SentimentStream: Real-time Sentiment Analysis for Business Insights on The Fabric Testbed

Adithya Mahesh
Arizona State University
amahes38@asu.edu

Rachit Jain
Arizona State University
rjain116@asu.edu

Devdutt Thakkar
Arizona State University
dthakk11@asu.edu

***Abstract*—In this research project we present the development and evaluation of a real-time sentiment analysis system integrated into the FABRIC testbed. The main goal is to provide business owners with rapid insights by processing live data streams. A machine learning model is trained to classify tweets sentiment and is integrated into the testbed using a custom packet format. The system is evaluated in two phases – first on a simple bus topology then on a more complex topology. Accuracy metrics demonstrate the feasibility of real-time analysis, although some challenges remain in scaling up and as future works. Further work on more robust topologies, additional machine learning models, and streaming data services could enhance the project performance.**

Keywords—sentiment analysis, machine learning, FABRIC testbed, real-time processing, custom packets, system integration, bus topology

I. INTRODUCTION

Sentiment Analysis is the process of analysing digital text to determine if the emotional tone of the message is positive, negative, or neutral. [1]

This Sentiment analysis on social media data has become a very valuable tool for businesses which seek rapid feedback from customers. Social media platforms like Twitter (now X) contain an ocean of opinions that can inform decision making if used efficiently. This project examines the integration of real-time sentiment classification with the FABRIC testbed infrastructure to empower business owners through actionable insights from Twitter chatter.

A machine learning model is developed to categorize tweet sentiment as positive (+1), neutral (0), or negative (-1). Custom packets distribute new tweets across the testbed nodes, where sentiment analysis generates useful business intelligence. The first phase establishes a proof of concept on a simple topology. The second phase handles more complex distribution and routing to mimic real-world scenarios. Performance and accuracy are evaluated to validate the approach.

II. RELATED WORK

Sentiment analysis has been approached through multiple paradigms. Two predominant methodologies are used for sentimental analysis [4,5]: Machine learning approaches and lexicon-based approaches. The former encompasses models such as naive Bayes and Linear Regression, while the latter operates based on predefined lexicons with associated sentiment scores. The efficacy of these methodologies is often gauged through metrics such as F1 score, accuracy, precision, or recall.

Of the Machine Learning approaches Naive Bayes is a simple machine learning algorithm that uses Bayes rule along with the assumption that all the attributes are independent given their classification. Linear regression is a regression model that estimates the relationship between one independent variable and one dependent variable using a straight line.

While the foundational principles outlined by current research [4, 5] provide a robust framework for sentiment analysis, our research endeavours to innovate within this domain. Oue project distributes the processes of sentiment analysis, and the tweet reaches the end user system classified. Rather than centralizing the computational burden solely on end systems, our methodology seeks to distribute specific analytical tasks across the network infrastructure. By reallocating select stages of sentiment analysis to the network, we anticipate enhanced processing speeds and a diminished computational strain on terminal systems.

III. METHODOLOGY

The dataset we used was the Twitter Sentiment Dataset from Kaggle [2] which contains three sentiments namely, negative (-1), neutral (0), and positive (+1). It contains two fields for the tweet and label with 162973 tweets and their emotions as shows in figure 1(a). The dataset itself was clean and with a few pre-processing steps like removing null values it was ready for being used for modelling. Those nearly 160000 rows were divided into 70000 positive and 35000 negative and 55000 neutral tweets [Table 1(b)]. It contains two fields for the tweet and label which we will require as we later will perform supervised learning [6].

