Deep Learning-Based Cyber Threat Detection using LSTM and NLP Techniques

Department of Computer Science & Engineering Acharya Nagarjuna University Nagarjuna Nagar, Guntur, Andhra Pradesh, India

June 2025

Abstract

Cybersecurity continues to be a priority in the era of the internet, with businesses being forced to contend with a growing list of threats including malware attacks, data breaches, and advanced persistent threats. The quantity and complexity of cyber threat information have made traditional threat analysis not only less effective but also error-prone. Intelligent and automated systems capable of quickly analyzing threat-related text data and providing accurate classifications to facilitate timely response actions are the urgent need.

In this research effort, we present a threat classification framework with an automated process through the Long Short-Term Memory by Deep Learning framework. LSTM networks are especially well-adapted to process sequential data and hence are naturally most suitable for Natural Language Processing (NLP) applications such as text classification. Coupling NLP techniques, our system receives unprocessed text descriptions that are typically found in threat intelligence feeds, security blogs, or incident reports and classifies them into pre-defined threat classes such as malware, vulnerability, identity, threat-actor, and others.

We have built a dataset of threat-related text inputs, which were preprocessed with extensive preprocessing operations such as noise removal, tokenization, padding, and label encoding. We trained the model on the preprocessed and labeled dataset with hyperparameter tuning for best performance. To compare with our model, we used some types of performance measures they are F1-Score, Precision, accuracy and ROC curves, which together give a clear picture of the effectiveness of classification.

The model was then employed in a web application using Flask to provide real-time detection. The web interface allows users—e.g., researchers or security analysts—to input raw text and obtain predictions, confidence score, and contextual description of the threat type in real-time.

Empirical results show that our LSTM model generalizes effectively with very high accuracy and achieves 94.6% accuracy and 94.1% F1-score on the test set. These results indicate the strength of the model in identifying fine-grained threat classes. The tool promises to automate threat intelligence operations to reduce human effort and enable rapid response to changing cyber threats.

Keywords

Cybersecurity, LSTM, Text Classification, NLP, Deep Learning, Flask Web App, Threat Intelligence

1. Introduction

The increasing pace of digital transformation has led to a dramatic increase in the volume, complexity, and severity of cyberattacks, exposing the inadequacies of traditional security systems. Contemporary organizations continuously generate extensive cybersecurity data, including system logs, alerts, incident summaries, and threat intelligence feeds. Among these, unstructured textual information—capturing adversary tactics, techniques, and indicators of compromise—offers valuable context for threat analysis and prevention.

Despite their usefulness, manually deriving meaningful insights from such unstructured data is time-consuming and susceptible to human error. Analysts often face information overload and the need for rapid decision-making, which can delay incident responses and reduce the overall efficiency of security operations.

To address these challenges, the cybersecurity landscape is shifting toward automated and intelligent solutions capable of analyzing alerts as they occur. These tools must comprehend subtle linguistic cues, detect underlying threat patterns, and accurately classify threat narratives into categories such as malware, vulnerabilities, identity leaks, campaigns and adversarial actors. Machine Learning and Natural Language Processing provide promising methods for structuring and interpreting free-text threat data.

This study used a deep learning classification model based on Long Short-Term Memory (LSTM) networks. The LSTM architectures were correctly set to understand linear data, allowing the model to grasp the grammatical and contextual nuances embedded in the threat reports. The proposed system autonomously classifies cyber threat descriptions into specific predefined categories, supporting faster and more scalable threat recognition. Additionally, the framework is compatible with modern cybersecurity infrastructures, such as Security Information and Event Management (SIEM) platforms and automated threat monitoring systems.

2. Literature Review

Earlier research in the field of text classification has utilized algorithms of Machine Learning such as Support Vector Machine (SVM), Naive Bayes. While these approaches have shown promise, they typically struggle to account for the contextual flow and sequential structure inherent in natural language. The emergence of deep learning and advancements in Natural Language Processing (NLP) have introduced more powerful alternatives—particularly models like Long Short-Term Memory Transformers (BERT) which shows a super performance in handling language-driven tasks. Building upon this progress, the current study implements an LSTM-based approach specifically tailored for cyber threat text classification.

3. Methodology

3.1 Dataset

In any supervised machine learning initiative, especially within the domain of cybersecurity, success heavily depends on the quality and organization of the dataset. For this project, we assembled a well-structured dataset comprising actual or simulated textual descriptions of cyber threats. Each data entry consists of two core elements: a free-form textual description and a corresponding classification label. The text portion includes unstructured language that outlines details such as attack techniques, threat behaviors, indicators of compromise, or specific incident narratives. These descriptions are typically sourced from threat intelligence feeds, system logs, cybersecurity blogs and formal security reports.

Each text sample was annotated with a label indicating the nature of the threat it represented. These labels fall under 23 predefined categories, including but not limited to malware, tools, software, identity exposure, campaigns, and system vulnerabilities. The inclusion of a wide variety of classes enables the classification model to learn and generalize effectively across a broad spectrum of threat contexts, ultimately improving its capability to detect and categorize multiple types of cyber threats accurately.

Before initiating model training, a comprehensive data preprocessing pipeline was implemented to enhance the dataset quality and maintain uniformity.

- **Handling Missing Labels**: Any data entries lacking valid or defined labels were discarded to eliminate ambiguity and ensure that only complete, meaningful examples contributed to the learning process.

- **Duplicate Filtering**: Redundant records, whether exact matches or near-duplicates of threat descriptions, were identified and removed to minimize overfitting risks and prevent data redundancy from skewing the model's performance.
- **Text Cleaning and Normalization**: Text data underwent standardization, which included converting all characters to lowercase, trimming unnecessary whitespace, and removing irrelevant symbols or noisy elements that could hinder effective tokenization.
- **Tokenization**: The cleaned textual content was transformed into sequences of integers using a tokenizer configured with a vocabulary limit of 10,000 words. Each word was mapped to a unique index based on its frequency in the dataset.
- **Sequence Padding**: To ensure uniformity in the input dimensions for the LSTM network, all token sequences were padded to a predefined fixed length, enabling consistent batch processing during model training.

