., by converting it to lowercase).
- **TF-IDF Transformation:** Converts the cleaned text into a numeric format understandable by the model.
- **Prediction:** The model predicts whether the sentiment is positive, neutral, or negative.

6. Applying Sentiment Analysis to Streaming Data

We apply the sentiment analysis function to the streaming data, allowing us to predict the sentiment for each news article in real-time.

```
# Apply the UDF to get the sentiment of the summary
df_with_sentiment = df.withColumn("sentiment", sentiment_udf(col("summary")))

# Select only the summary and the predicted sentiment
df_selected = df_with_sentiment.select("summary", "sentiment")
```

Explanation:

- The sentiment analysis UDF is applied to each incoming news article, generating a new sentiment column that contains the sentiment prediction for every article.

7. Writing Data to Cassandra

The sentiment predictions are then written to **Cassandra**, where the results are stored for future querying and analysis.

```
# Define Cassandra keyspace and table
cassandra_keyspace = "stock_stream"
cassandra_table = "news_sentiment"

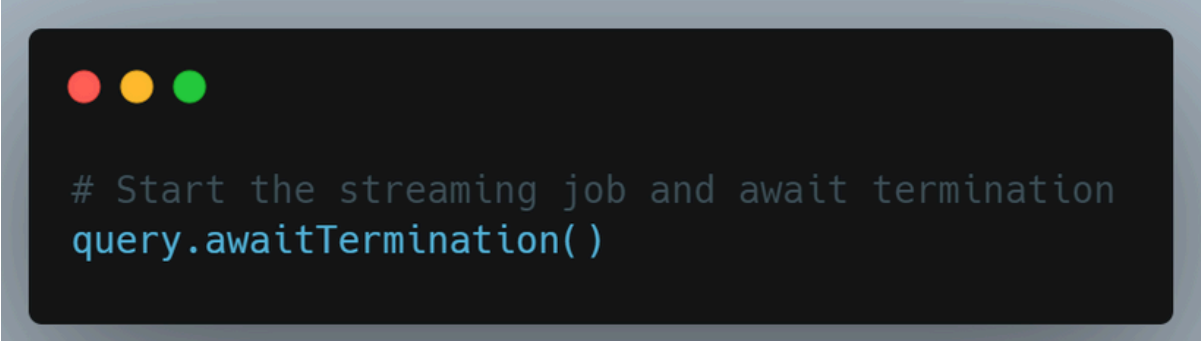
# Write the data to Cassandra using foreachBatch for micro-batch processing
query = df_with_sentiment.writeStream \
    .outputMode("append") \
    .foreachBatch(lambda batch_df, batch_id:
        batch_df.write \
            .format("org.apache.spark.sql.cassandra") \
            .options(table=cassandra_table, keyspace=cassandra_keyspace) \
            .mode("append") \
            .save()) \
    .start()
```

Explanation:

- The **Cassandra keyspace** and **table** store the summarized news and their associated sentiment, ensuring that all predictions are stored for further use.

8. Running the Stream

Finally, the stream is set to run indefinitely, processing incoming news articles as they arrive in real-time.




```
# Start the streaming job and await termination
query.awaitTermination()
```

Explanation:

- The stream keeps running, continuously reading new data from Kafka, processing it with the sentiment model, and storing the results in Cassandra.

Cassandra Table Schema

The table in Cassandra that stores the sentiment analysis results can be created with the following schema:



```
CREATE TABLE stock_stream.news_sentiment (
    summary text PRIMARY KEY,
    sentiment text
);
```

Schema Explanation:

- The summary field contains the cleaned news article summary.
- The sentiment field contains the predicted sentiment (positive, neutral, or negative).

Conclusion

This real-time stock sentiment analysis system integrates **Kafka**, **Spark**, and **Cassandra** to provide a scalable and fast way to analyze stock-related news in real-time. The system can be

expanded by integrating more complex NLP models or enhanced with better alerting mechanisms and visualizations to provide valuable insights into the relationship between news sentiment and stock prices.

Real-World Applications

This project addresses several real-world challenges, particularly in the stock market:

☐ **Investor Decision-Making:**

- By analyzing public sentiment from news articles, investors can monitor market sentiment and make informed decisions regarding stock purchases or sales. For instance, if negative sentiment is detected towards a particular company, investors may choose to divest.

☐ **Market Trend Prediction:**

- Understanding sentiment trends can help predict market movements. If positive sentiment is rising in a sector, it may indicate potential stock price increases, assisting investors in timing their entries and exits.

☐ **Risk Management:**

- By assessing negative sentiments towards specific stocks or sectors, investors can mitigate risks and avoid potential losses. This is crucial for maintaining a balanced portfolio.

Conclusion

This sentiment analysis project leverages machine learning to transform qualitative news data into quantitative insights, aiding investors in navigating the complexities of the stock market. By employing NLP techniques, this project helps to address critical challenges in investment decision-making, market trend prediction, and risk management.