Sentiment Analysis on Live Streaming Comments Using Deep Learning Algorithms

Tejas Pinjala (TSP220001), Shariq Azeem (MSA200007), Dhanyan Muralidharan (DTM220000), Thennannamalai Malligarjunan (TXM230003)

ABSTRACT

The explosion of big data has necessitated the evolution of systems that can handle continuous, rapid data streams, giving rise to a burgeoning field focused on real-time analytics. This discipline addresses the need for instantaneous insights into current events, posing significant challenges due to the infeasibility of storing every data instance. Data preprocessing is crucial to ensure that only concise summaries are maintained in main memory. With the rapid influx of data, it is imperative to process each piece quickly enough to avoid data loss, while also achieving high analytical precision. Apache Kafka is a widely used, open source distributed streaming platform, integral to building applications that rely on networked data streams. Kafka has emerged as a leading solution for managing streaming data in diverse contexts. It employs a publish/subscribe (pub/sub) messaging system, enabling real-time data processing and distribution across multiple network nodes. This makes Kafka a highly scalable and fault-tolerant solution, capable of managing vast amounts of data efficiently.

I. Introduction and Background Work

Real-time sentiment analysis of social streams enables an understanding of the immediate thoughts and expressions of social media users. This capability offers significant opportunities in data-driven marketing, disaster prevention, business crisis management, and social media debates. Real-time sentiment analysis has applications across nearly all business and industry sectors.

Data stream mining involves extracting informational structures as models and patterns from continuous, evolving data streams. Traditional data analysis methods require data storage and offline processing using complex algorithms that make multiple passes over the data. However, data streams are inherently infinite and generated at high rates, making it impractical to store them in main memory. This context presents challenges in storage, querying, and mining, particularly related to the computational resources needed to analyze such large volumes of data. Consequently, numerous approaches have been studied and developed to provide accurate and efficient algorithms.

In real-time data stream mining, data is processed online to ensure results are current and queries can be answered with minimal delay. Current solutions for data stream sentiment analysis typically perform

sentiment analysis offline on stored data samples. While this method is effective in some scenarios, it is inadequate for real-time applications. In big data contexts, the volume of data increases significantly over time, causing uniprocessor solutions to slow down, resulting in data loss, a reduced learning curve, and lower accuracy due to less data available for training. This inefficiency introduces high costs to these solutions. Considering the steps beyond data processing such as sentiment analysis and classification, we leverage the distributed environments offered by Kafka and Pyspark.

In this project, we relay Youtube Live comments into Kafka topics for real-time processing and feed it to a deep learning model (RNN) to predict the overall sentiment of a topic of discussion. We also leverage Pyspark's streaming dataframe to perform sentiment analysis with minimum latency.

II. Data Stream Processing

Apache Kafka is engineered as a distributed streaming platform capable of processing vast amounts of real-time data. Fundamentally, a Kafka cluster offers a publish-subscribe (pub/sub) messaging service, allowing data producers to publish information to Kafka topics and enabling consumers to subscribe to these topics and receive data instantly. In this system, producers are tasked with sending data to Kafka topics in various formats, such as text, binary, or JSON. Each data entry published to a Kafka topic includes a message with a key and a value: the key serves for identification, partitioning, and indexing, while the value holds the actual data content. YouTube live chat data is fetched using a python API and punched into Kafka to create topics, which are then fed into a pyspark dataframe object.

We perform a series of data manipulations to achieve stemming, stop word removal and named entity recognition. This is done by employing NLTK and sparkNLP libraries. At this stage, the pyspark streaming dataframe consists of data that is ready to be fed into a deep learning Neural Network for sentiment analysis.

III. RNN and LSTM models for Sentiment Analysis

Deep learning involves utilizing artificial neural networks for various learning tasks by employing multiple layers within these networks. This approach leverages the advanced learning capabilities of neural networks, which were previously considered feasible only with one or two layers and limited data. Neural networks, inspired by the structure of the human brain, consist of numerous information processing units known as neurons, arranged in layers that function collaboratively. These networks learn to execute tasks, such as classification, by adjusting the weights of the connections between neurons, mimicking the brain's learning process.

Recurrent Neural Networks (RNNs) are a type of neural network characterized by connections between neurons that form directed cycles. Unlike feedforward neural networks, RNNs can utilize their internal memory to process sequences of inputs, making them well-suited for sequential data.

This memory enables RNNs to perform the same task for each element in a sequence, with each output dependent on all previous computations, effectively remembering information about the sequence processed so far. The hidden stateacts as the network's memory, capturing information from all previous time steps. Unlike feedforward neural networks that use different parameters at each layer, RNNs share the same parameters across all time steps, significantly reducing the number of parameters to learn.