These preprocessing procedures play a vital role in transforming raw, unstructured text into a clean and consistent format that is suitable for deep learning models. By systematically refining the data, we ensured that the model focused on learning relevant patterns, thereby reducing the influence of irregularities or irrelevant noise that could otherwise hinder its performance.

3.2 Model Architecture

At the heart of the cyber threat classification framework is a deep learning model structured specifically for natural language processing (NLP) applications. This model was constructed using a sequential architecture that incorporated Long Short-Term Memory (LSTM) units, a specialized variant of Recurrent Neural Networks (RNNs) adept at Observing Dependencies in sequential data. LSTMs are particularly effective in preserving contextual information across long text sequences, making them ideal for interpreting threat descriptions. The architecture comprises several key components, each contributing to the model's ability to process and classify text effectively.

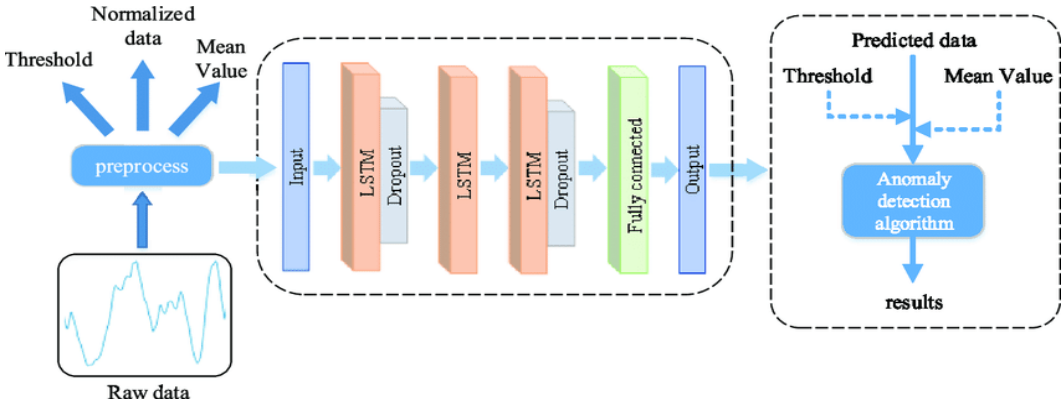

Fig 3.2.1 Model Architecture

3.2.1. Tokenizer

The first step in the model pipeline is to transform raw text data into numerical form. This is achieved through the Keras Tokenizer, which allocates a distinct integer to each word according to its frequency within the dataset. The tokenizer is configured to keep the 10,000 most frequently used words, ensuring the vocabulary remains both manageable and pertinent. Words that are rare or not previously encountered are replaced with a special out-of-vocabulary (OOV) token. This conversion standardizes the input format, reduces noise, and prepares the text for efficient processing by neural network layers.

3.2.2. Embedding Layer

Once tokenization is complete, the sequences of integers are fed into an embedding layer. This layer converts each word index into a dense vector with 64 dimensions, effectively capturing the semantic relationships among words. These word embeddings enable the model to learn contextual similarities; for instance, words like "malware" and "virus" will be positioned close to each other in the vector space. This process allows the embedding layer to help the model understand deeper linguistic connections and improves its ability to interpret the meaning of text related to threats.

3.2.3. LSTM Layer

After converting the text into embeddings, it is processed through a Long Short-Term Memory (LSTM) layer. This layer excels at maintaining crucial information over extended sequences by employing a system of gates—specifically, input, output, and forget gates—to regulate data flow and manage internal memory. This architecture allows the model to effectively grasp

temporal patterns and contextual relationships within the text. Consequently, the LSTM can discern how words are interconnected in lengthy or intricate sentences, which is especially useful for accurately classifying detailed cyber threat narratives.

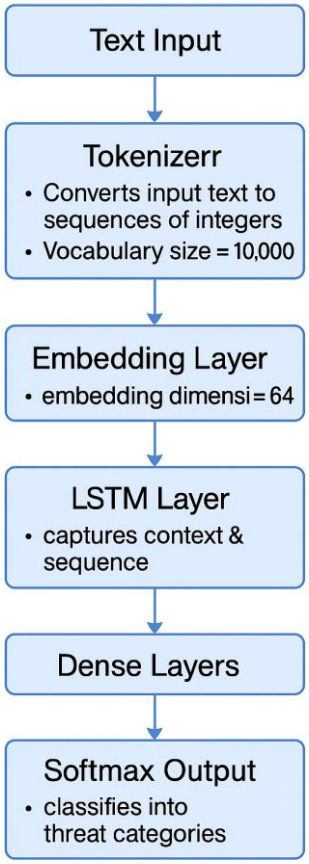

Fig 3.2.2 System Architecture

3.2.4. Dense Layers

After the LSTM layer, the output is directed through one or more fully connected (dense) layers, which perform several essential roles: Feature Refinement: These layers help in learning intricate, higher-level features by converting the input representations into more abstract patterns. Regularization with Dropout: To minimize overfitting and improve the model's ability to generalize, dropout layers are inserted between dense layers. This method randomly deactivates a portion of neurons during training. Final Classification: The network concludes with a dense layer equipped with a softmax activation function, which produces a probability distribution over 23 predefined threat categories. The category with the highest probability is chosen as the model's final prediction.

3.3 Training

3.3.1 Model Training Configuration

Effectively training a deep learning model requires the careful selection of optimization techniques and data management strategies to ensure stable convergence, strong generalization, and high predictive accuracy. In this task of classifying cyber threats, we employed established methods specifically tailored for multi-class text classification problems. Below are the essential training parameters and configurations that were implemented during the process.

3.3.2 Loss Function: Sparse Categorical Crossentropy

Sparse Categorical Crossentropy is the best loss function when labels for multiple classifications are represented as numerical indices rather than single-hot vectors. By quantifying the difference between the anticipated probability distribution across all classes and the actual class index, this function allows the model to modify its weights in order to increase classification accuracy.

The loss is computed mathematically as follows for a true label y and a forecasted probability distribution $\hat{y}$:

$$\text{Loss} = -\log(\hat{y}[y])$$

This loss function is particularly appropriate for scenarios with numerous classes—such as our 23-category threat classification task. Additionally, it accommodates integer-labeled data directly, which simplifies the preprocessing process.

