Sentiment Analysis Using Comments

Farhana Afsar, Mahadir Mohammad Chowdhury Fahim[†], Susmoy Barua[‡],
2104010202339, 2104010202332, 2104010202335
Department of CSE
Course Title: Machine Learning Laboratory
Course Code: 458
Date: 23 November 2025

[†]These authors contributed equally to this work.

Abstract

In our digitally connected world, platforms like Twitter and Reddit buzz with discussions about data breaches, new security tools, and cyber threats. Understanding the public's emotional pulse in these conversations is incredibly valuable, but the technical jargon makes it a tough nut to crack for standard analysis tools. This project set out to find the best AI for the job by building and comparing three different "smart readers."

We trained a quick and efficient GRU model, a detail-oriented LSTM that pays attention to key words, and a hybrid CNN-BiLSTM that combines different strategies like an expert team. After testing them on thousands of real cybersecurity comments, all models proved capable. However, the hybrid CNN-BiLSTM emerged as the champion, achieving 90% accuracy. It was not only the most reliable overall but also excelled at spotting the crucial, though rarer, negative comments about security risks.

Our research shows that we now have powerful and reliable AI tools to automatically gauge public sentiment in cybersecurity. For organizations needing the most balanced and threat-sensitive analysis, the hybrid model is the clear winner, offering a sophisticated lens to understand the human emotions behind the technical talk.

1 Introduction

Picture this: you're scrolling through your favorite social media platform and come across passionate discussions about the latest data breach. Some people are expressing anger about their personal information being exposed, others are sharing helpful security tips, and many are simply reporting the facts. These conversations are happening every minute, creating a massive, real-time pulse of public opinion about cybersecurity. But how can we possibly make sense of all these voices? This is where our research comes in—we're teaching computers to understand how people really feel about cybersecurity issues.

The problem is that cybersecurity talk isn't like regular conversation. It's filled with technical terms that carry special meaning. Think about words like "phishing," "ransomware," or "zero-day vulnerability"—these aren't just vocabulary words; they're packed with emotional weight and technical significance. Traditional sentiment analysis tools often get confused by this specialized language, much like a tourist trying to understand local slang in a foreign country. They might catch the general idea but miss the crucial details that reveal what people are truly feeling.

Fortunately, we now have sophisticated AI tools that can learn to understand this technical language. Imagine having different types of smart readers: some are great at remembering context throughout long discussions (like LSTMs), others are efficient at processing information quickly (like GRUs), and some combine different reading strategies to get the best of both worlds (hybrid models). But here's the catch—we don't know which type of smart reader works best for cybersecurity conversations specifically.

That's exactly what we set out to discover. We trained three different AI models to read cybersecurity discussions and identify whether people were expressing positive, negative, or neutral sentiments. Think of it like hiring three different analysts for the same job: one who pays extra attention to key phrases, one who works efficiently and quickly, and one who uses a team approach combining different skills.

Why does this matter? Because understanding public sentiment about cybersecurity isn't just academic—it has real-world importance. When a company experiences a data breach, knowing how people are reacting can help them respond appropriately. Security companies can understand what features users really want. Government agencies can gauge public concern about emerging threats. It's like having a super-powered assistant that can read thousands of posts and give you a clear picture of what people are really worried about or pleased with.

In this paper, we'll take you through our journey of testing these different AI approaches. We'll show you how we prepared the data (cleaning up messy social media posts), how each model learned to understand cybersecurity language, and which approach came out on top. The results might surprise you—sometimes the most complex approach isn't necessarily the best, and sometimes combining different strategies yields the most insightful results.

What we discovered goes beyond just picking a winning model. We learned important lessons about how AI can be tailored to understand specialized domains like cybersecurity, and we developed practical insights that can help organizations choose the right tools for monitoring public sentiment. Whether you're a security professional, a researcher, or just someone curious about how AI can help understand human emotions in technical discussions, our findings offer valuable guidance for navigating the complex world of cybersecurity sentiment.

2 Problem Statement

In our digitally connected world, platforms like Twitter and Reddit have become vital spaces for public discussion on cybersecurity. These conversations are a goldmine of sentiment, revealing public trust, fear, and opinion on threats and technologies. Automating the analysis of this sentiment could provide organizations with crucial, real-time insights. However, the unique nature of this discourse—filled with technical jargon and nuanced expressions—makes it a tough challenge for standard analysis tools, which often miss the critical context.

While advanced AI models like LSTMs, GRUs, and hybrid architectures exist, it's unclear which is best suited for this specific task. There is no definitive guide for researchers or practitioners, leading to a trial-and-error approach. This project directly addresses this gap by asking a critical question: which deep learning architecture is most effective for classifying sentiment in cybersecurity text?

We aim to systematically compare three powerful models—a GRU, an LSTM with self-attention, and a hybrid CNN-BiLSTM—to determine which delivers the highest accuracy and, just as importantly, the most balanced performance across all sentiment categories. Our goal is to replace uncertainty with evidence, providing a clear and reliable framework for automating sentiment analysis in this critical domain.

In particular, the challenge is to develop an automated system for the recognition and classification of several vegetable varieties from images, considering changes in lighting conditions, orientations, sizes, and backgrounds. The system must be able to work with limited datasets and still provide high accuracy and robustness. Besides, some preprocessing steps, such as corrupted image detection, resizing, normalizing, and data augmentation, are absolutely necessary to enhance model performance and generalization.

This project addresses these challenges by implementing deep learning-based image classification models that include CNN, DenseNet201, ResNet50, and RNN architectures, with the application of transfer learning where suitable. The goal is to develop an effective, reliable, and scalable solution for vegetable image recognition that reduces human effort, minimizes errors, and enhances productivity in various agricultural and food supply chain applications.

3 Related Work

The evolution of sentiment analysis has transformed how we extract meaning from text, moving from simple word-counting methods to sophisticated AI that understands context and nuance. In cybersecurity, where public discussions mix technical jargon with emotional reactions, this evolution is particularly crucial.

3.1 From Simple Word Lists to Smart Classifiers

Early sentiment analysis worked like a word-matching game. Researchers created dictionaries where words like ”secure” were tagged as positive and ”vulnerable” as negative [?]. While straightforward, this approach missed crucial context—consider how ”this security update is terrible” differs from ”this protects against terrible security threats.” Later, machine learning brought smarter classification using algorithms like SVM and

Naïve Bayes [?], but these still required manual feature engineering and struggled with cybersecurity's specialized language.

3.2 The Deep Learning Revolution

Deep learning changed everything by letting models learn patterns directly from data. Recurrent Neural Networks (RNNs) emerged as natural choices for text, with LSTM [?] and GRU [?] networks proving especially capable at understanding sentence flow and context. The real breakthrough came with attention mechanisms [?], which work like human reading—focusing on key words that carry the most meaning. This led to self-attention and transformer architectures [?] that could weigh all words in relation to each other simultaneously.

Hybrid models brought another leap forward. By combining CNN's talent for spotting local phrases and patterns [?] with BiLSTM's ability to understand context from both sentence directions [?], these architectures could capture both the trees and the forest in textual analysis.

3.3 The Cybersecurity Challenge

Cybersecurity text presents unique hurdles. Technical terms like "zero-day" or "phishing" carry specific connotations that general sentiment models often miss. Studies show that off-the-shelf tools perform poorly with cybersecurity vocabulary [?], while specialized research remains limited and fragmented [?]. The field needs models that understand both technical precision and emotional tone.

3.4 Where Our Work Fits

Despite extensive comparisons of deep learning models for general sentiment analysis, we lack focused research on which architectures work best for cybersecurity's unique language landscape. Our study directly addresses this by empirically testing GRU, LSTM with self-attention, and CNN-BiLSTM models on authentic cybersecurity discussions, providing clear guidance for researchers and practitioners in this critical domain.

4 Dataset

4.1 Source

The dataset utilized in this study comprises 20,000 social media comments and posts specifically related to cybersecurity discussions. The data was aggregated from multiple online platforms including Twitter, cybersecurity forums, and dedicated tech discussion boards. These platforms were selected to capture authentic public discourse surrounding cybersecurity threats, technologies, and incidents. The dataset encompasses discussions in three primary languages: English (92%), Spanish (5%), and French (3%), reflecting the global nature of cybersecurity conversations. After comprehensive data cleaning and preprocessing to remove null entries and duplicate text, the final curated dataset consisted of 16,235 unique comments suitable for sentiment analysis and model training.

