

RESEARCH ARTICLE

A BERT-Enhanced Exploration of Web and Mobile Request Safety Through Advanced NLP Models and Hybrid Architectures

SALMI SALIM  **AND OUGHDIR LAHCEN**

National School of Applied Sciences, Engineering Systems and Applications Laboratory, Sidi Mohamed Ben Abdellah University, Fez 30000, Morocco

Corresponding author: Salmi Salim (salim.salmi@usmba.ac.ma)

ABSTRACT In the rapidly evolving landscape of digital technology, the security of web and mobile applications stands paramount. As these platforms become increasingly integrated into our daily lives, the need for robust safety measures becomes imperative. This research paper delves into the intricate realm of web and mobile request safety, unraveling a multi-faceted exploration that combines traditional feature engineering with state-of-the-art machine learning models. Beginning with foundational models like TextCNN and TextRNN, we scrutinize their effectiveness in discerning the safety of requests. Advancing our investigation, we delve into the capabilities of sophisticated architectures, including Bidirectional LSTMs, DistilBERT, and RoBERTa. Beyond individual assessments, we introduce hybrid models that synergize the strengths of various approaches, establishing a comprehensive defense against emerging security threats. Throughout this research, we navigate the intricacies of model training, evaluation, and performance metrics. From accuracy and precision to recall and confusion matrices, each metric paints a nuanced picture of the efficacy of these models in ensuring the safety of web and mobile interactions. In a world where cyber threats loom large, the significance of this research lies not only in its technical contributions but also in its practical implications. By providing insights into innovative strategies for enhancing the security and resilience of digital applications, this paper contributes to the ongoing discourse on fortifying the digital infrastructure.

INDEX TERMS Web and mobile security, request safety, machine learning, natural language processing, cybersecurity.

I. INTRODUCTION


In the dynamic realm of web and mobile security, the imperative for robust threat detection mechanisms has never been more pronounced. The escalating frequency and sophistication of cyber threats underscore the urgency for innovative strategies that transcend conventional models [1]. In response, we present a pioneering solution that harnesses the potency of hybrid models, uniting the capabilities of convolutional neural networks (CNNs), recurrent neural networks (RNNs), and cutting-edge transformer models like RoBERTa.

The escalating intricacy of cyber threats targeting web and mobile platforms has propelled the need for adaptive and

intelligent security measures. Conventional methodologies often fall short in holistically identifying and countering diverse threat vectors, prompting the exploration of advanced models adept at comprehending the nuances of textual data.

Our journey commences by delving into diverse deep learning architectures designed for text classification. From the early adoption of TextCNN, excelling in capturing local patterns, to the integration of TextRNN, renowned for its sequential understanding capabilities, each model contributes to an intricate tapestry of threat detection mechanisms. Additionally, we explore the potential of Bidirectional LSTM models, enhancing contextual understanding by considering both past and future information in the input sequence.

The emergence of transformer models has revolutionized natural language processing. Our approach incorporates state-of-the-art transformer architectures such as RoBERTa,

The associate editor coordinating the review of this manuscript and approving it for publication was Arianna Dulizia .

facilitating a deeper semantic understanding of textual data. The fusion of transformer embeddings with traditional models forms the bedrock of our hybrid architecture, amalgamating the strengths of local and global contextual information.

Acknowledging the necessity for a holistic threat assessment, our models extend beyond textual content to incorporate additional features. Variables like sentence length, punctuation usage, and custom-engineered indicators provide the models with a broader context, elevating their discernment between safe and potentially malicious requests.

The apex of our exploration culminates in the development of a Hybrid Model, seamlessly integrating TextCNN and RoBERTa. This innovative approach seeks to exploit the synergies between convolutional and transformer-based architectures, offering a comprehensive understanding of both local and global textual patterns.