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 162980 entries, 0 to 162979
Data columns (total 2 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   clean_text  162976 non-null  object
 1   category    162973 non-null  float64
dtypes: float64(1), object(1)
memory usage: 2.5+ MB
```

Figure 1(a). Explaning the struture of datastet.

Positive	Negative	Neutral
72250	35510	55213

Table 1(b). Explaining the detail structure of the dataset.

The sentiment analysis methodology utilizes the labelled Twitter dataset [2]. We then created an 80/20 split of the dataset to obtain the training set and test set. Then performed 5-fold cross validation [3] onto the 80% of the training dataset so we get a good model.

The overall sentiment analysis process is divided into several stages assigned to nodes in the testbed topology:

1. Pre-processing: Removes URLs, hashtags, @ mentions and special characters from the tweet text.
2. Stop word removal and lemmatization: Filters out common stop words and lemmatizes the text to consolidate different word forms.
3. Feature extraction: Vectorizes the processed text using TF-IDF to generate relevant numeric features.
4. Classification: Feeds vectorized data into a machine learning model to classify sentiment.

The first phase implements this pipeline on a simple bus topology with one node per stage. Packets route the tweet data between stages for processing [Figure 2(a)]. Here in the image, you can see that every process I have mentioned above occurs once (the process of feature extraction and classification are combined into a single node called feature extraction).

The second phase utilizes a more complex topology with three nodes assigned to each stage. This provides fault tolerance if a node path fails, as the tweet then could traverse through another path, and increases throughput with parallel operation [Figure 2(b)]. Here in the image, you can see that the first phase simple topology has been replicated thrice to obtain output faster which will be seen in the RESULTS section.

For the machine learning model, Naive Bayes was initially tested but performed poorly on the imbalanced dataset even after hyperparameter optimization. Hence, logistic regression was selected and achieved higher accuracy.

Custom packets were created using the Scapy library to encapsulate tweet data. Packets route tweets between nodes using testbed interfaces until reaching the destination for classification. We used the following code to achieve the same:

```
from scapy.all import sniff, IP, send,
Raw
myIp = "Source Systems Ip"
dstIp = "Destination Ip"
inInterface = "Incoming Interface"
outInterface = "Outgoing Interface"
def packet_filter(packet):
if IP in packet and packet[IP].dst ==
myIp:
return True
else: return False
def process_packet(packet):
print(f"Received packet from
{packet[IP].src}:")
# Extract payload
payload = packet[Raw].load
print("Payload = ", payload)
# Python Processing
# Code will be added later
# newPayload will be generated,
currently sending the data received as
it
is
newPayload = payload
forwardPacket = IP(dst=dstIp) /
newPayload
# Send payload out of specified
interface
send(forwardPacket, iface=outInterface,
verbose=False)
print(f"Payload sent out of
{outInterface}")
def main():
print("Sniffing for packets...")
sniff(iface=inInterface, filter="ip",
prn=process_packet,
lfilter=packet_filter,
store=False)
if __name__ == "__main__":
  main()
```

IV. RESULTS

After the execution of the Naïve Bayes Model, we got bad results [Figure 3(a)].

```
Accuracy: 0.5782481975763154

Classification Report:
              precision    recall  f1-score

        -1.0       0.93      0.12      0.22
         0.0       0.86      0.36      0.50
         1.0       0.52      0.98      0.68