Deep LSTM RNNs have been effectively employed in speech recognition by stacking multiple LSTM layers, resulting in deeper architectures. Unlike traditional LSTM RNNs, which are already considered deep due to their unrolled, time-shared parameter structure, deep LSTM RNNs involve inputs traversing several nonlinear layers. Each input at a given time step passes through multiple LSTM layers, in addition to time propagation, allowing the network to learn at various temporal scales. This layered approach offers better parameter utilization, distributing them across multiple layers rather than simply increasing memory size.

Sentiment classification at the document level involves determining the overall sentiment of an opinion document, such as an online review, to ascertain whether it conveys a positive or negative sentiment. This is typically approached as a binary classification task. Alternatively, it can be framed as a regression task to predict an overall rating score (e.g., from 1 to 5 stars) or as a five-class classification task.

Sentiment classification is considered a specific type of document classification, where the representation of the document is crucial, as it must accurately reflect the conveyed information through words or sentences. Traditionally, the Bag-of-Words (BoW) model is used to create text representations in natural language processing (NLP) and text mining. This model treats a document as a collection of its words. Using BoW, a document is converted into a fixed-length numerical feature vector, where each element can represent word occurrence, word frequency, or TF-IDF score, with the vector's dimension equal to the vocabulary size. Typically, these document vectors are very sparse because any given document contains only a small subset of the vocabulary. Early neural networks utilized such feature settings for text representation.

Like document-level sentiment classification, the representation of sentences produced by neural networks is crucial for sentence-level sentiment classification. Given that sentences are generally shorter than documents, incorporating syntactic and semantic information (e.g., parse trees, opinion lexicons, and part-of-speech tags) can be beneficial.

IV. Methodology

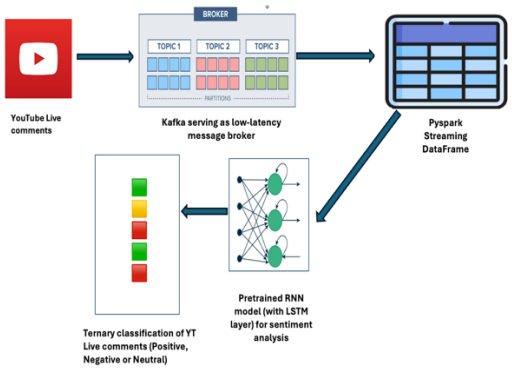
In this project, we manipulate and move data through the series of broad steps mentioned below:

1. Youtube live chat data is fetched using a python API

2. The fetched data is punched into Kafka, where the latter serves as a low-latency message broker

3. The data in the Kafka topic from step 2 is fed into a streaming dataframe using structured streaming library in the Pyspark framework

4. Each record in the streaming dataframe is fed into a deep learning model which classifies it into one of the three output classes (Positive, Negative or Neutral)



RNN with an LSTM layer:

The Recurrent Neural Network based model employs an LSTM layer. It is important to note that just one layer of the RNN model employs LSTM units and not the whole network. It was observed in our trials that this change led to a substantial improvement in the model's training and testing accuracies. Our empirical observations showed that this change increased the model training time, but this came with considerable increase in the model's accuracy.

Standard LSTM RNN architectures comprise an input layer, a recurrent LSTM layer, and an output layer. The recurrent LSTM layer features connections directly linking cell output units to input units, input gates, output gates, and forget gates. The model's parameter count can be expressed as

$$N = nc \times nc \times 4 + ni \times nc \times 4 + nc \times no + nc \times 3,$$

where nc is the number of memory cells, ni is the number of input units, and no is the number of output units.

This configuration, particularly for tasks with numerous output units and memory cells, can lead to computational challenges due to its complexity.

To address these challenges, the Long Short-Term Memory Projected (LSTMP) architecture was introduced. This model incorporates a linear projection layer after the LSTM layer, with recurrent connections now stemming from this projection layer. This setup significantly reduces the number of parameters by the ratio nr/nc, where 'nr' represents the units in the recurrent projection layer. Consequently, it allows for increasing the model's memory size while maintaining manageable parameter numbers in the recurrent connections and output layer.

Expanding on this concept, the deep LSTMP architecture involves stacking multiple LSTM layers, each with its own recurrent projection layer. This configuration enhances the model's memory capacity independently of the output layer and recurrent connections.