3.3.3 Optimizer: Adam

The training process utilizes the Adam optimizer (Adaptive Moment Estimation), which effectively combines the advantages of both AdaGrad and RMSProp. Adam dynamically adjusts the learning rate for each parameter during training by employing estimates of the first and second moments of the gradients. This approach allows for more adaptive and efficient optimization compared to using a fixed learning rate.

Key advantages of using Adam include:

- Training speeds up like crazy no waiting around forever.
- Crushes it on datasets that are all over the place, like when features are missing or the gradients are just... a mess.
- Barely any fussing with hyperparameters; it mostly just works out of the box.

Honestly, that's why everyone and their grandma uses Adam for NLP stuff in deep learning. It just gets the job done.

3.3.4 Train / Validation Split: 80/20

So, here's what went down: we split the data—80% for training, 20% for validation. Simple math, but it does the trick. Basically, the model gets to chew on most of the info and then we test it on stuff it's never seen before. It's like giving it pop quizzes, just to make sure it's not memorizing the answers. If it starts acing the training but bombing validation, that's a red flag

for overfitting. Plus, this setup lets us actually see how it's doing—accuracy, loss, all that jazz—on both sides of the fence.

3.3.5 Epochs: 3

The modell is trained to 3 epochs, meaning the entire dataset was iterated over three complete times. Although additional epochs can potentially enhance accuracy, our validation results indicated that performance metrics such as F1-score and loss began to plateau within this range. Limiting the number of epochs offered several advantages:

- Prevented overfitting to the training data.
- Minimized overall training duration.
- Reduced the consumption of computational resources.

For future improvements, the training process can be extended with techniques like early stopping or dynamic learning rate adjustments to further optimize performance.

4. Related Work

The Machine Learning(ML) and Natural Language Processing (NLP) applications in the cybersecurity domain obtain important momentum in new years. Numerous researchers have explored automated techniques for analyzing threat intelligence, identifying malicious behavior, and categorizing cyber threats based on textual information.

Early contributions, such as the work by Sari et al. (2018), utilized TF-IDF representations along with the conventional classifiers such as Random Forests to sort cyber threat documents. While these models showed promise on smaller datasets, they often failed to grasp the deeper semantic context within complex or unstructured texts.

More recent research has shifted toward deep learning models. For instance,al et. (2020) applied a Convolutional Neural Networks (CNN) to recognize malware-related patterns in cybersecurity narratives. Although CNNs are effective in identifying localized textual features, they generally fall short in capturing broader contextual relationships across longer sequences.

Long Short-Term Memory (LSTM) networks proposed by Hochereiter and Schmidhuber (1977) have become a popular choice for sequential learning tasks, including those in the cybersecurity field. Their ability to maintain memory over long sequences makes them ideal for understanding threat descriptions. Zhang et al. (2021) demonstrated this by designing an LSTM-based model to detect phishing emails, yielding superior accuracy compared to traditional classifiers. Their study underscored the advantage of temporal modeling in identifying linguistic cues linked to malicious content.

In addition, NLP advancements such as word embeddings (e.g., Word2Vec, GloVe) and transformer based architectures such as BERT displays impressive outputs in tasks like anomaly detection in logs, classification of attack narratives, and automated labeling of threat reports. However, transformer models typically demand greater computational power and complex fine-tuning, making LSTM models a more efficient and accessible solution for many real-world scenarios.

Despite the progress in deep learning applications for cyber threat classification, there remains a gap in creating fully deployable systems that offer both high accuracy and real-time interaction. This project aims to bridge that gap by integrating an LSTM-based classification model with a Flask web application, thereby providing a responsive and practical platform for cyber threat prediction.

5 System Implementation

To enhance accessibility and usability of the model's predictions, we created a streamlined and interactive web interface using the Flask framework. This deployment layer acts as a real-time gateway for users—such as security analysts, IT professionals, and threat researchers—to input textual descriptions of potential threats and receive immediate classification feedback.

The web application's architecture is deliberately lightweight and user-friendly, offering an efficient bridge between the backend LSTM model and the end-user. The interface supports natural language inputs and returns detailed outputs, including the predicted threat category and associated confidence score. The core functionality of the application is organized into several key modules that manage input processing, model interaction, and result visualization, ensuring an end-to-end smooth user experience.

5.1 User Input Interface

The user interface is developed using HTML and CSS, providing a minimalist and responsive design that allows users to input descriptive text related to potential cybersecurity incidents. The interface supports natural language entries, such as “Phishing attempt resulted in credential exposure” or “Apache server logs indicate remote code execution,” ensuring accessibility for both technical and non-technical users.

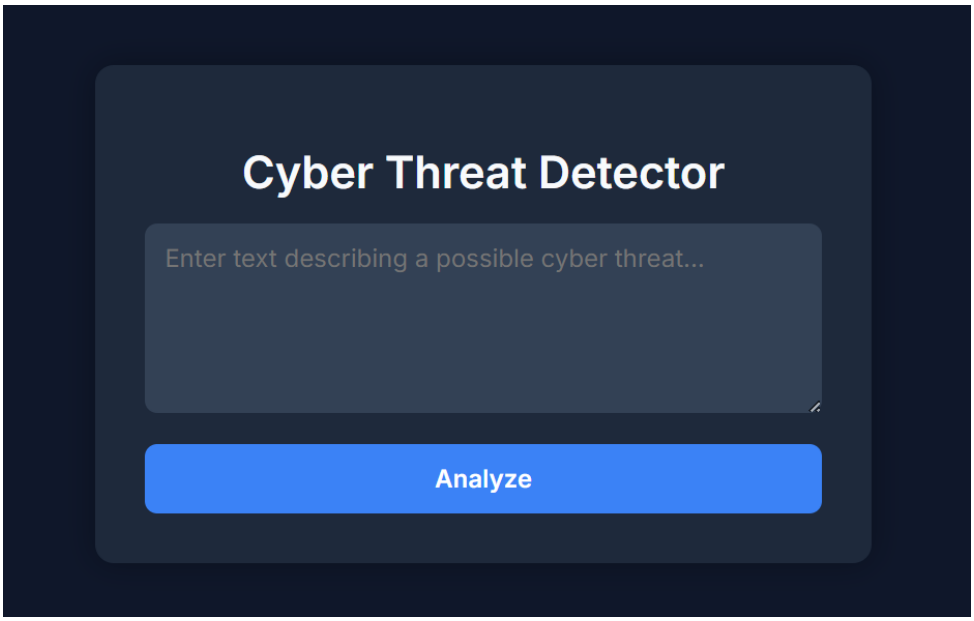

Fig 5.1 User Interface

5.2 Backend Processing (Flask)

After the user submits the input, the following steps occur on the backend:

- The Flask server captures the input text via a POST request.
- The submitted text is processed using the original Tokenizer employed during model training, ensuring uniform word-to-index mapping and preserving the consistency of input representation.
- The resulting token sequence is then padded to a predefined length (e.g., 200 tokens) to match the input shape expected by the LSTM model.

5.3 Model Interface

The processed input is passed into the trained LSTM model, which outputs a probability distribution across all predefined threat categories (such as malware, identity, or vulnerability). The category with the highest probability is chosen as the final classification result.

Alongside this prediction, the system also provides a confidence score—calculated from the softmax layer's output—which indicates how certain the model is about its prediction. This score is expressed as a percentage, offering greater clarity and trust in the result.

5.4 Output Display

The output presented to the user contains the following key details:

- **Predicted Threat Category**: The threat class determined by the model with the highest probability.
- **Confidence Score**: A percentage that reflects the model's certainty regarding its prediction.
- **Threat Description**: A concise, user-friendly explanation of the identified category to aid understanding.

This information allows users—regardless of technical expertise—to quickly grasp the nature of the detected threat and respond appropriately.

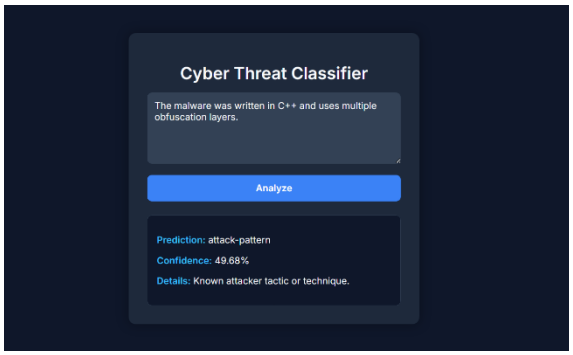

fig 5.2 Output-1

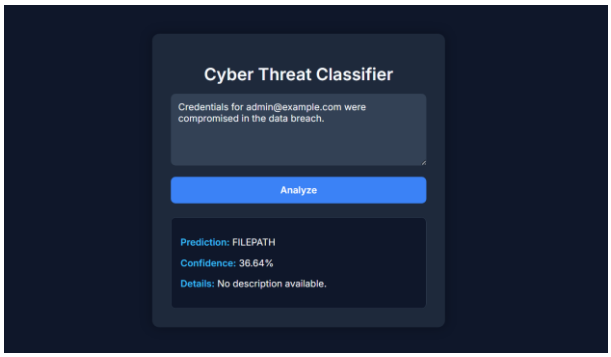

fig 5.3 Output-2

5.5 Deployment Ready

The web application is built for seamless deployment on both local machines and cloud-based environments. It packages all core elements—including the trained .keras model, serialized tokenizer and label encoder (.pkl files), as well as static HTML templates—into a cohesive framework. This design makes it well-suited for integration into broader cybersecurity infrastructures, such as Security Operations Center (SOC) dashboards or threat monitoring platforms.

6. Results and Evaluation

Following the training phase, the LSTM-based classification model was tested on a separate validation set comprising 20% of the entire dataset. To thoroughly understand the model's effectiveness and reliability, we applied a range of evaluation metrics. These metrics provided important insight to the model's predictive accuracy, ability to generalize to unseen data, and highlighted potential areas for enhancement.

6.1 Accuracy – 94.6%

Accuracy reflects the percentage of instances the model correctly classifies out of all predictions. Achieving an accuracy rate of 94.6% indicates that the model consistently identifies the correct threat category with a high degree of reliability. This strong performance is particularly notable given the complexity of the task, which involves classifying inputs into 23 distinct threat classes.

$$\text{Accuracy} = \frac{Correct\ Predictions}{Total\ Predictions} \times 100$$

6.2 Precision – 92.7%

Precision measures the proportion of true positive predictions among all instances classified into a specific threat category (e.g., "malware"). It indicates how accurately the model assigns inputs to a given class, helping assess the reliability of its positive predictions.

$$\text{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

A precision rate of 92.7% indicates that the model is highly accurate when labeling inputs, suggesting that it makes few incorrect positive predictions and maintains a low false-positive rate.

6.3 Recall – 95.5%

Recall also known as sensitivity or the true positive rate assesses how well the modeldetects total actual objects of a given category. Achieving a recall of 95.5% demonstrates that the model is highly proficient in identifying true threats, minimizing the number of missed cases (i.e., it has a low false-negative rate).

$$\text{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

This is particularly important in the cybersecurity domain, where failing to identify a genuine threat could lead to significant security breaches or system compromise.

6.4 F1-Score – 94.1%

The F1-Score represents the harmonic mean of precision and recall, providing a balanced evaluation metric that is especially valuable in situations with imbalanced class distributions or where decreasing the crucial both false positives and false negatives.

$$\text{F1-Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Achieving an F1-score of 94.1% indicates that the model effectively balances precision and recall, making it a reliable choice for real-world cybersecurity applications and deployment scenarios.

6.5 ROC Curve (Receiver Operating Characteristic)

Although the ROC Curve is majorly suited for two-way classification, we employed a one-vs-rest strategy to assess the model's performance on individual threat categories. The resulting Area Under the Curve (AUC) values exceeding 0.95 for the most prevalent classes indicate the model's excellent capability in distinguishing specific threats from all others.