Our hybrid model undergoes meticulous training with a laser focus on optimizing performance metrics such as accuracy, precision, and recall. The evaluation process entails rigorous testing on a diverse dataset, culminating in a comprehensive analysis of the model's proficiency in accurately categorizing requests as safe or potentially harmful.

This advanced threat detection system holds immense promise for fortifying the security posture of web and mobile applications. By seamlessly integrating with existing security frameworks, our solution furnishes an additional layer of defense adept at adapting to emerging threats and safeguarding sensitive user data.

In an era where cyber threats evolve in sophistication, our hybrid approach to threat detection stands as a beacon of innovation. The union of traditional and transformer-based models heralds new horizons for web and mobile security, promising a safer digital environment for users and organizations alike. As we steadfastly evolve, our commitment to remain at the forefront of cybersecurity ensures a secure and resilient digital future.

II. LITERATURE REVIEW

applications have ended up an indispensable portion of our day by day lives, taking care of touchy information. Subsequently, they have become profitable targets for potential attacks. To protect these applications against both known and obscure dangers, a classification approach is utilizing the well-established DATASET Hypertext Transfer Protocol. The authors in [2] presented a character-level neural arrangement of convolution on the HTTP data set. This includes extricating highlights from HTTP requests and categorizing them as typical or atypical. The objective is to realize precision, speed, and cost-effectiveness within the execution of a Web Application Firewall. This specialized firewall demonstrates to be more proficient in ensuring applications against known attacks compared to ordinary firewalls. Utilizing the same dataset, authors in [3] utilized an assortment of machine learning strategies, such as Arbitrary Timberland, Choice Tree, AdaBoost, Calculated

Relapse, Stochastic Slope Plummet, and Naïve Bayes. These calculations were connected to classify HTTP demands, and an include-choice handle was conducted to distinguish the five most significant highlights for identifying web assaults. The evaluation of these procedures was based on the accuracy, rate, and F measures. The comes about uncovered that all procedures displayed tall exactness, review, rate, and F-measures, with the exemption of Naïve Bayes.

Authors in [4] conducted an overview that digs into various machine learning procedures utilized in Web Interruption Location Frameworks. Their center was on the engineering of web interruption location frameworks based on machine learning standards. To demonstrate and approve their discoveries. In their work, authors [5] dove into the method of ordinary web demands from the client to the server. They particularly centered on classifying web assaults and categorizing them based on their types. Additionally, the Authors investigated the methodology employed by assailants in a five-step prepare. The beginning step includes gathering data almost the vulnerabilities of the application. Along these lines, this information is utilized to pick up get to the net application. The third step envelops getting chairperson benefits, taken after by the fourth step, which rotates around keeping up get to. At last, the aggressors execute a web application attack through a DoS assault to block other clients from getting to the application.

Authors in [6] connected machine learning methods to identify dazzle Cross-Site Scripting assaults. The method of reasoning behind this approach is that existing XSS discovery procedures regularly battle to recognize dazzle XSS assaults. To address this restriction, the Authors utilized machine learning to perceive noxious behavior inside their dataset. Authors in [7] presented an arrangement for classifying SQL infusions inside SQL questions. They utilized the Gap-Weighted String calculation to distinguish shared characters in inquiry strings as a yield for a closeness metric. Subsequently, a Bolster Vector Machine was prepared on these similitude measurements to classify obscure test queries, accomplishing an amazing precision. In a related setting, authors in [8] conducted a comparison between employing a one-class SVM exclusively to classify HTTP demands and joining a feature extraction strategy. The include extraction method involved extricating 10 numeric highlights from the complete ask, counting HTTP strategy, headers, inquiry, URL, and body. Among these numeric highlights were tallies for the number of letters, digits, add up to length, and non-alphanumeric characters. The classification comes about demonstrated that utilizing to include extraction strategy with the one-class SVM altogether upgraded the discovery results.