V. Results and Analysis

a. Model Results

Experiment Number	Parameters Chosen	Results
1	**Recurrent Neural Net: Number of Layers:** 3 Embedding(10000, 64) SimpleRNN(32) Dense(1) **Dropout Parameters:** SimpleRNN: dropout=0.2	**Train/Validation/Test:** 70/10/20 **Training Accuracy:** 15.90% **Testing Accuracy:** 8.22% **Training RMSE:** 0.85 **Test RMSE:** 0.91
2	**Recurrent Neural Net: Number of Layers:** 4 Embedding(10000, 64) SimpleRNN(128) SimpleRNN(64) Dense(1) **Dropout Parameters:** Default value	**Train/Validation/Test:** 70/10/20 **Training Accuracy:** 48.55% **Testing Accuracy:** 24.26% **Training RMSE:** 0.65 **Test RMSE:** 0.77
3	**RNN with LSTM: Number of Layers:** 5 Embedding(5000, 100) SpatialDropout1D(0.2) SimpleRNN(32) LSTM(100) Dense(1) **Dropout Parameters:** SpatialDropout1D=0.2 SimpleRNN: dropout=0.2, recurrent_dropout=0.2 LSTM: dropout=0.2, recurrent_dropout=0.2	**Testing Accuracy:** 82.17% **Training RMSE:** 0.33 **Test RMSE:** 0.35

We have trained three different models with the dataset. The first model is just a plain RNN that has 3 layers: Embedding, SimpleRNN and a Dense layer. This model has a low training and testing accuracy. This indicates that it struggles to learn the pattern in the data The high RMSE values for both the training and testing show that the model's predictions are significantly off from the actual values. Just the SimpleRNN layer with a dropout of 0.2 might not be sufficient to capture the complexity of the sentiment.

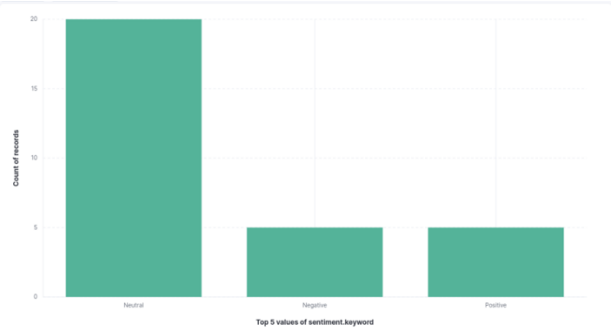
The second experiment we looked at was adding an additional RNN layer to the existing one to see if it would make any significant changes. Although, adding more layers and increasing the size of the SimpleRNN layers improved the model's performance, having a training accuracy of 48.55% and a testing accuracy of 24.26% is still not optimal. There is a noticeable drop in

accuracy from training to testing, which could be due to some overfitting. The RMSE values are lower than in the first experiment but the model as a whole is still not ideal to use.

The third model significantly outperformed the previous two models. The combination of SimpleRNN and LSTM layers, along with spatial dropout, created a complex and robust model that captured the sentiment. The training and testing accuracies were very close and the model does well with new data. The low RMSE values further confirm that the model's predictions are close to the actual sentiment labels. Based on these results, the RNN with the LSTM model is the most effective for sentiment analysis since it had high accuracy and low prediction error.

b. Kafka and ELK Results

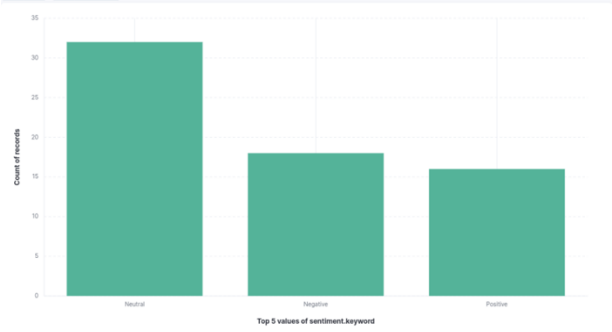
We integrated Kafka for real-time data streaming and the deep learning model for sentiment analysis to provide insights into sentiment dynamics of YouTube live streaming comments. The data shows a trend of increasing viewer engagement over time and keeps a running count of the positive, negative and neutral feedback.



First Interval Analysis:

- Neutral Comments: The highest number of comments fall into the neutral category, with a count of 20.
- Negative Comments: There are fewer negative comments, with a count of approximately 4.
- Positive Comments: There are a similar number of positive comments as negative, also around 4.

In the first interval, most comments are neutral and indicate that viewers are either providing non-opinionated feedback or general statements. The presence of both negative and positive comments suggests a mixed reaction, though both are significantly less frequent than neutral comments.

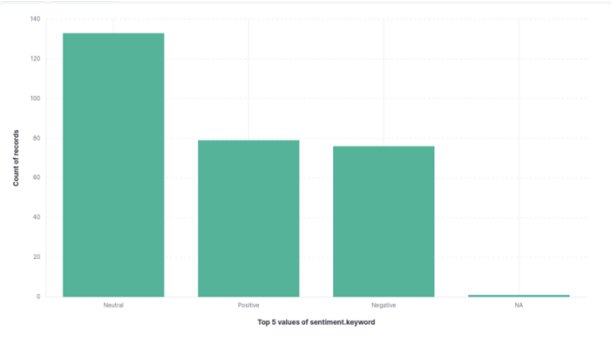


Second Interval Analysis:

- Neutral Comments: The neutral comments count remains the highest, but with a slight decrease to around 30.
- Negative Comments: Negative comments have increased, with a count of approximately 15.
- Positive Comments: Positive comments have also increased, with a count of around 20.