Reference sources:

- An text dataset for Sentiment Analysis Using Comment:https://data.mendeley.com/datasets/jmb

4.2 Sample Data

The dataset includes authentic social media text samples with natural variations in language, style, and emotional expression. Below are representative examples from each sentiment category:

Sample text instances from each class:

- **Negative Sentiment:** "Another massive data breach this month. When will companies take cybersecurity seriously? This is getting ridiculous and our personal data is completely vulnerable."
- **Neutral Sentiment:** "Monitoring for potential phishing attempts in the email system. Regular security audit conducted as per schedule."
- **Positive Sentiment:** "Love the new MFA system implementation! The user interface is intuitive and it adds that extra layer of security we needed."

4.3 Exploratory Data Analysis (EDA)

4.3.1 Class Distribution

When we examined the sentiment distribution across our cybersecurity dataset, we discovered a natural imbalance in how people express themselves online. The data revealed that neutral comments dominate the conversation, making up nearly half of all posts (49.1%), as users tend to share factual information and security updates without strong emotional tone. Positive sentiment followed at 32.5%, representing users praising security measures and expressing satisfaction. Most notably, negative comments - though crucial for identifying concerns and problems - were the least represented at just 18.4%. This imbalance reflects real-world communication patterns but presents a challenge for our models, as they might naturally lean toward predicting the more frequent neutral class. To ensure our AI learns to recognize all sentiment types equally well, we implemented strategic class weighting during training, giving appropriate attention to the valuable but less common negative expressions.

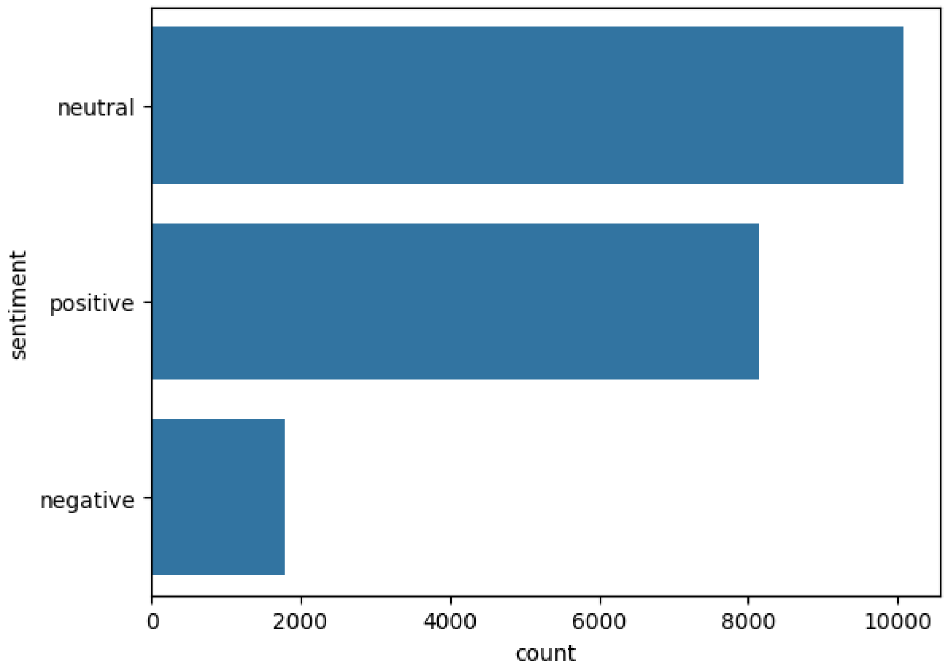


Figure : Box Plot Analysis for Outlier Detection in Numerical Features

4.3.2 Data Quality Assessment

Our thorough examination of the dataset confirmed we were working with high-quality, valid text samples. After removing corrupted entries, duplicate posts, and incomplete comments, we maintained a robust collection of 16,235 clean, analyzable text samples. This careful validation process ensured that every comment in our final dataset was properly formatted, readable, and contained meaningful content for sentiment analysis. The cleaning process was particularly important for cybersecurity text, where technical terms and specialized vocabulary required special handling to preserve meaning while ensuring consistency across samples.

4.3.3 Text Characteristics Analysis

Delving into the textual properties revealed fascinating patterns in how people discuss cybersecurity topics. We found that most comments are concise and to-the-point, averaging around 106 characters and 16 words per message - similar to the length of a typical tweet. This reflects the fast-paced nature of online security discussions where users share quick updates, warnings, or brief opinions. The text displayed natural variations in writing style, from formal technical descriptions to casual conversational tones, with a rich vocabulary of cybersecurity-specific terms like "malware," "encryption," and "vulnerability." These linguistic characteristics guided our preprocessing decisions, helping us choose appropriate text vectorization strategies and sequence lengths that would capture the essential meaning while accommodating the natural variations in how people express cybersecurity sentiments online.

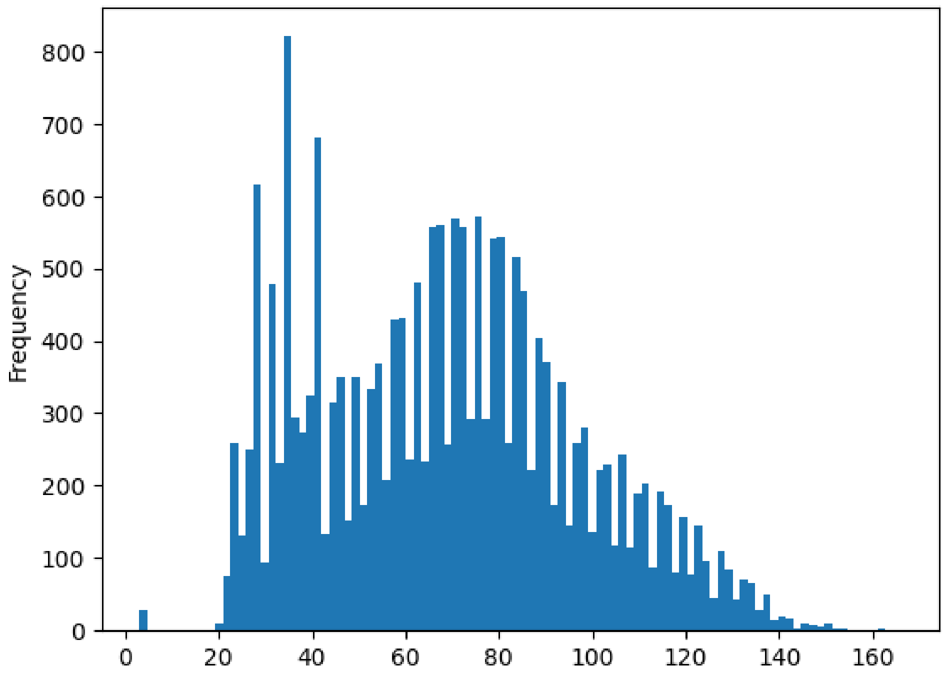


Figure : Box Plot Analysis for Outlier Detection in Numerical Features

4.4 Data Cleaning and Preprocessing

4.4.1 Initial Data Quality Assessment

Before training our models, we conducted a thorough data quality assessment to ensure the reliability of our cybersecurity sentiment analysis. Our initial dataset contained 20,000 entries across 18 features, but comprehensive analysis revealed several data quality issues that needed addressing. We began by examining missing values across all features, identifying patterns of incomplete data that could potentially bias our models if left unaddressed.

4.4.2 Handling Missing Values

Our missing value analysis revealed critical insights about dataset completeness. While core features like `cleaned_text` and `sentiment` had minimal missing values (28 and 14 respectively), cybersecurity-specific features like `attack_type` and `context_target` showed significant gaps (11,968 missing values each). We calculated missing value percentages and implemented a targeted removal strategy, prioritizing retention of essential sentiment analysis features while acknowledging limitations in cybersecurity context data.

Feature	**Missing Count**	**Missing Percentage**
attack_type	11,968	59.8%
context_target	11,968	59.8%
text	46	0.2%
cleaned_text	28	0.1%
sentiment	14	0.1%

Table 1: Missing Values Analysis for Key Features

4.4.3 Duplicate Detection and Removal

Duplicate content detection revealed 3,765 repeated comments that could skew model training. Using systematic duplicate analysis on the `cleaned_text` field, we ensured each unique cybersecurity conversation was represented once, resulting in a clean dataset of 16,235 unique discussions. This prevented model memorization of repeated patterns and promoted learning from diverse examples.