Authors in [9] investigated to utilize of two classifiers, to be specific choice tree and irregular timberland, for recognizing (DDoS) assaults. Their information source was the KDDCup'99 dataset, and through the disarray matrix, they found that the D tree classifier given the foremost favorable arrangement compared to the arbitrary woodland.

In a comparable setting, authors in [10] tended to the location of Disseminated Dissent of Benefit Assaults utilizing three classification methods: Naïve Bayes, Multilayer Perceptron, and Irregular Timberland. The MLP stood out due to different execution highlights such as Preparing Calculation, Neural Network Engineering, and Exchange Work, accomplishing the most noteworthy precision. Authors in [11] proposed a novel profound component based on a normal arrange for precisely finding and classifying DDoS assaults in multi-layer applications. This component utilized feedforward back-propagation, accomplishing an amazing exactness. Moreover, authors in [12] executed a web lumberjack model framework based on a gathering algorithm to illustrate the plausibility of deriving PINs or passwords entered quietly by clients of portable phones in mobile web applications. The internet lumberjack altogether moved forward in precision compared to the past. Its usage included information collection, highlight extraction, and show preparing.

Authors in [13] propose a robust and effective approach for detecting Arabic hate speech on Twitter, aiming to address a notable gap in existing literature. Their methodology introduces the Arabic BERT-Mini Model (ABMM), leveraging bidirectional encoder representations from transformers (BERT) to analyze Twitter data. The ABMM model classifies tweets into three categories: normal, abuse, and hate speech. Through rigorous experimentation and comparison with state-of-the-art approaches, the ABMM model demonstrates exceptional performance, achieving an impressive accuracy score of 0.986. This study contributes significantly to the field of Arabic hate speech detection and underscores the importance of tailored approaches for addressing language-specific challenges in social media content moderation.

Table 1 presents a concise summary of research papers in the field of web application security. Each row outlines a different study, including authorship, methodologies, datasets, key findings, and limitations. This overview offers a quick insight into various approaches and challenges addressed in the literature.

III. TextCNN: CONVOLUTIONAL NEURAL NETWORKS FOR EFFICIENT TEXT CLASSIFICATION

TextCNN is a convolutional neural network architecture designed for text classification tasks. It utilizes convolutional layers to capture local patterns and hierarchical representations of textual data [14]. The architecture typically consists of the following components:

- **Embedding Layer:** Given a sequence of words in a text, each word is represented as a dense vector through an embedding layer. Let x_i represent the embedding vector for the i^{th} word in the input sequence.
- **Convolutional Layer:** Convolutional operations are applied to the embedded representations to capture local patterns. A filter of size h is applied to k consecutive words, producing a feature map c_i through the

convolution operation:

$$c_i = f(W \cdot x_{i:i+h-1} + b)$$

where $x_{i:i+h-1}$ is the concatenation of embedding vectors from i to $i + h - 1$, W is the filter matrix, b is the bias term, and f is the activation function (commonly ReLU).

- **Max Pooling Layer:** Max pooling is performed over the feature maps to extract the most salient information. The maximum value p_i is taken from each feature map c_i :

$$p_i = \max(c_i)$$

- **Fully Connected Layer:** The pooled features are flattened and passed through one or more fully connected layers to capture global dependencies:

$$y = g(W' \cdot p + b')$$

where W' and b' are the weight matrix and bias term of the fully connected layer, and g is the activation function.

- **Output Layer:** The final output layer uses softmax activation for multi-class classification:

$$\hat{y} = \text{softmax}(y)$$

- **Training Objective:** The model is trained to minimize the cross-entropy loss between the predicted distribution \hat{y} and the true distribution y :

$$\text{Loss} = - \sum_i y_i \cdot \log(\hat{y}_i)$$

where y_i is the ground truth probability distribution for the class i , and \hat{y}_i is the predicted probability.

This architecture is effective for various natural language processing tasks, such as sentiment analysis and text categorization, due to its ability to capture local and global contextual information in the input text.

