National College of Ireland

Lightweight Deep Learning Approaches for Efficient Detection of Cyber-Attack Text

MSc Research Project
Data Analytics

Hemanth Ponduru
Student ID: 23416297

School of Computing
National College of Ireland

Supervisor: Cristina Hava Muntean

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Hemanth Ponduru ……
Student ID:	23416297 ……
Programme:	MSc Data Analytics …… **Year:** 2025 ……
Module:	Research Practicum Part 2 ……
Lecturer:	Cristina Hava Muntean ……
Submission Due Date:	11/12/2025 ……
Project Title:	Lightweight Deep Learning Approaches for Efficient Detection of Cyber-Attack Text ……
Word Count:	1065 …… **Page Count:** 6 ……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	P.Hemanth ……
Date:	10/12/2025 ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Lightweight Deep Learning Approaches for Efficient Detection of Cyber-Attack Text

Hemanth Ponduru
23416297

1. Introduction

This project aims to build an efficient text-classification system to detect cyber-attack–related content from short cybersecurity narratives. The task is challenging because such text is often noisy, brief, and highly variable, so critical attack indicators may appear with different wording and limited context. The pipeline provides a reproducible workflow that preprocesses text, establishes classical machine-learning baselines, advances to deep-learning sequence models, and finally applies an attention-enhanced LSTM to emphasize the most informative tokens for accurate detection.

2. System Specifications

Hardware requirements

- **CPU:** Modern multi-core CPU (recommended: 4+ cores)
- **GPU:** Optional for classical ML; recommended for deep learning (NVIDIA CUDA-capable GPU, 6 GB VRAM or higher for comfortable training)
- **RAM:** Minimum 4 GB (recommended 16 GB+)
- **Storage:** At least 5 GB free (datasets, cached models)

Software requirements

- **OS:** Windows 10/11, Linux (Ubuntu 20.04+), or macOS (DL training is smoother with Linux + NVIDIA GPU)
- **Python:** Python 3.8+ (recommended 3.9/3.10 for wide library compatibility)
- **Environment:** Conda or venv-based isolated environment for reproducibility
- **CUDA/cuDNN (if GPU is used):** CUDA + cuDNN versions compatible with the chosen TensorFlow build (GPU support requires strict version alignment)

3. Required Libraries

The project uses NumPy and Pandas for data handling and preprocessing, and scikit-learn for TF-IDF feature extraction, train–test splitting, baseline machine-learning models, and hyperparameter tuning. XGBoost is included to implement a high-performing boosting

baseline on TF-IDF features. TensorFlow/Keras is used to build and train the deep-learning models, including tokenization, sequence padding, embeddings, LSTM layers, and the attention mechanism. Matplotlib supports plotting of training/validation loss and basic performance visualizations.

The necessary libraries can be installed using the following commands:

pip install numpy==2.3.5 pandas==2.3.3 scikit-learn==1.7.2 matplotlib==3.10.7
pip install xgboost==3.1.2
pip install tensorflow==2.20.0
pip install seaborn==0.13.2-word cloud==1.9.4

4. Dataset description

Dataset source : https://www.kaggle.com/datasets/ramoliyafenil/text-based-cyber-threat-detection (Cyber Threat Dataset: Network, Text & Relation)

The data set in this study is the labelled textual data on cybersecurity incidents, in particular, various entities and patterns of malicious activities. Each record has several attributes like index, text, entities, relations, comments and respective labels which point to certain cyber-related terms like malware, attack-pattern, and time. The dataset format facilitates the recognition of entities by giving the start and end positions of each annotated label in the text, which allows mapping exact textual content and threat indicators. As an example, sentences that explain the behaviour of attacks or mechanisms of infection are annotated using appropriate entity tags that can help understand the context. These annotated data were essential in training and testing the lightweight attention-based models because they enabled the model to learn how to selectively focus on critical threat indicators in sentences. Through this organized data, the study will guarantee that the model that is able to understand semantic connections in the texts of cybersecurity and be computationally efficient throughout the process of extracting features and classifying them.

5. Models Implemented

The modelling approach follows a clear progression to justify performance improvements. It begins with classical machine-learning baselines using TF-IDF features, including Multinomial Naive Bayes as a lightweight probabilistic reference model, Random Forest to evaluate ensemble learning behaviour, and XGBoost as a stronger boosted baseline. It continues on to deep learning with an LSTM model that learns sequence-dependent regularities using tokenized text rather than using sparse n-gram weighting. Lastly, lightweight hybrid architecture is developed based on the LSTM with attention layer in which attention gives greater relevance to the most informative words in every sequence that enhances the quality of detection by targeting important attack indicators.