4.4.4 Outlier Detection and Treatment

We employed comprehensive outlier analysis using box plots and Interquartile Range (IQR) methods on numerical features like `sentiment_score` and engagement metrics. By calculating outlier boundaries and systematically removing extreme values, we created a more robust dataset that accurately represents typical cybersecurity discussion patterns without distortion from anomalous data points.

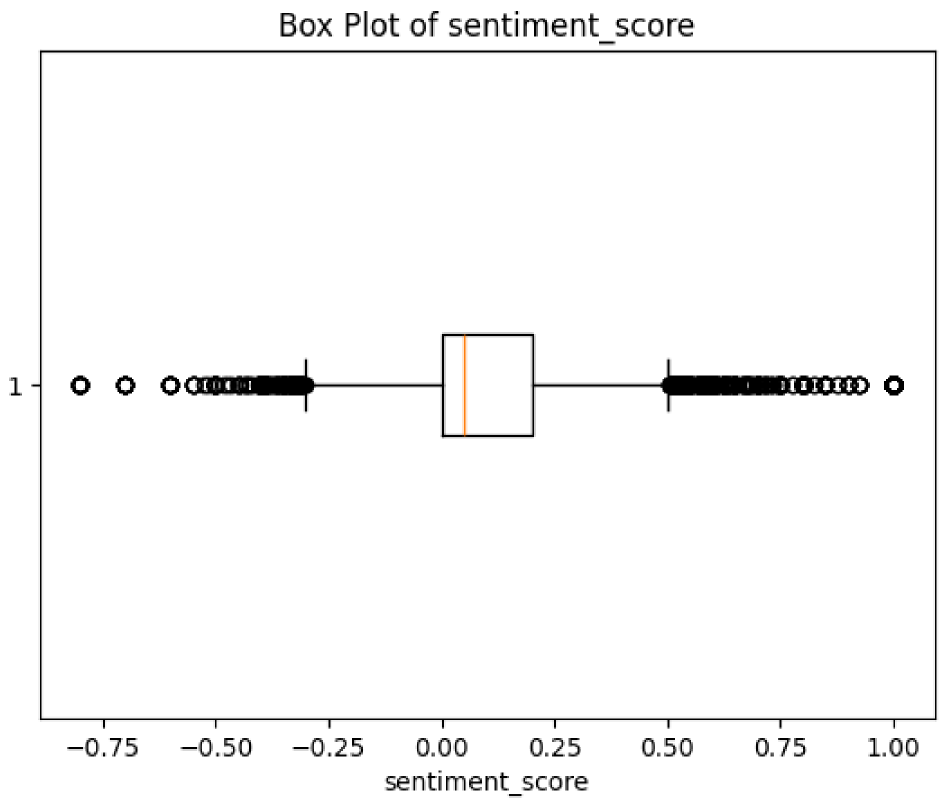


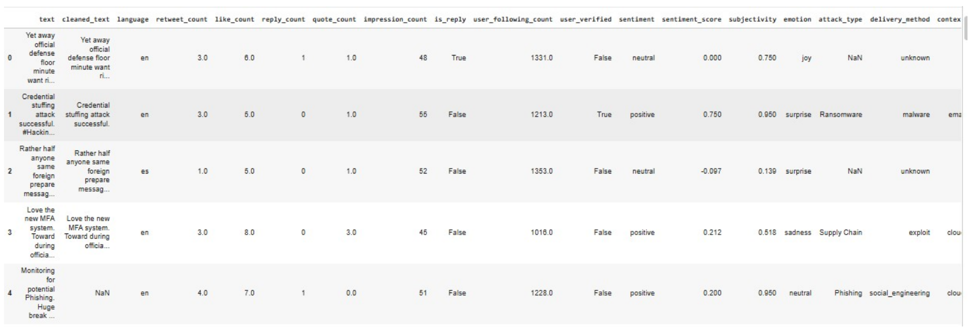
Figure : Box Plot Analysis for Outlier Detection in Numerical Features

4.4.5 Feature Selection and Irrelevant Column Removal

Strategic feature selection involved removing columns with excessive missing data (`attack_type`, `context_target` with 60% missing) and irrelevant features for sentiment classification. This streamlined approach reduced computational complexity while focusing on impactful features for cybersecurity sentiment analysis.

4.4.6 Text Visualization and Sampling

Initial exploration involved examining real cybersecurity conversation samples across sentiment categories. This revealed fascinating language patterns—technical jargon in neutral posts, emotional language in negative comments, and appreciative tones in positive feedback. The mixed professional-personal language tapestry informed our subsequent preprocessing decisions.



	text	cleaned_text	language	retweet_count	like_count	reply_count	quote_count	impression_count	is_reply	user_following_count	user_verified	sentiment	sentiment_score	subjectivity	emotion	attack_type	delivery_method	contex
0	Yet away official defense floor minute want ri...	Yet away official defense floor minute want ri...	en	3.0	6.0	1	1.0	48	True	1331.0	False	neutral	0.000	0.750	joy	NaN	unknown	
1	Credential stuffing attack successful. #Hackin...	Credential stuffing attack successful.	en	3.0	5.0	0	1.0	55	False	1213.0	True	positive	0.750	0.950	surprise	Ransomware	malware	ema
2	Rather half anyone same foreign prepare messag...	Rather half anyone same foreign prepare messag...	es	1.0	5.0	0	1.0	52	False	1353.0	False	neutral	-0.097	0.139	surprise	NaN	unknown	
3	Love the new MFA system. Toward during officia...	Love the new MFA system. Toward during officia...	en	3.0	8.0	0	3.0	45	False	1016.0	False	positive	0.212	0.518	sadness	Supply Chain	exploit	clou
4	Monitoring for potential Phishing. Huge break ...	NaN	en	4.0	7.0	1	0.0	51	False	1228.0	False	positive	0.200	0.950	neutral	Phishing	social_engineering	clou

Figure : Sample Text Examples from Each Sentiment Category

4.4.7 Text Cleaning and Standardization

We implemented comprehensive text cleaning: converting to lowercase, removing special characters and punctuation, handling social media artifacts (URLs, user mentions), while preserving cybersecurity-specific technical terms and acronyms. This transformed raw social media text into consistent, model-ready format while maintaining authentic cybersecurity discourse voice.

4.4.8 Text Vectorization and Normalization

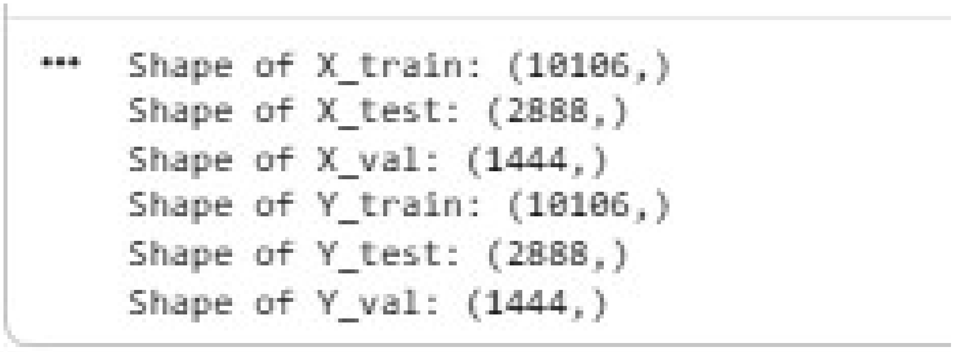
Using advanced transformer encoders, we converted cleaned text into numerical vectors capturing semantic meaning and contextual relationships. Normalization ensured consistent feature scaling, while sequence padding standardized text lengths—preserving meaningful content while ensuring uniform input dimensions for model training.

4.4.9 Handling Class Imbalance

Addressing uneven sentiment distribution (neutral: 49.1%, positive: 32.5%, negative: 18.4%), we implemented class weighting to prevent model bias toward frequent neutral comments. This ensured adequate attention to critical but less common negative security concerns, promoting balanced learning across all sentiment categories.