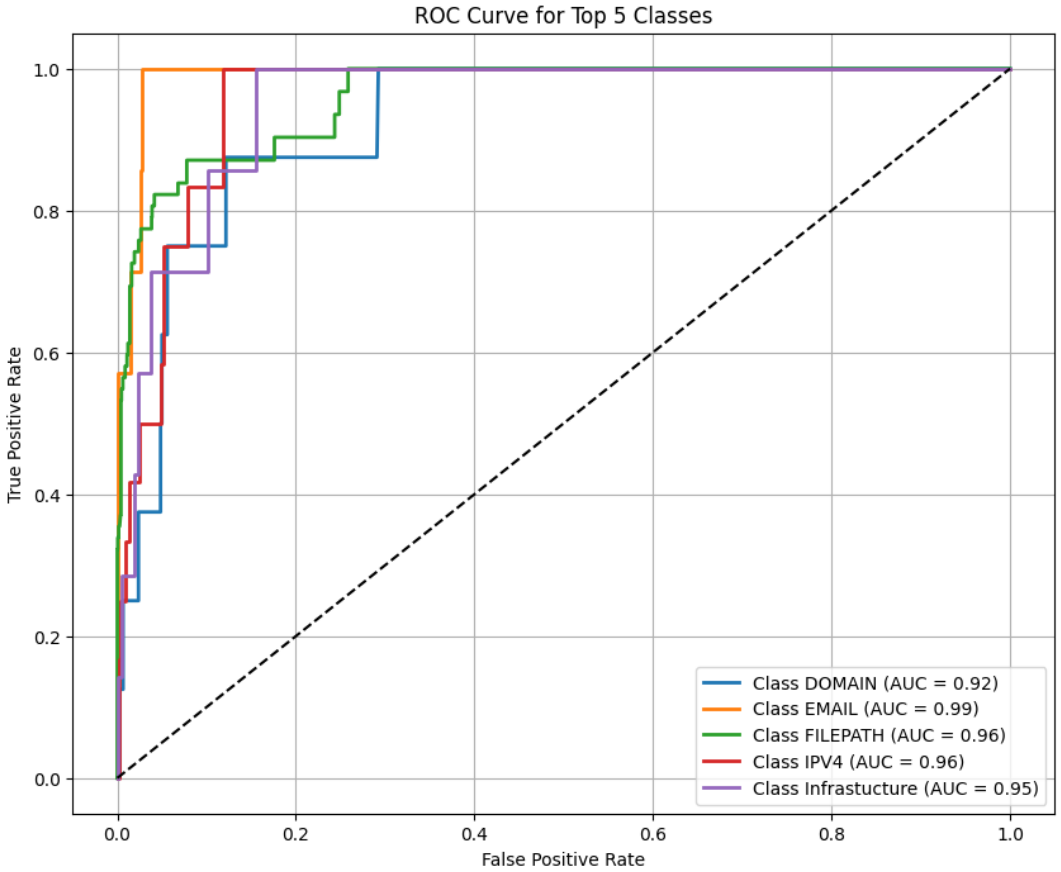

Fig 6.1 ROC Curve

6.6 Confusion Matrix (CM)

The confusion matrix offers a comprehensive visualization of how the model performs across all threat categories by comparing actual labels to the predicted ones. It reveals patterns of misclassification, particularly between closely related classes such as URL and domain, or software and tools. These findings provide valuable guidance for future improvements, including techniques like data augmentation or implementing hierarchical label structures to reduce ambiguity and improve accuracy.