6. Code snippets and plots:

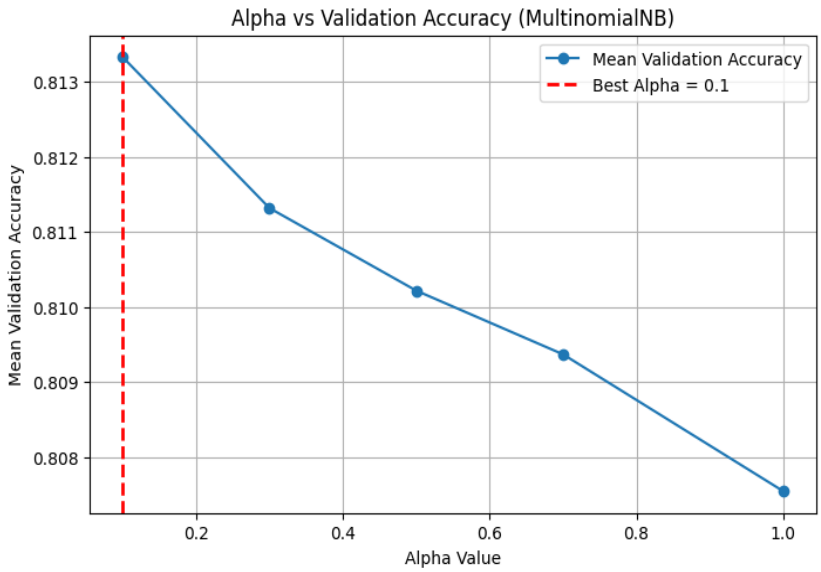
1) Text Cleaning

```
# Initialize tools
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

#  Define preprocessing function
def preprocess_text(text):
    #  Convert to lowercase
    text = str(text).lower()

    #  Remove URLs and email addresses
    text = re.sub(r'http\S+|www\S+|https\S+', '', text)
    text = re.sub(r'\S+@\S+', '', text)

    #  Remove website patterns
    text = re.sub(r'\S+\[\.\]\S+|\S+\.com|\S+\.net|\S+\.org', '', text)

    #  Remove digits and punctuation and special characters
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'[^a-z\s]', '', text)

    #  Remove extra spaces
    text = re.sub(r'\s+', ' ', text).strip()

    #  Tokenize
    tokens = word_tokenize(text)

    #  Lemmatize and remove stopwords
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]

    #  Join tokens back into a single string
    cleaned_text = ' '.join(tokens)

    return cleaned_text

#  Apply cleaning to dataframe column
df['text'] = df['text'].apply(preprocess_text)

#  result
print(" Advanced text preprocessing completed successfully!")
df.head()
```

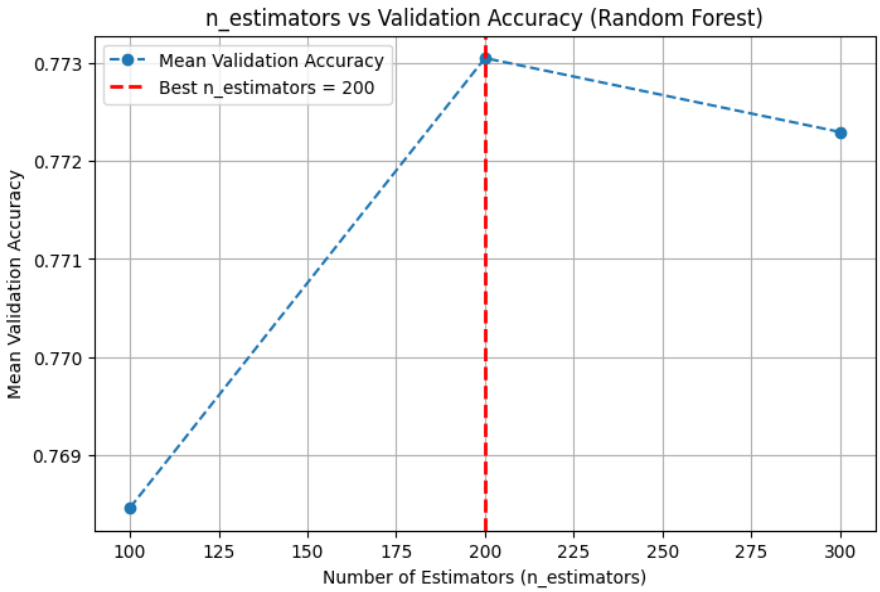
2) Splitting data

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

X = df['text']
y = df['label']