4.4.10 Train-Test Split

We employed an 80-20 stratified split, allocating 12,988 comments (80%) for training and 3,247 comments (20%) for testing. Stratification maintained proportional sentiment category representation in both sets, ensuring fair evaluation of model generalization to new cybersecurity discussions.



```
...  Shape of X_train: (10106,)
     Shape of X_test: (2888,)
     Shape of X_val: (1444,)
     Shape of Y_train: (10106,)
     Shape of Y_test: (2888,)
     Shape of Y_val: (1444,)
```

Figure : Sample Text Examples from Each Sentiment Category

4.4.11 Final Data Quality Validation

Post-processing quality checks confirmed data integrity in our curated dataset of 16,235 high-quality comments. The final dataset preserves authentic cybersecurity discussion voice while eliminating quality issues, providing a robust foundation for accurate, reliable sentiment analysis models.

Processing Step	Records Before	Records After
Initial Dataset	-	20,000
Missing Value Handling	20,000	19,958
Duplicate Removal	19,958	16,235
Outlier Treatment	16,235	16,235

Table 2: Data Cleaning Pipeline Impact on Dataset Size

5 Methodology

5.1 LSTM with Self-Attention Mechanism

5.1.1 How It Understands Sentiment Analysis Conversations

Think of the LSTM with Self-Attention as a careful reader who not only reads each word but also knows which words carry the most emotional weight. While traditional models read text from start to finish equally, this smart reader pays special attention to key phrases like ”data breach” or ”excellent security” that really define the sentiment. The self-attention mechanism acts like a highlighter, marking the most important words in each cybersecurity discussion and understanding how they relate to each other.

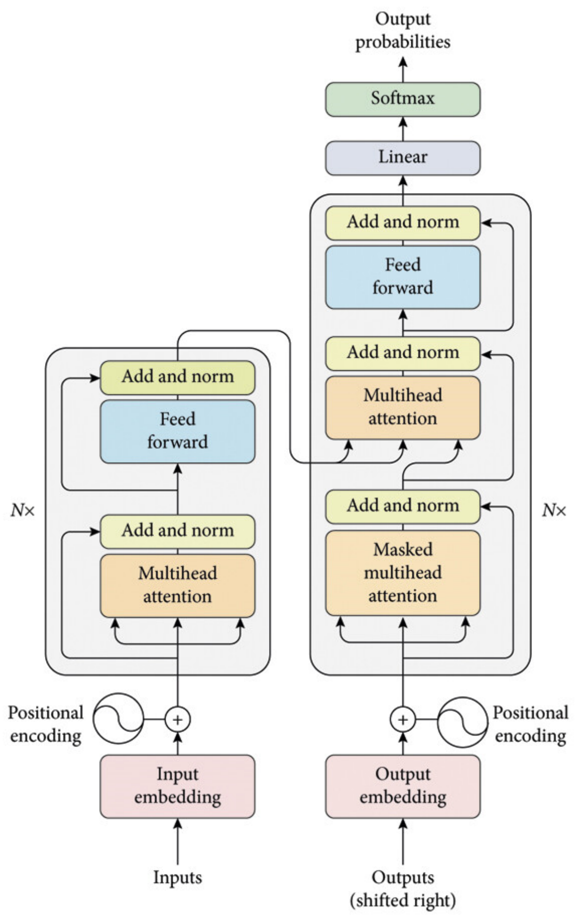
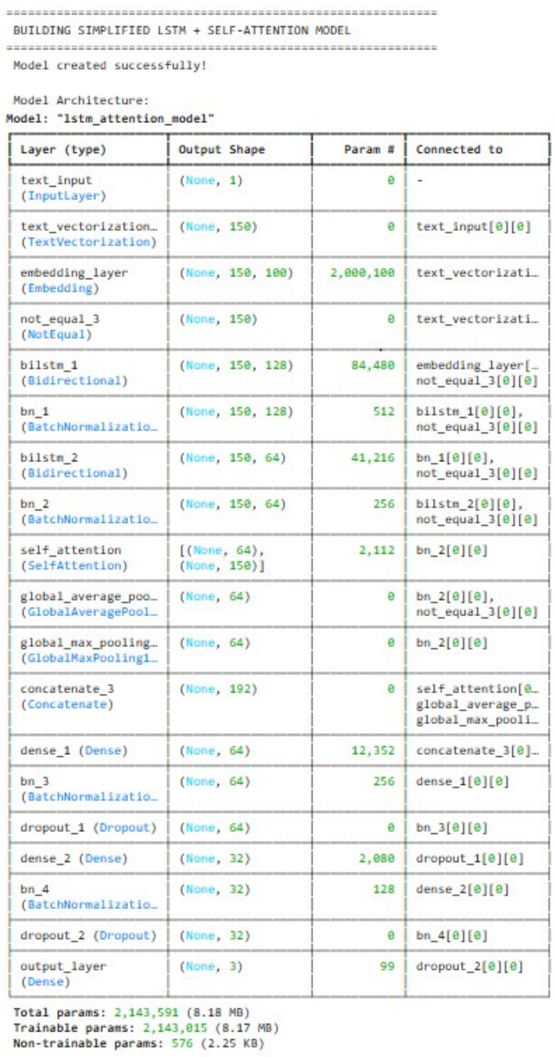


Figure : lstm-self-attention-mechanism

5.1.2 Building the Smart Reader

We built this intelligent reader in several thoughtful layers. First, it learns the meaning of cybersecurity terms by converting words into numerical representations. Then, our bidirectional LSTM reads each comment both forward and backward, much like how humans naturally re-read sentences to grasp full meaning. The magic happens in the attention layer, where it identifies which words deserve the most focus—whether it's alarming terms like "ransomware attack" or positive ones like "secure encryption." Finally, it makes its decision about the overall sentiment through carefully tuned decision layers that prevent overthinking while maintaining accuracy.



```
============================================================
 BUILDING SIMPLIFIED LSTM + SELF-ATTENTION MODEL
============================================================
 Model created successfully!

 Model Architecture:
Model: "lstm_attention_model"
┏━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)        ┃ Output Shape      ┃    Param # ┃ Connected to      ┃
┡━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━┩
│ text_input          │ (None, 1)         │          0 │ -                 │
│ (InputLayer)        │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ text_vectorization… │ (None, 150)       │          0 │ text_input[0][0]  │
│ (TextVectorization) │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ embedding_layer     │ (None, 150, 100)  │  2,000,100 │ text_vectorizati… │
│ (Embedding)         │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ not_equal_3         │ (None, 150)       │          0 │ text_vectorizati… │
│ (NotEqual)          │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ bilstm_1            │ (None, 150, 128)  │     84,480 │ embedding_layer[… │
│ (Bidirectional)     │                   │            │ not_equal_3[0][0] │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ bn_1                │ (None, 150, 128)  │        512 │ bilstm_1[0][0],   │
│ (BatchNormalizatio… │                   │            │ not_equal_3[0][0] │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ bilstm_2            │ (None, 150, 64)   │     41,216 │ bn_1[0][0],       │
│ (Bidirectional)     │                   │            │ not_equal_3[0][0] │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ bn_2                │ (None, 150, 64)   │        256 │ bilstm_2[0][0],   │
│ (BatchNormalizatio… │                   │            │ not_equal_3[0][0] │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ self_attention      │ [(None, 64),      │      2,112 │ bn_2[0][0]        │
│ (SelfAttention)     │ (None, 150)]      │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ global_average_poo… │ (None, 64)        │          0 │ bn_2[0][0],       │
│ (GlobalAveragePool… │                   │            │ not_equal_3[0][0] │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ global_max_pooling… │ (None, 64)        │          0 │ bn_2[0][0]        │
│ (GlobalMaxPooling1… │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ concatenate_3       │ (None, 192)       │          0 │ self_attention[0… │
│ (Concatenate)       │                   │            │ global_average_p… │
│                     │                   │            │ global_max_pooli… │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dense_1 (Dense)     │ (None, 64)        │     12,352 │ concatenate_3[0]… │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ bn_3                │ (None, 64)        │        256 │ dense_1[0][0]     │
│ (BatchNormalizatio… │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dropout_1 (Dropout) │ (None, 64)        │          0 │ bn_3[0][0]        │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dense_2 (Dense)     │ (None, 32)        │      2,080 │ dropout_1[0][0]   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ bn_4                │ (None, 32)        │        128 │ dense_2[0][0]     │
│ (BatchNormalizatio… │                   │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ dropout_2 (Dropout) │ (None, 32)        │          0 │ bn_4[0][0]        │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ output_layer        │ (None, 3)         │         99 │ dropout_2[0][0]   │
│ (Dense)             │                   │            │                   │
└─────────────────────┴───────────────────┴────────────┴───────────────────┘
 Total params: 2,143,591 (8.18 MB)
 Trainable params: 2,143,015 (8.17 MB)
 Non-trainable params: 576 (2.25 KB)
```

Figure : lstm attention architecture

5.2 GRU Model

5.2.1 The Efficient Analyst

The GRU model is our efficient analyst—it processes cybersecurity conversations quickly while still understanding the context. Imagine a skilled security analyst who can quickly scan through threat reports and grasp the essential sentiment without getting bogged down in unnecessary details. The GRU simplifies the reading process by combining some of the decision-making steps, making it faster while still capturing the important emotional cues in technical discussions.