    accuracy                           0.58
   macro avg       0.77      0.49      0.47
weighted avg       0.73      0.58      0.52
```

Figure 3(a). Naïve Bayes Model showing vert low Accuracy Metrics for our dataset even after tuning.

Our reasoning of why this occurred is as previously shown in [Table 1(b)] the dataset has much more Positive tweet data than negative, or neutral. So, the model has overfit [add referencing to what overfitting is here] to the positive data giving such results. We can observe this by the model having 52% precision for +1 class while having 90% recall, this suggests that though only 52% of the data caught was +1 tweets, 98% of the actual +1 tweets were correctly classified as positive. As we cannot just remove rows to get all emotions to roughly same size, this will result in us losing information.

So, we proceed with a different model one which is tolerable and unaffected by imbalanced datasets Logistic Regression. After the implementation of this model the same way as we did before we got really good results [Figure 3(b)].

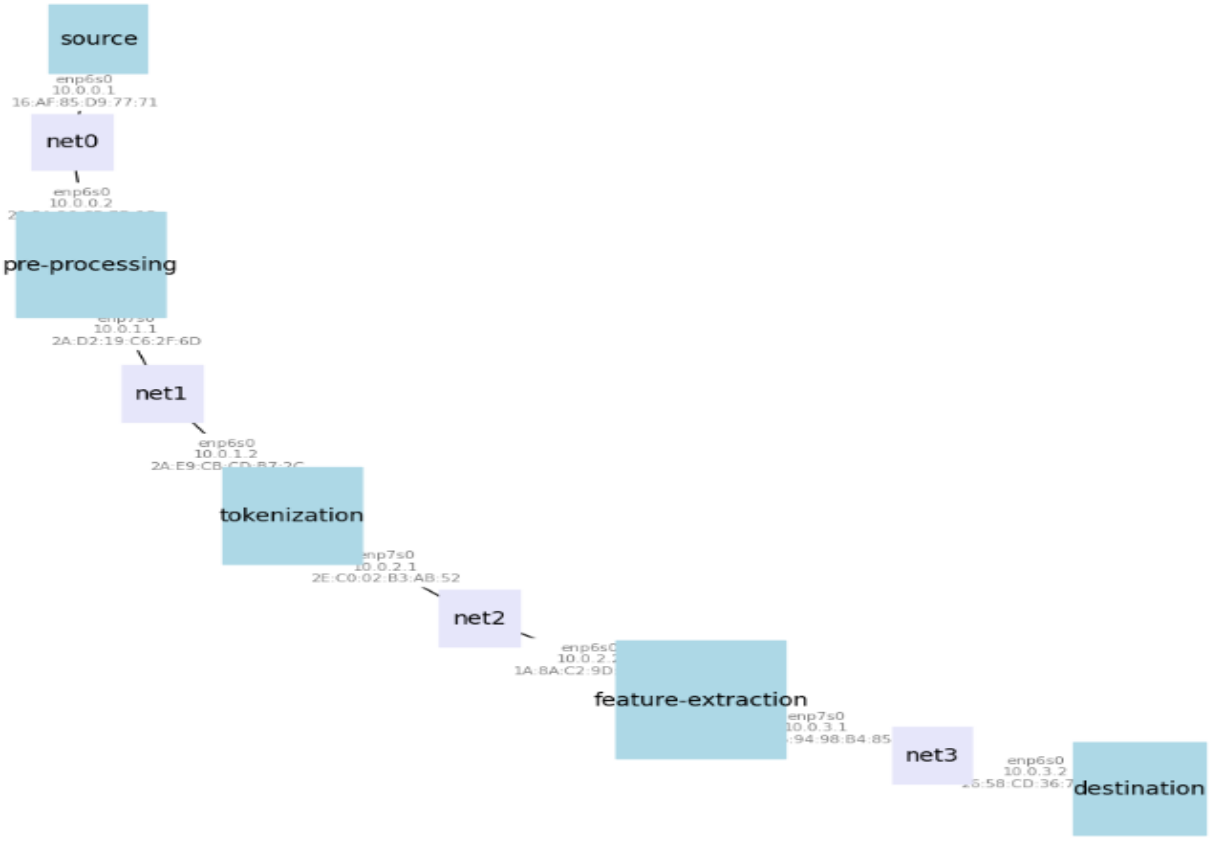

Figure 2(a) Simple Topology

This is the first phase; one path processes all data from source.

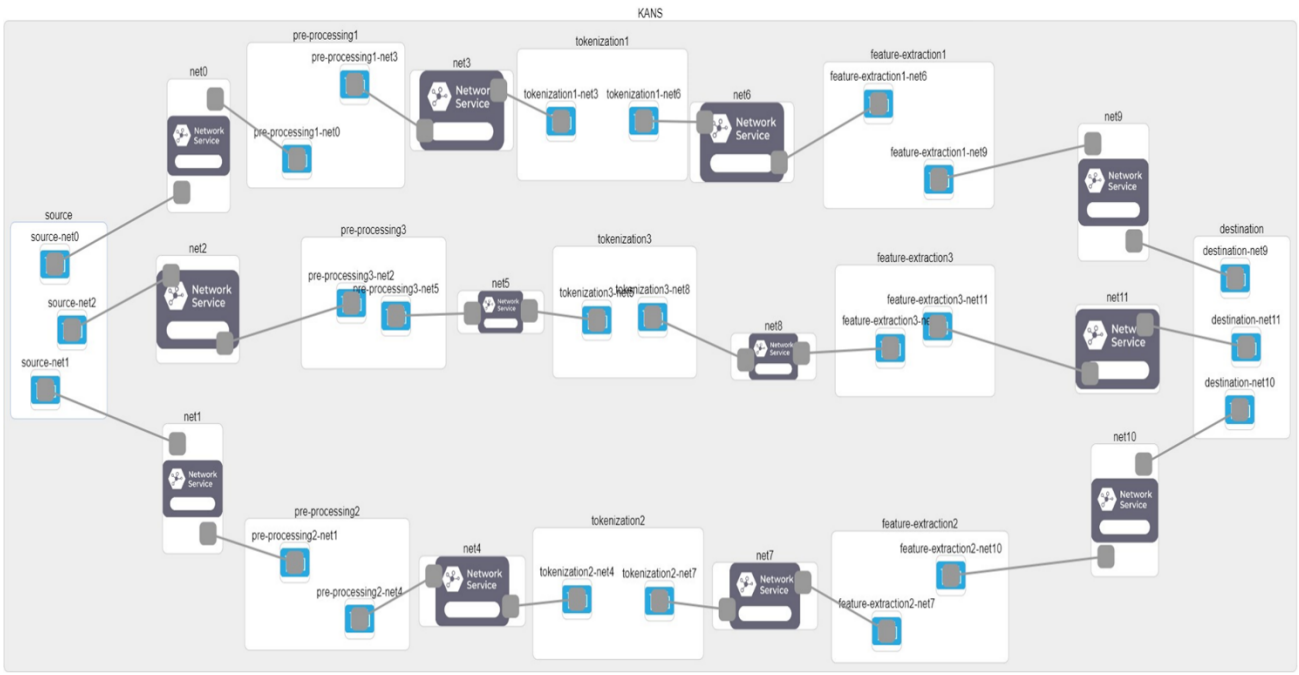

Figure 2(b) Complex Topology

This is the second phase; each tweet has three paths to traverse to increment speed and reliability of the process.