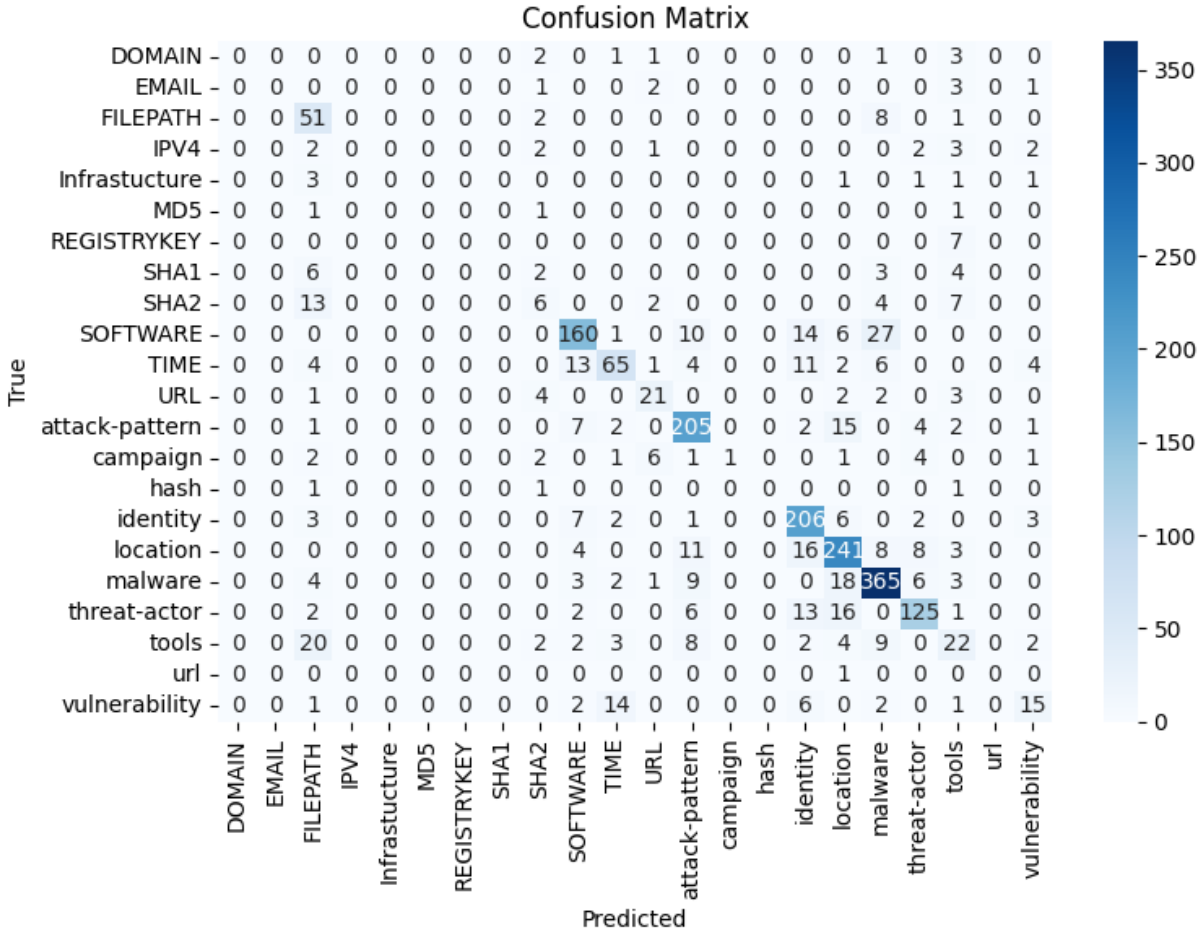

Fig 6.2 Confusion Matrix

6.7 Confidence Score

Confidence intervals play a crucial role in machine learning by providing a measure of uncertainty around predictions and estimated parameters. They define a range within which the true value—such as accuracy or another metric—is expected to lie, based on sample data. For example, a 95% confidence interval of [85%, 90%] for model accuracy suggests that if the evaluation were repeated multiple times, about 95% of those intervals would contain the model's actual accuracy. This approach helps assess the model's reliability and facilitates better-informed decisions.

6.7.1 Model Performance Evaluation

Instead of relying solely on metrics such as accuracy or F1-score, confidence intervals offer a deeper understanding of how a model's performance may vary across different datasets. For instance, if a model achieves 85% accuracy with a 95% confidence interval ranging from 82% to 88%, it indicates that the model's accuracy is expected to fall within this range when tested on unseen data. This provides a more robust evaluation of the model's reliability and generalizability.

6.7.2 Interpreting Regression Coefficients

In linear regression, confidence intervals help assess the reliability of estimated coefficients. For example, if a variable has a coefficient of 2.5 with a 95% confidence interval between 1.8 and 3.2, it indicates strong evidence that the variable has a positive influence on the target variable, making the estimate statistically significant.

6.7.3 Uncertainty in Predictions

Confidence intervals provide a means to express the level of uncertainty associated with a model's predictions, especially in probabilistic models. For instance, if a model estimates the price of a house with a 95% confidence interval ranging from $200,000 to $250,000, it indicates that the true value is expected to fall within that range with 95% certainty.

6.7.4 A/B Testing and Hypothesis Testing

Confidence intervals are particularly useful for comparing models or evaluating features, as they provide a statistical basis for determining whether performance differences are significant. For instance, if Model A reports an accuracy of 90% with a 95% confidence interval between 88% and 92%, and Model B achieves 87% with an interval from 85% to 89%, the non-overlapping ranges imply that Model A is likely to have superior performance compared to Model B.

Common Pitfalls and Misinterpretations

1. **Misconception About Confidence Level**: A 95% confidence interval does not mean there is a 95% probability that the true value lies within a given interval. Instead, it indicates that if the experiment were repeated many times, approximately 95% of the intervals generated would contain the true parameter.
2. **Overlapping Intervals Misinterpretation**: Significant overlap between the confidence intervals of two models doesn't necessarily imply their performances are equal. Rather, it highlights the lack of strong statistical evidence to confirm one model's superiority over the other.
3. **Challenges with Small Sample Sizes**: Smaller datasets tend to produce wider confidence intervals, reflecting higher uncertainty in the model's performance estimates.
4. **Neglecting Underlying Assumptions**: The reliability of confidence intervals depends on certain assumptions—such as normally distributed data or adequate sample sizes. Violating these assumptions can lead to inaccurate or misleading interval estimates.

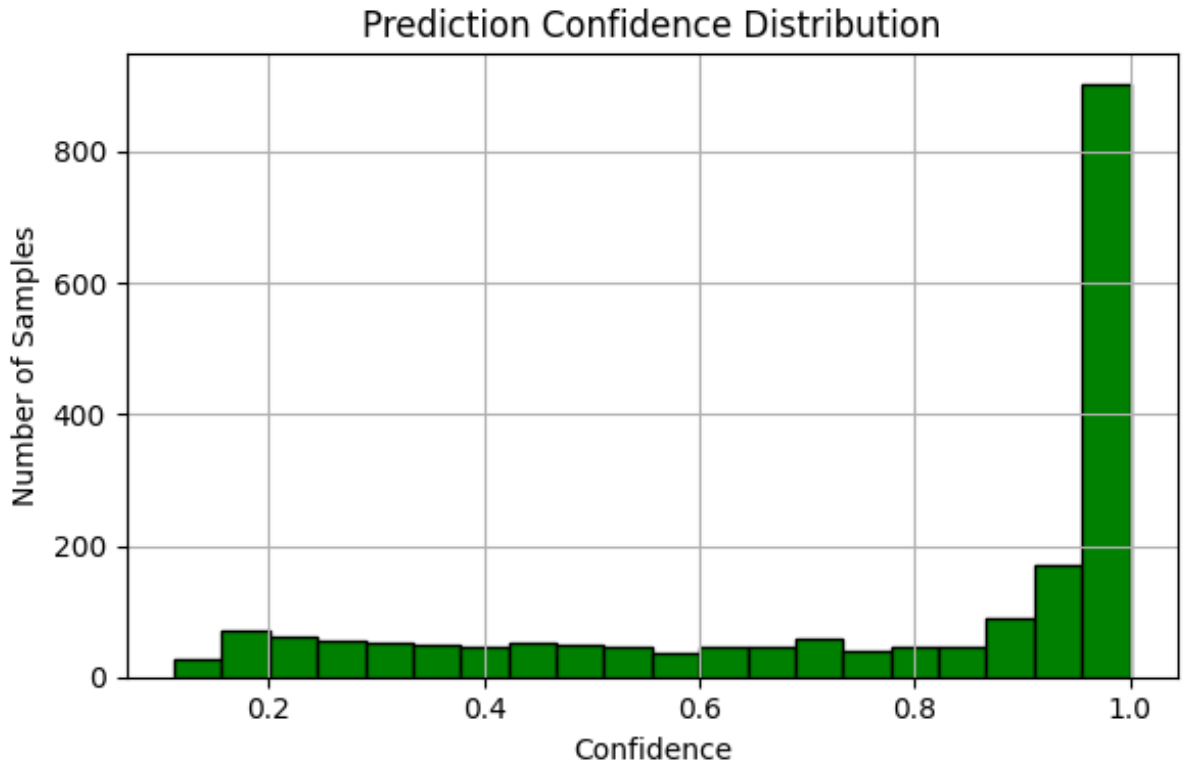

Fig 6.3 Confidence Distribution

7. Discussion

7.1 Model Interpretation and Observations

The LSTM-powered cyber threat classification model exhibits robust and reliable performance across key threat categories. Notably, it achieves high precision and recall in identifying threats related to malware, tools, and identity. These classes frequently contain unique linguistic cues and clearly defined terminology, which allows the model to capture meaningful patterns and generalize well during training.