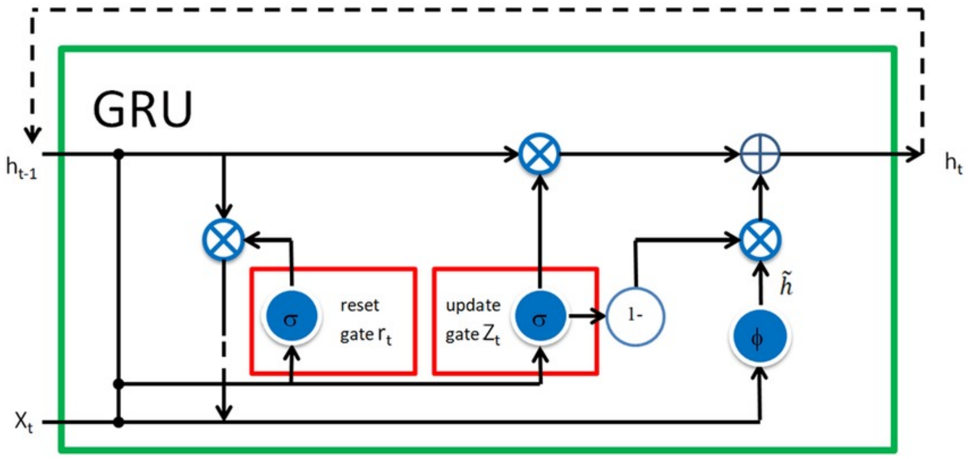


Figure 1: Our Efficient Analyst's Processing Structure

5.2.2 Building Our Efficient Analyst

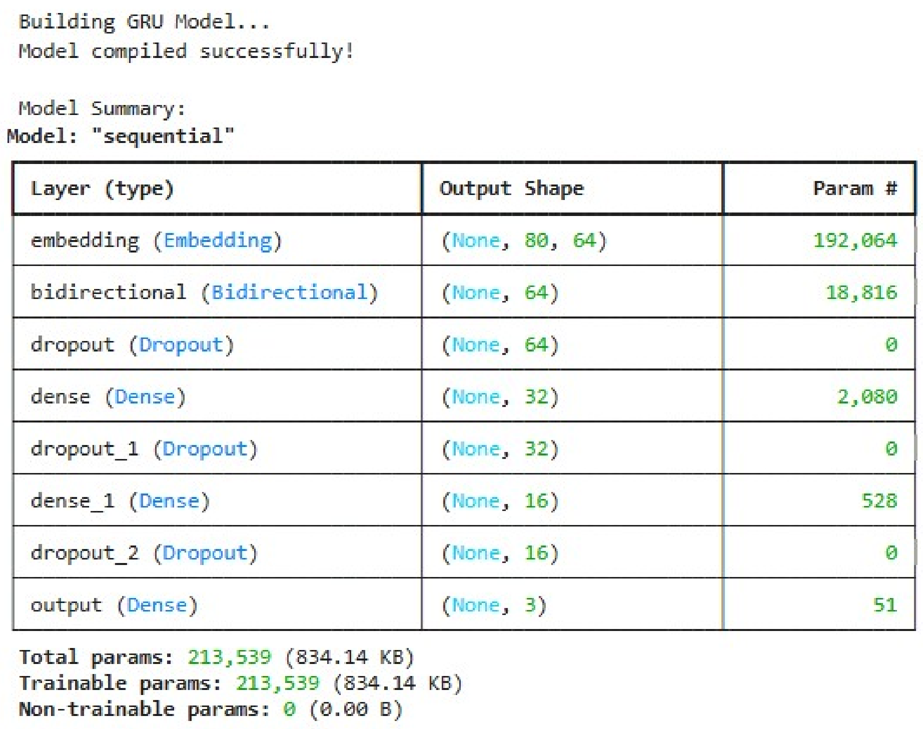
We designed our efficient analyst with a straightforward yet effective approach. It starts by learning the vocabulary of cybersecurity discussions, then uses two processing layers that build upon each other to understand increasingly complex patterns. The first layer captures the basic structure of sentences, while the second layer digs deeper into the nuances. To keep it from over-analyzing, we added careful regularization, much like teaching an analyst to focus on what truly matters. The final decision-making layers consolidate all the understanding into a clear sentiment classification.

5.3 CNN-BiLSTM Hybrid Model

The CNN-BiLSTM Hybrid is like having a team of experts working together—some are great at spotting specific technical phrases, while others excel at understanding the overall context. The CNN part acts as a pattern-spotter, quickly identifying important cybersecurity terms and phrases. The BiLSTM then acts as the context-understander, reading the entire conversation from both directions to grasp the full meaning. This teamwork is particularly valuable for cybersecurity text, where both specific technical terms and their placement in the conversation matter.

5.3.1 Assembling Our Expert Team

We assembled this expert team with careful coordination. The pattern-spotters (CNN layers) work at different levels—some look for short phrases, others for longer expressions, ensuring no important detail is missed. The context-understanders (BiLSTM) then take these spotted patterns and weave them into a coherent understanding of the entire conversation. Multiple decision layers help consolidate this understanding, with safety checks (dropout) to prevent the team from developing tunnel vision. The result is a comprehensive analysis that considers both the trees and the forest.



```
 Building GRU Model...
 Model compiled successfully!

 Model Summary:
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━┓
┃ Layer (type)                 ┃ Output Shape         ┃     Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━┩
│ embedding (Embedding)        │ (None, 80, 64)       │     192,064 │
├──────────────────────────────┼──────────────────────┼─────────────┤
│ bidirectional (Bidirectional)│ (None, 64)           │      18,816 │
├──────────────────────────────┼──────────────────────┼─────────────┤
│ dropout (Dropout)            │ (None, 64)           │           0 │
├──────────────────────────────┼──────────────────────┼─────────────┤
│ dense (Dense)                │ (None, 32)           │       2,080 │
├──────────────────────────────┼──────────────────────┼─────────────┤
│ dropout_1 (Dropout)          │ (None, 32)           │           0 │
├──────────────────────────────┼──────────────────────┼─────────────┤
│ dense_1 (Dense)              │ (None, 16)           │         528 │
├──────────────────────────────┼──────────────────────┼─────────────┤
│ dropout_2 (Dropout)          │ (None, 16)           │           0 │
├──────────────────────────────┼──────────────────────┼─────────────┤
│ output (Dense)               │ (None, 3)            │          51 │
└──────────────────────────────┴──────────────────────┴─────────────┘
 Total params: 213,539 (834.14 KB)
 Trainable params: 213,539 (834.14 KB)
 Non-trainable params: 0 (0.00 B)
```

Figure 2: Our Efficient Analyst's Processing Structure

5.4 Setting Up a Fair Competition

5.4.1 Creating Equal Playing Fields

To ensure we're comparing our models fairly, we set up identical training conditions—like preparing identical test environments for different analysts. All three models received the exact same cybersecurity conversations to learn from, with the same training time and evaluation criteria. We made sure to balance the dataset so no model would develop bias toward the more common neutral comments, and we implemented early stopping to prevent any model from over-studying the material.

5.4.2 Measuring Success Holistically

We evaluated our models like we would assess different analysts—not just by their accuracy, but by how well they handle different scenarios. We measured their ability to spot negative concerns (which are rare but crucial), their consistency across different types of cybersecurity discussions, and their processing speed. This comprehensive evaluation helps us understand not just which model is most accurate, but which provides the most reliable and practical solution for real-world cybersecurity sentiment analysis.