In the second interval, while neutral comments are still the most frequent, there is a noticeable increase in both negative and

positive comments. This suggests that viewers are becoming more expressive and opinionated as the live stream progresses. The rise in positive comments might indicate increasing engagement or appreciation, while the increase in negative comments could reflect growing dissatisfaction or criticism.



Third Interval Analysis:

- Neutral Comments: Neutral comments remain the highest, with a count of approximately 130.
- Positive Comments: Positive comments have increased significantly, with a count of around 80.
- Negative Comments: Negative comments also increased, with a count of around 70.
- NA (Not Available): A small number of comments are categorized as NA, indicating either unrecognized sentiment or a processing issue.

In the third interval, the volume of comments has increased substantially. Neutral comments are still predominant, but the significant rise in both positive and negative comments suggests heightened viewer engagement. The higher count of positive comments could indicate a favorable reception to the content during this period, while the increase in negative comments shows ongoing critical feedback.

VI. Conclusion and Future Work

The introduction of deep learning models, particularly Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and Convolutional Neural Networks (CNNs), marked a substantial leap in sentiment analysis capabilities. The development of transformer architecture-based models like BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), and their successors, revolutionized sentiment analysis.

The practice of pre-training language models on large corpora and fine-tuning them on specific tasks has also led to significant improvements. This application of Transfer Learning can help with real-time information processing and decision making with a massive impact across a wide range of business and other use cases. As evidenced by the above mentioned advancements and their impact on real-time text-based sentiment analysis, the future of such analysis holds significant potential for advancing both the field of natural language processing (NLP) and practical applications across industries.

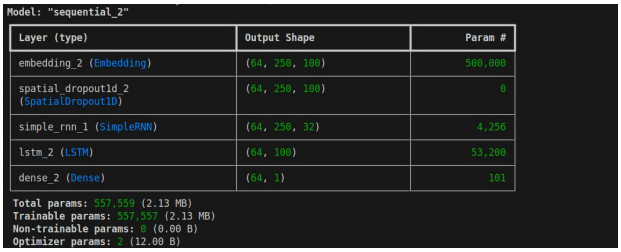
As streaming data becomes more prevalent, there will be a growing need for real-time sentiment analysis to provide immediate insights. This is crucial for applications like customer service, brand monitoring, and financial markets, where timely reactions to sentiment shifts can offer

a competitive advantage. Models like BERT and GPT will continue to improve the accuracy of sentiment analysis. These models can capture contextual nuances better, allowing for more precise sentiment detection in dynamic, streaming environments.

Combining sentiment analysis with other forms of data, such as behavioral data, demographic information, and geographical data, will lead to more comprehensive and actionable insights. For instance, integrating sentiment analysis with real-time sales data could help in adjusting marketing strategies on the fly.

Future sentiment analysis systems will likely incorporate adaptive algorithms capable of learning and adjusting to new language trends, slang, and emerging topics in real-time. This adaptability will help maintain the relevance and accuracy of sentiment analysis despite the evolving nature of language.

Advances in distributed computing and cloud technologies will enhance the scalability and efficiency of sentiment analysis on large-scale streaming data. Techniques such as edge computing could enable real-time processing and analysis of data close to its source, reducing latency and improving responsiveness.



Model: "sequential_2"

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(64, 250, 100)	500,000
spatial_dropout1d_2 (SpatialDropout1D)	(64, 250, 100)	0
simple_rnn_1 (SimpleRNN)	(64, 250, 32)	4,256
lstm_2 (LSTM)	(64, 100)	53,200
dense_2 (Dense)	(64, 1)	101

Total params: 557,559 (2.13 MB)
Trainable params: 557,557 (2.13 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2 (12.00 B)

VII. References

Deep Learning for Sentiment Analysis: A Survey - Lei Zhang et.al https://arxiv.org/vc/arxiv/papers/1801/1801.07883v1.pdf

Distributed Real-Time Sentiment Analysis for Big Data Social Streams - Amir Hossein et.al 2019 https://arxiv.org/pdf/1612.08543

A Survey on Networked Data Streaming With Apache Kafka https://www.researchgate.net/publication/373025500_A_Survey_on_Networked_Data_Streaming_with_Apache_Kafka

Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling - Hasim Sak et.al https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43905.pdf