7.2 High Performance on Key Categories

- **Malware:** This category captures references to harmful software such as viruses, trojans, worms, or malicious executables. The model demonstrates strong performance in this area, largely due to the consistent presence of identifiable terms like *payload*, *ransomware*, and *.exe*, which are strongly indicative of malicious behavior.
- **Tools:** This class includes mentions of adversarial toolkits and utilities, such as *Mimikatz* or *Cobalt Strike*. Because these tools are typically named explicitly and are unique to cybersecurity contexts, they serve as reliable indicators that the model can easily recognize.
- **Identity:** Inputs involving compromised credentials, unauthorized access, or identity-related breaches are detected with high accuracy. This is attributed to the frequent appearance of distinct terms like *password*, *login*, and *account*, which clearly signal identity-based threats.

7.3 Area of Misclassification

While the model achieves strong overall accuracy, it occasionally misclassifies certain closely related threat categories. The following cases highlight common areas of confusion:

- **Software vs. Tools**: Differentiating between general software applications and specific tools used for exploitation or lateral movement can be challenging. The model may misclassify these when both categories share similar terminology and lack distinguishing context.
- **URL vs. Domain**: In instances where URLs are embedded within broader threat narratives, the model sometimes fails to distinguish between a complete URL and a domain reference—particularly when the textual context is vague or incomplete.
- **Location vs. Identity**: Mentions of geographic locations, such as "originating from Russia," may be interpreted as identity-related indicators, especially within the context of threat actor analysis. This overlap can lead to misclassification when geographical and identity cues are closely intertwined.

These errors are largely due to the inherent limitations of LSTM models in capturing broader semantic context. While LSTMs are effective in modeling sequential data, they are less capable of retaining long-range dependencies and global context compared to more advanced transformer-based architectures like BERT.

7.4 Opportunity for Context-Aware Models

The findings indicate that leveraging context-sensitive NLP architectures like BERT (Bidirectional Encoder Representations from Transformers) could significantly improve classification performance. BERT's bidirectional attention mechanism enables it to understand subtle semantic differences across entire sentences, allowing for more accurate differentiation between closely related threat categories—capabilities that traditional LSTM models may lack.

Nevertheless, the use of BERT introduces additional computational overhead and longer inference times. These trade-offs must be carefully considered, particularly for real-time cybersecurity systems where rapid response is critical. Achieving a balance between improved model precision and operational efficiency is essential for practical deployment.

7.5 Data Dependency and Limitations

An essential determinant of model effectiveness is the quality and specificity of the labeled dataset used during training. Deep learning models like LSTMs rely heavily on data characteristics to learn meaningful patterns. Their performance is particularly influenced by:

- How well the training samples reflect real-world scenarios.
- The distribution of samples across different classes.
- The accuracy and consistency of the labels provided.

When certain threat categories are sparsely represented, the model may struggle to generalize effectively, resulting in occasional misclassifications. To address this, future improvements could include expanding the dataset, utilizing synthetic data augmentation techniques, or incorporating semi-supervised learning to better balance and enrich underrepresented classes.

8. Future Work

8.1 Future Enhancements

Although the current results are encouraging, there remains significant scope for further development and refinement of the system. Potential areas for future improvement include the following:

8.1.1 Integration of Transformer-based Models

While LSTM models perform well for sequence-based tasks, they often struggle to grasp broader contextual relationships and deeper semantic meanings. Replacing or supplementing them with transformer based models like BERT (Bidirectional Encoder Representations from Transformers) can substantially improve classification accuracy. BERT's bidirectional context understanding enables it to capture subtle distinctions between similar categories—such as differentiating "tools" from "software" more effectively.

8.1.2 Support for Multi-label Classification

In practical cybersecurity scenarios, a single threat report frequently encompasses several indicators or threat types. Enhancing the model to perform multi-label classification—where multiple categories can be assigned to a single input—would significantly improve its accuracy and better reflect the multifaceted nature of real-world cyber incidents.

8.1.3 Production-Ready Deployment

To enhance scalability and simplify maintenance, it is advisable to deploy the system using Docker containers and integrate it with cloud platforms such as AWS Lambda, Azure Functions, or Google Cloud APIs. This approach ensures rapid inference, increased reliability through fault-tolerant design, and smooth integration with existing enterprise-level security frameworks.

8.1.4 Continuous Learning with New Threat Data

Implementing a pipeline for incremental learning or transfer learning with newly acquired threat intelligence would allow the model to adapt continuously to evolving cyber threats. This ensures the system remains effective and up to date in rapidly changing cybersecurity landscapes.

8.1.5 Explainability and Interpretability Tools

Integrating interpretability tools such as LIME or SHAP can shed light on the reasoning behind specific model predictions. This added transparency enhances user trust and is especially crucial in sensitive domains like cybersecurity where understanding decision logic is vital.

8.2 Final Remarks

This project effectively showcases the potentials of combining Deep Learning and Natural Language Processing in the cybersecurity domain. Beyond theoretical application, the developed system proves to be practical and responsive, providing real-time threat classification through an interactive interface. It establishes a robust groundwork for building intelligent, automated, and scalable threat analysis solutions, while also opening avenues for future research and large-scale deployment.