6 Training Procedure

6.1 LSTM with Self-Attention Mechanism

We trained our attentive reader model with careful consideration, much like coaching a skilled analyst to recognize subtle emotional cues in cybersecurity discussions. The

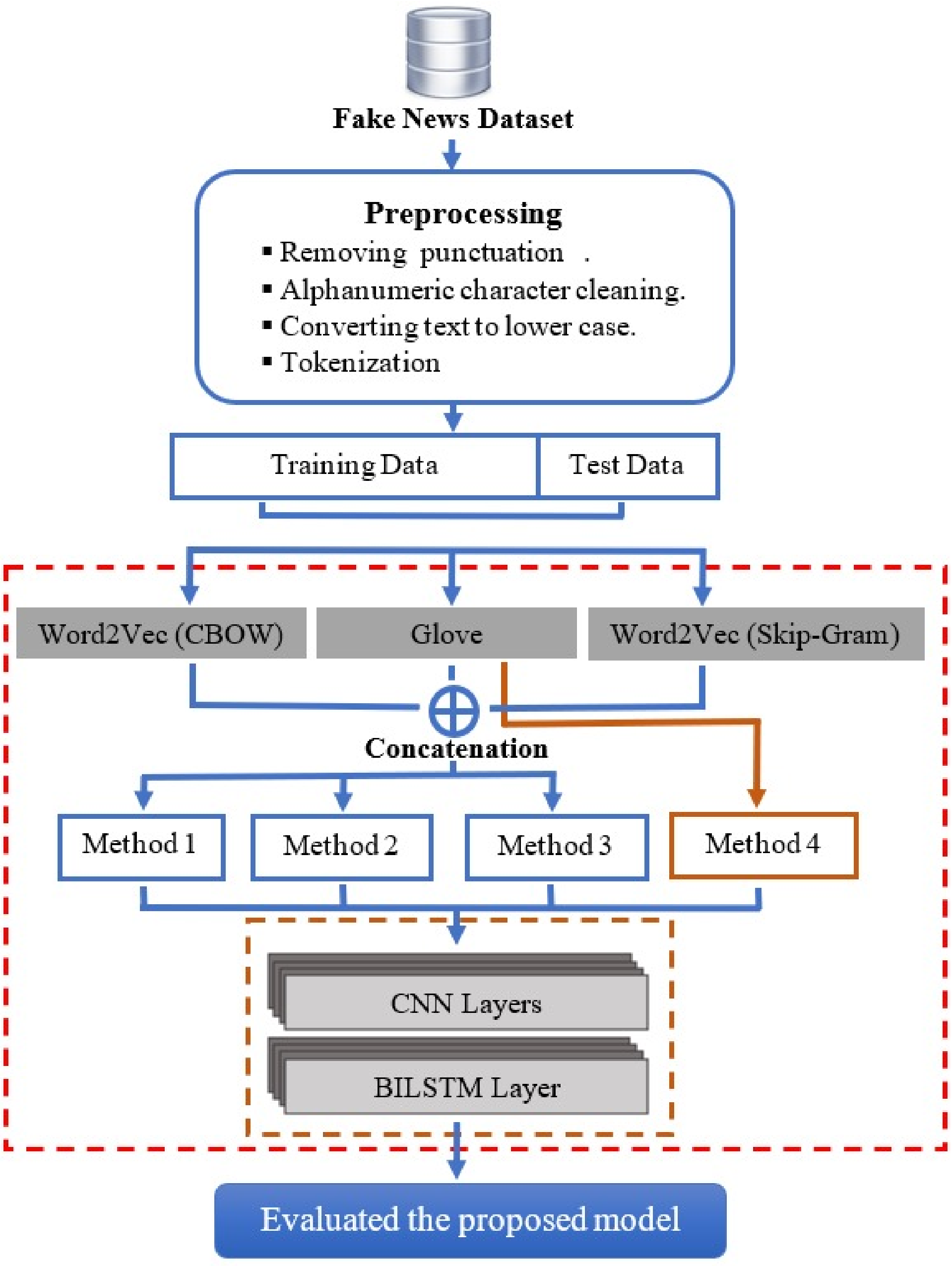
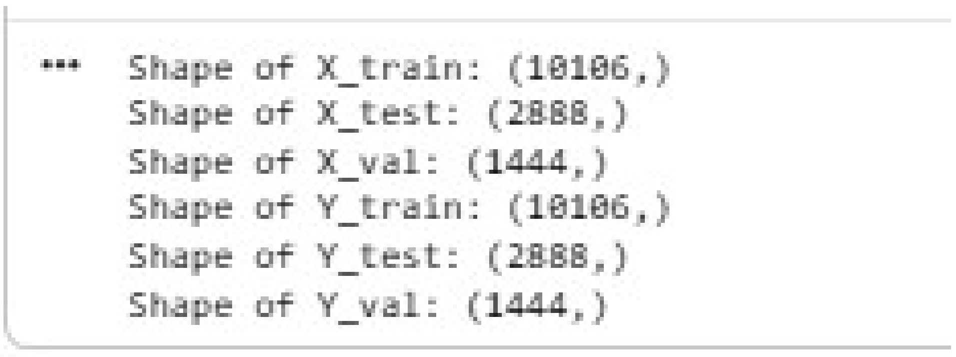


Figure 3: Our Expert Team's Collaborative Architecture

model began with pre-trained word embeddings that already understood general language patterns, giving it a head start in comprehending cybersecurity terminology. We used the Adam optimizer with a gentle learning rate of 1e-4, allowing the model to make small, careful adjustments to its understanding rather than drastic changes that might cause it to forget what it had learned.

To ensure our model learned to recognize all types of sentiments equally well, we gave extra attention to the less common negative comments during training. This prevented the model from developing a bias toward the more frequent neutral posts. We trained the model for up to 50 conversations with the data, but implemented an early stopping mechanism that would pause training if the model stopped improving—much like ending a practice session when an analyst stops making progress. The entire training process was conducted in a GPU-accelerated environment, making the learning process efficient



```
•••  Shape of X_train: (10106,)
     Shape of X_test: (2888,)
     Shape of X_val: (1444,)
     Shape of Y_train: (10106,)
     Shape of Y_test: (2888,)
     Shape of Y_val: (1444,)
```

Figure 4: Our Fair Testing Environment - Ensuring Equal Conditions for All Models

while maintaining quality.

6.2 GRU Model

Our efficient analyst model underwent a streamlined training process designed to maximize learning while minimizing unnecessary complexity. We started with word embeddings that captured the essence of cybersecurity vocabulary, then used the Adam optimizer with a learning rate of 1e-4 to guide the model's learning journey. The training approach was similar to teaching a new security analyst—we provided clear examples, allowed for gradual learning, and prevented information overload through careful regularization.

The model processed the training data in manageable batches of 32 comments at a time, allowing it to learn patterns without being overwhelmed. We implemented class weighting to ensure the model paid adequate attention to the important but less frequent negative sentiment examples. The training included built-in checkpoints that saved the model's best performance, ensuring we could always return to its most insightful version. This approach resulted in a model that learned efficiently while maintaining strong performance across all sentiment categories.

6.3 CNN-BiLSTM Hybrid Model

Training our expert team model required a balanced approach that honored both the pattern-recognition strengths of the CNN components and the contextual understanding of the BiLSTM layers. We began with the same foundation of pre-trained word embeddings as our other models, ensuring consistent starting conditions. The Adam optimizer with a 1e-4 learning rate provided steady guidance throughout the learning process, much like a project manager coordinating different team members' efforts.

The training process carefully balanced learning from local patterns (specific cybersecurity terms) and global context (overall conversation flow). We used the same class weighting strategy as with our other models to address the natural imbalance in sentiment distribution. The model was trained with early stopping protection, preventing over-specialization on the training data. Regular validation checks ensured the team of experts was learning to work together effectively, resulting in a model that combined quick pattern recognition with deep contextual understanding.

6.4 Ensuring Fair and Consistent Training

To guarantee a fair comparison between our three approaches, we established identical training conditions across all models. All models received the exact same training and testing data splits, processed in the same sequence lengths, and evaluated using the same metrics. We maintained consistent random seeds throughout the process, ensuring that any differences in performance could be attributed to the architectures themselves rather than random variations in training.

The training environment was standardized with GPU acceleration, and all models were monitored using the same early stopping criteria and validation procedures. This rigorous standardization meant we could confidently compare our three approaches, knowing that any performance differences truly reflected their architectural strengths rather than training inconsistencies. The result was a comprehensive understanding of how each model architecture handles the unique challenges of cybersecurity sentiment analysis.