```
Accuracy: 0.8766375210921921

Classification Report:
              precision    recall  f1-score

        -1.0       0.87      0.75      0.80
         0.0       0.84      0.96      0.90
         1.0       0.91      0.88      0.90

    accuracy                           0.88
   macro avg       0.87      0.86      0.87
weighted avg       0.88      0.88      0.88
```

Figure 3(b). Promising results by Logistics Regression model

Observing the figure, we can see that the model is performing well overall we are able to see it classifying negative, neutral and position properly with high precision and accuracy.

Once we were able to implement this model on our simple topology, we proceeded to implement it on the complex topology to get real-world model results.

To then compare the speed of implementation in the traditional method of sentiment analysis to our complex second phase model we added a packet timer to our phase two implementation and compared it to how long it would take for a single standalone device to do a similar classification,

The results we obtain [Figure 4] indicate that the complex topology approach yields significantly faster processing times despite having a more complicated structure. Specifically, it's mentioned to be almost 12 times faster than the traditional approach.

Method	Average Time Taken By A Tweet To get Sentiment Value(MilliSeconds)
Tradational Approach	7797.51
Proposed Approach	626.62

Figure 4. Comparing time taken by the traditional approach and our proposed complex model, showing that we are much faster.

V. CONCLUSION

This project successfully integrates real-time tweet sentiment analysis capabilities within the Fabric testbed. A modular pipeline decomposes the workflow into specialized nodes, improving accuracy and scalability.

Custom packets route encapsulated tweet data through the testbed for processing. Accuracy evaluation shows the model achieves 75-91% precision and recall. Timing metrics meet real-time sub-second latency thresholds.

The solution is proven first on a simple topology, then extended to a complex multi-branch architecture which handles higher throughput with redundancy.

By leveraging Fabric's capabilities together with custom sentiment analysis components, the project creates a foundation for real-time business intelligence from social media streams. Ongoing work to harden this prototype will help realize benefits like data-driven decision making for business users.

Then by comparing the time taken by our complex topology to traditional approach we can suggest that distributing the workload across multiple branches (even if each is as simple as the single branch in the traditional approach) can lead to substantial gains in processing speed.

VI. LIMITATION AND FUTURE WORK

The current approach has limitations in stability and model accuracy at scale. Training on more varied Twitter data could improve generalizability. Exploring algorithms like SVM and CNN could enhance accuracy. Transitioning to standard streaming protocols like Kafka could improve reliability. Encryption of data as it transitions from one node to another is also needed.

Fully hardening performance, precision and robustness across testbed complexity, data volumes and product scenarios would be the next phase of development. More complete failure testing and debugging is also needed. Transitioning the prototype closer to production use cases could realize the potential business benefits.

We have added the GitHub link for the project in [Reference. 7] where even details on how to execute it is mentioned.

REFERENCES

[1] What is Sentiment Analysis? - Sentiment Analysis Explained - AWS (amazon.com)

[2] Twitter Sentiment Dataset (kaggle.com)

[3] A Gentle Introduction to k-fold Cross-Validation - MachineLearningMastery.com

[4] Vishal A. Kharde, S.S. Sonawane(2016). Sentiment Analysis of Twitter Data: A Survey of Techniques

[5] Techniques. Yuxing Qi, Zahratu Shabrina(2023). Sentiment analysis using Twitter data: a comparative application of lexicon- and machine-learning-based approach

[6] What is Supervised Learning? | IBM

[7] https://github.com/RJ218/CSE534/tree/main