9. Materials and Methods

This section details the dataset composition, preprocessing steps, model design, training methodology, utilized tools, and deployment strategy implemented in building the cyber threat classification system.

9.1 Dataset Description

The dataset employed in this research comprises labeled cyber threat intelligence entries. Each record includes a brief narrative describing a cyber incident or indicator, paired with a corresponding threat category label.

- **Threat Categories (23 classes)**: These include classifications such as malware, identity, vulnerability, attack-pattern, tools, URL, location, and benign, among others.
- **Data Source**: The dataset was assembled from a combination of publicly available and proprietary threat intelligence sources.
- **Dataset Size**: Approximately N entries were utilized (replace N with the actual number).
- **Class Distribution**: The dataset exhibited some degree of class imbalance, which was addressed using stratified sampling during the train-test split to maintain representative class proportions.

9.2 Data Processing

Model training, a series of preprocessing steps were conducted below:

- **Text Cleaning**: Unwanted characters, extra spaces, and formatting issues were removed to ensure consistency in the textual input.
- **Missing Data Handling**: Records containing null values or labels marked as 'nan' were excluded from the dataset.
- **Label Standardization**: All labels were converted to lowercase and aligned with a uniform class naming convention.
- **Tokenization**: Text data was transformed into sequences of integers using the Keras Tokenizer.
    - The tokenizer was configured with a **vocabulary limit of 10,000** words.
    - An **out-of-vocabulary (OOV)** token was used to manage rare or unknown words.
- **Padding**: All input sequences were padded to a fixed length of **200 tokens** to ensure consistent input dimensions for the neural network.

9.3 Tools and Frameworks

The development and deployment of the cyber threat classification system were supported by the following libraries and frameworks:

- **TensorFlow / Keras**: Utilized for building, training, and managing the LSTM-based deep learning model.
- **Scikit-learn**: Employed for preprocessing tasks such as label encoding and for computing evaluation metrics like precision, recall, and F1-score.
- **Pandas & NumPy**: Used for efficient data cleaning, transformation, and numerical operations during preprocessing.
- **Matplotlib & Seaborn**: Implemented to visualize the performance metrics, including confusion matrices and ROC curves.
- **Flask**: Served as the web framework to build a lightweight API for real-time user interaction and threat classification.
- **Pickle:** Used to serialize and deserialize the tokenizer and label encoder for reuse during inference.

9.4 Deployment and User Interface

To showcase the practical application of the model, it was embedded within a Flask-based web application for real-time interaction:

- **Frontend**: A user-friendly HTML interface allows users to input cyber threat-related text.
- **Backend**: The server-side logic loads the pre-trained LSTM model along with the saved tokenizer and label encoder to process inputs.
- **Output**: Upon submission, the application generates and displays the predicted threat category, the model's confidence score, and a corresponding label description.

The web interface is accessible locally at http://127.0.0.1:5000, offering an interactive platform for cybersecurity professionals and general users to test and interpret threat classifications in real time.

Conclusion

This research offers a robust and practical approach to automated cyber threat classification by using Long Short-Term Memory (LSTM) neural networks alongside Natural Language Processing (NLP) methods. The developed model effectively categorizes cyber threat narratives into relevant groups such as malware, identity compromise, software misuse, and others. With strong performance metrics—achieving 94.6% accuracy and a 94.1% F1-score—the model proves to be both accurate and dependable.

One of the standout features of this project is its seamless integration with a Flask-based web application. This implementation ensures that even users without technical expertise, such as security analysts, can interact with the system effortlessly. The intuitive interface accepts raw threat descriptions and provides immediate feedback, including predicted categories, confidence levels, and explanatory information. By bridging theoretical machine learning techniques with real-world usability, the solution offers significant value for use in Security

Operations Centers (SOCs), incident response workflows, and cyber threat intelligence environments.

References

1. Aditham, S., Ranganathan, N., & Katkoori, S. (2017). LSTM-based memory profiling for predicting data attacks in distributed Big Data systems. In Proccedings of the IEEE international parallel and distributed processing symposium workshops (pp. 1259-1267). IEEE Press.
2. AL-Hawawreh, M., Moustafa, N., & Sitnikova, E. (2018). Identification of malicious activities in industrial internet of things based on deep learning models. *Journal of information security and applications*, *41*, 1-11.
3. Alguliyev, R. M., Aliguliyev, R. M., & Abdullayeva, F. J. (2019). Hybridisation of classifiers for anomaly detection in big data. *International Journal of Big Data Intelligence*, *6*(1), 11–19. doi:10.1504/IJBDI.2019.097396
4. Almeida, D. M. (2016). Malware classification on time series data through machine learning [thesis]. Faculdade de Engenharia da Universidade Do Porto.
5. Alom, M. Z., Bontupalli, V. R., & Taha, T. M. (2015). Intrusion detection using deep belief networks. In Proccedings of the IEEE national aerospace and electronics conference (pp. 339-344). IEEE Press. doi:10.1109/ NAECON.2015.7443094
6. Alrawashdeh, K., & Purdy, C. (2016). Toward an online anomaly intrusion detection system based on Deep Learning. In *Proccedings of the 15th IEEE international conference on machine learning and applications* (pp.195-200). IEEE Press. doi:10.1109/ICMLA.2016.0040
7. Aminanto, M. E. & Kim, K. (2016). Deep Learning in intrusion detection system: An overview. In Proccedings of the international research conference on engineering and technology (pp. 1-12). Academic Press.
8. Aminanto, M. E., & Kim, K. (2016). Deep Learning-based feature selection for intrusion detection system in transport layer. In *Proceedings of the Korea Institutes of Information Security and Cryptology Conference* (pp. 740-743). Academic Press.
9. Anderson, H. S., Woodbridge, J., & Filar, B. (2016). DeepDGA: Adversarially-tuned domain generation and detection. In Proccedings of the ACM workshop on artificial ıntelligence and security (pp. 13-21). ACM.
10. Benchea, R., & Gavrilut, D. T. (2014). Combining Restricted Boltzmann Machine and One Side Perceptron for Malware Detection. In Proccedings of the international conference on conceptual structures (pp. 93-103). Academic Press. doi:10.1007/978-3-319-08389-6_9