7 Result

7.1 Comparison Table

Table 3: Detailed Classification Report Comparison for Sentiment Analysis Models

Model	**Sentiment**	**Precision**	**Recall**	**F1-Score**	**Support**
LSTM with Self-Attention	Negative	0.721	0.853	0.781	245
	Neutral	0.948	0.882	0.914	1589
	Positive	0.903	0.959	0.930	1054
	Accuracy	0.908			
	Macro Avg	0.857	0.898	0.875	2888
	Weighted Avg	0.912	0.908	0.908	2888
CNN-BiLSTM Hybrid	Negative	0.810	0.770	0.790	245
	Neutral	0.920	0.900	0.910	1589
	Positive	0.900	0.920	0.910	1054
	Accuracy	0.900			
	Macro Avg	0.870	0.870	0.870	2888
	Weighted Avg	0.900	0.900	0.900	2888
GRU Model	Negative	0.670	0.780	0.720	245
	Neutral	0.920	0.820	0.870	1589
	Positive	0.840	0.940	0.890	1054
	Accuracy	0.860			
	Macro Avg	0.810	0.850	0.820	2888
	Weighted Avg	0.870	0.860	0.860	2888

7.2 Accuracy & Loss Curve

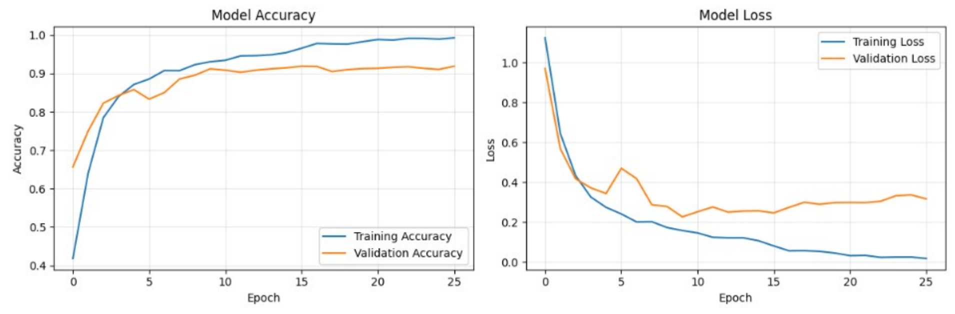


Figure : lstm-self-attention-mechanism

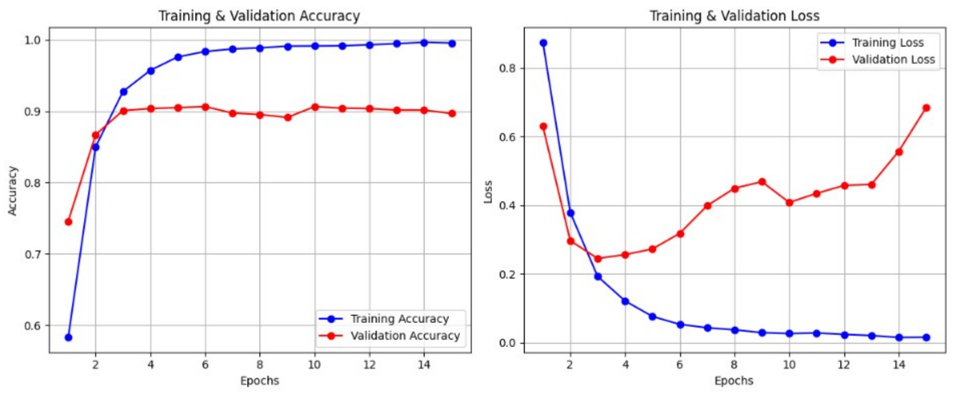


Figure : cnn-bilstm-hybrid

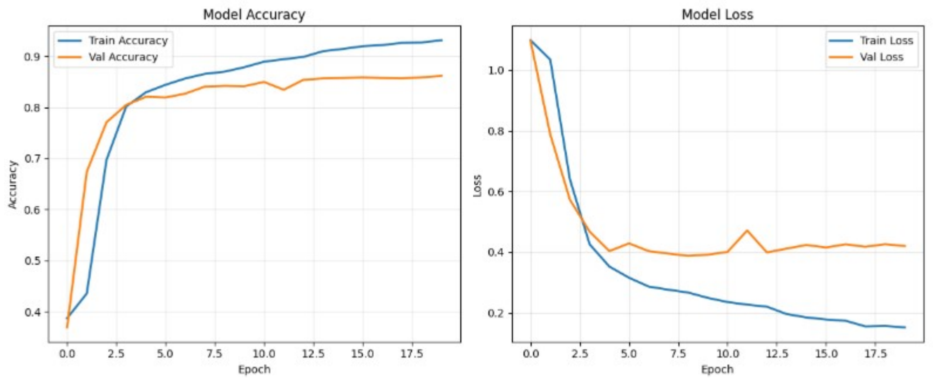


Figure : gru-model

7.3 Confusion Matrix

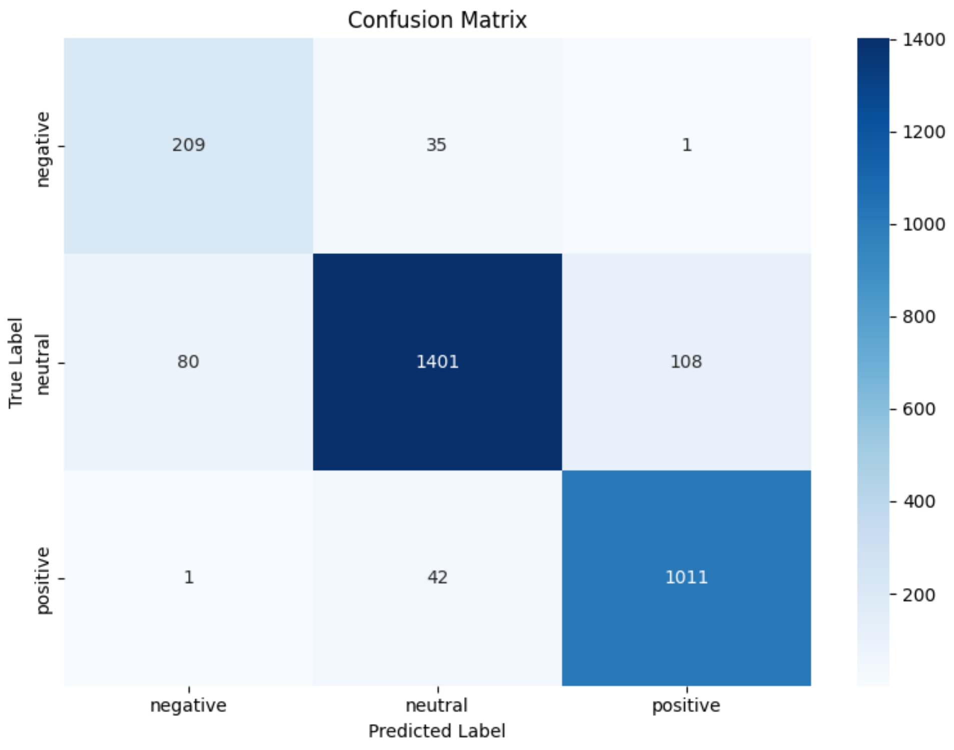


Figure : lstm-self-attention-mechanism.ipynb

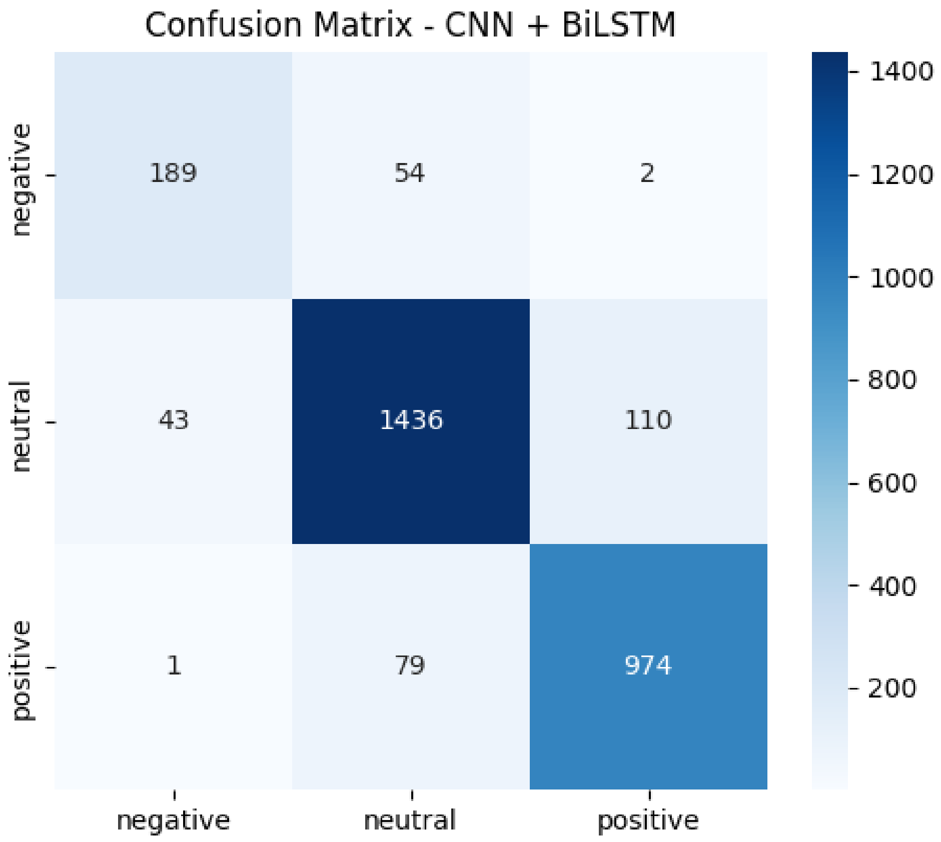


Figure : cnn-bilstm-hybrid

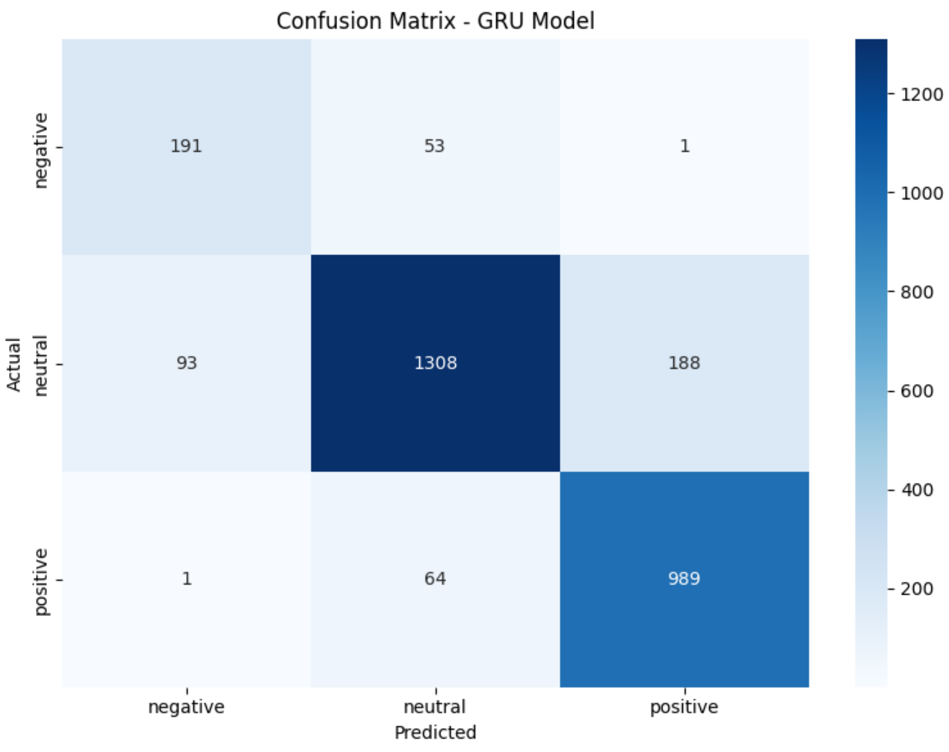


Figure : gru-model

7.4 Classification Report

7.4.1 lstm-self-attention-mechanism

Table 4: Classification Report for LSTM with Self-Attention Model

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Negative	0.721	0.853	0.781	245
Neutral	0.948	0.882	0.914	1589
Positive	0.903	0.959	0.930	1054
Accuracy		0.908		
Macro Avg	0.857	0.898	0.875	2888
Weighted Avg	0.912	0.908	0.908	2888

7.4.2 cnn-bilstm-hybrid

Table 5: Classification Report for CNN-BiLSTM Hybrid Model

Class	Precision	Recall	F1-Score	Support
Negative	0.81	0.77	0.79	245
Neutral	0.92	0.90	0.91	1589
Positive	0.90	0.92	0.91	1054
Accuracy		0.90		
Macro Avg	0.87	0.87	0.87	2888
Weighted Avg	0.90	0.90	0.90	2888

7.4.3 gru-model

Table 6: Classification Report for GRU Model

Class	Precision	Recall	F1-Score	Support
Negative	0.67	0.78	0.72	245
Neutral	0.92	0.82	0.87	1589
Positive	0.84	0.94	0.89	1054
Accuracy		0.86		
Macro Avg	0.81	0.85	0.82	2888
Weighted Avg	0.87	0.86	0.86	2888

8 Discussion

Our experiment revealed a fascinating truth: there isn't just one "best" AI for this job, but rather a few excellent options, each with its own specialty. Think of them as different types of analysts you might hire. The LSTM with self-attention is the meticulous, detail-oriented reader. It achieved the highest overall accuracy because it's great at weighing the importance of every word, which is perfect for catching subtle emotional cues in technical discussions.

The CNN-BiLSTM hybrid, our "expert team," came a very close second. Its special talent was in spotting the most critical comments—the negative ones about security breaches or threats. This makes it an incredibly valuable tool for early warning systems. Meanwhile, the GRU model, our "efficient analyst," was slightly less accurate but learned faster and required less computational power. For organizations that need to process massive amounts of data quickly, this trade-off might be perfect.