# TF-IDF Vectorization
tfidf = TfidfVectorizer(max_features=5000, ngram_range=(1,2))
X_tfidf = tfidf.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42, stratify=y)
```

3) Naive Bayes hyperparameter plot



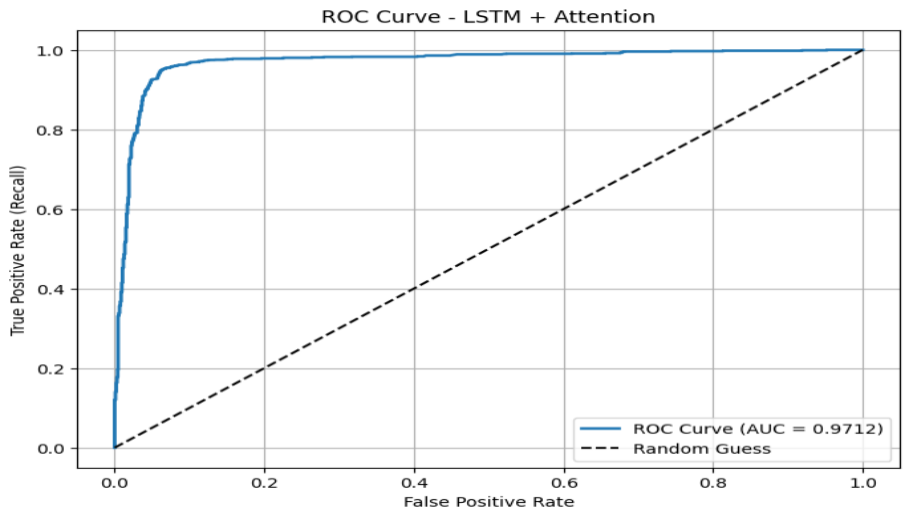
4) Random Forest hyperparameter plot



5) LSTM + Attention Model

```python
#  Custom Attention Layer
class Attention(layers.Layer):
    def __init__(self, **kwargs):
        super(Attention, self).__init__(**kwargs)

    def build(self, input_shape):
        self.W = self.add_weight(name="att_weight", shape=(input_shape[-1], 1),
                                 initializer="normal")
        self.b = self.add_weight(name="att_bias", shape=(input_shape[1], 1),
                                 initializer="zeros")
        super(Attention, self).build(input_shape)

    def call(self, x):
        e = tf.keras.backend.tanh(tf.keras.backend.dot(x, self.W) + self.b)
        a = tf.keras.backend.softmax(e, axis=1)
        output = x * a
        return tf.keras.backend.sum(output, axis=1)

#  Build Model
EMBED_DIM = 128
LSTM_UNITS = 128

model = Sequential([
    Embedding(input_dim=VOCAB_SIZE, output_dim=EMBED_DIM, input_length=MAX_LEN),
    LSTM(LSTM_UNITS, return_sequences=True),
    Attention(),
    Dropout(0.4),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')])

model.compile(optimizer=Adam(learning_rate=1e-3),
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.summary()
```

6) ROC Curve plot LSTM + Attention



7) Comparison of evaluation metrics

Table 1. Model Performance Comparison

Model	Accuracy	Precision	Recall	F1 Score
Naive Bayes	82.26%	80.82%	84.30%	82.52%
Random Forest	79.72%	91.24%	65.45%	76.22%
XGBoost	86.56%	91.81%	80.09%	85.55%
LSTM	91.52%	91.82%	91.03%	91.43%
LSTM + Attention	**94.31%**	**93.40%**	**95.29%**	**94.33%**

7. Conclusion:

The project provides a repeatable pipeline to identify cyber-attack text, including preprocessing, feature extraction, model training and fair comparison through consistent evaluation. The LSTM + Attention model is the most efficient of all models since the sequence learning is incorporated in the model, as well as, the token-level focusing, which enables the model to consider the context, and, at the same time, to prioritize the most significant indicators of an attack. The workflow is conducive to repeated experimentation, and can be further expanded in future research by incorporating more powerful language models, enhancing robustness to noisy text, and incorporating more cybersecurity indicators to be applied in the real world.

References:

1. https://www.tensorflow.org/
2. https://scikit-learn.org/stable/
3. https://xgboost.readthedocs.io/en/stable/
4. https://pandas.pydata.org/docs/
5. https://numpy.org/doc/