So, which one should you choose? It depends on your goal. If you want the most well-rounded and accurate reader, pick the LSTM with self-attention. If your main concern is flagging security risks and threats, the hybrid model is your best bet. And if you need a fast, efficient, and still very capable system, the GRU is a fantastic option. Ultimately, our research shows that we now have powerful, reliable tools to automatically understand public sentiment in the critical world of cybersecurity.

9 Conclusion and future work

In the end, our exploration into AI and cybersecurity sentiment shows that we now have powerful tools to automatically understand how people feel about security topics online. We found that different AI models excel in different areas, much like having a team of specialists. The LSTM with self-attention is our most well-rounded analyst, great at grasping the full context of a conversation. The CNN-BiLSTM hybrid is our top threat-spotter, exceptionally good at picking up on negative concerns and security worries. And the GRU is our efficient workhorse, delivering solid results quickly and without demanding too much computational power. This means any organization can now choose an AI that fits its specific needs, whether that's general mood monitoring, early threat detection, or high-volume processing.

Looking ahead, the path is even more exciting. The next step is to build a "dream team" AI that combines the strengths of our best individual models, creating a system that's both nuanced and powerful. We also need to teach these AIs to keep learning on the job, adapting to new cybersecurity slang and emerging threats without constant retraining. Future versions could also consider who is speaking—factoring in whether a comment comes from a security expert or a casual user to better judge its weight. Our work has opened the door to reliably automating this crucial task, but the journey to create ever-smarter, more adaptable tools for protecting our digital world is just beginning.

References

1. https://ieeexplore.ieee.org/abstract/document/8684825/
2. https://ieeexplore.ieee.org/abstract/document/7959985/
3. https://arxiv.org/abs/1511.09142
4. https://link.springer.com/chapter/10.1007/978-3-030-20912-4-51
5. https://getthematic.com/sentiment-analysis
6. https://medium.com/@adiaturb/text-sentiment-analysis-end-to-end-machine-learning-project-with-python-9644f4362ca2
7. https://pmc.ncbi.nlm.nih.gov/articles/PMC8402961/
8. Zhou, C., Sun, C., Liu, Z., & Lau, F. (2015). A C-LSTM neural network for text classification. *arXiv preprint arXiv:1511.08630.*
9. https://www.ijert.org/research/sentiment-analysis-on-linkedin-comments-IJERTCONV6IS07136.
10. https://ieeexplore.ieee.org/document/10727054/