IV. TextRNN AND BiLSTM: UNRAVELING SEQUENTIAL DEPENDENCIES

TextRNN, short for Text Recurrent Neural Network, is a neural architecture tailored for processing sequential data in natural language. It employs recurrent layers to capture contextual dependencies across words in a sentence. The core idea involves passing information from previous time steps to the current one, enabling the model to understand the sequential nature of textual information [15].

$$h_t = \text{RNN}(x_t, h_{t-1})$$

Here, x_t represents the input at time step t , h_t is the hidden state at time t , and RNN is the recurrent function. The model learns to update its hidden state based on both the current input and the information from the previous time step.

TABLE 1. Overview of research papers in web application security based on literature review.

Authors	Methodologies	Dataset	Year	Key Findings	Limitations
[2]	Character-level CNN	HTTP Dataset	2018	Efficient Web Application Firewall using CNN for HTTP requests classification.	Limited evaluation on diverse datasets, potential overfitting due to dataset bias.
[3]	Various ML techniques	HTTP Dataset	2017	Evaluation of ML techniques for HTTP requests classification, identifying significant features for web attack detection.	Lack of comparison with state-of-the-art methods, performance may vary on different datasets.
[4]	Survey	N/A	2016	Overview of ML techniques in Web Intrusion Detection Systems (WIDS) architecture.	Relies on existing literature, may not cover all recent advancements in the field.
[5]	Classification	Web attack types	2016	Classification of web attacks and investigation of attacker methodology.	Limited dataset size, generalization to real-world scenarios may be challenging.
[6]	ML for XSS detection	N/A	2018	ML approach for detecting blind Cross-Site Scripting (XSS) attacks.	Limited evaluation on diverse XSS attack scenarios, potential performance degradation in noisy environments.
[7]	SQL injection detection	SQL queries	2018	Classification of SQL injections using Gap-Weighted String algorithm and SVM.	Reliance on specific features, may not generalize well to various SQL injection types.
[8]	One-class SVM	HTTP requests	2017	Comparison of one-class SVM with feature extraction for HTTP request classification.	Sensitivity to parameter tuning, performance may degrade with imbalanced datasets.
[9]	Decision Tree & RF	KDDCup'99	2017	Detection of Distributed Denial of Service (DDoS) attacks using decision tree classifier.	Limited scalability to large-scale DDoS attacks, potential false positives in dynamic network environments.
[10]	Naïve Bayes, MLP, RF	N/A	2017	Detection of Distributed Denial of Service (DDoS) attacks using MLP as the most effective classifier.	Sensitivity to hyperparameters, may require extensive computational resources for training.
[11]	Deep Learning	Multi-layer apps	2020	Novel deep feature-based method for DDoS attack detection and classification.	Potential overfitting due to complex model architecture, interpretability may be challenging.
[12]	Web logger model	Mobile web apps	2017	Implementation of web logger model for deriving PINs/passwords entered in mobile web applications.	Limited applicability to specific mobile platforms, potential privacy concerns with data logging.

A. BiLSTM (BIDIRECTIONAL LONG SHORT-TERM MEMORY)

BiLSTM is an extension of the traditional LSTM (Long Short-Term Memory) architecture. It incorporates information from both past and future time steps by utilizing two separate LSTM layers - one processing the sequence in forward order and the other in reverse [16]. This bidirectional approach allows the model to capture contextual information from both directions, enhancing its understanding of the input sequence, Figure 1.

$$\vec{h}_t = \text{LSTM}_{\text{forward}}(x_t, \vec{h}_{t-1})$$

$$\overleftarrow{h}_t = \text{LSTM}_{\text{backward}}(x_t, \overleftarrow{h}_{t+1})$$

$$h_t = [\vec{h}_t, \overleftarrow{h}_t]$$

Here, \vec{h}_t and \overleftarrow{h}_t represent the hidden states from the forward and backward LSTMs, respectively. The final hidden state h_t is obtained by concatenating the two.

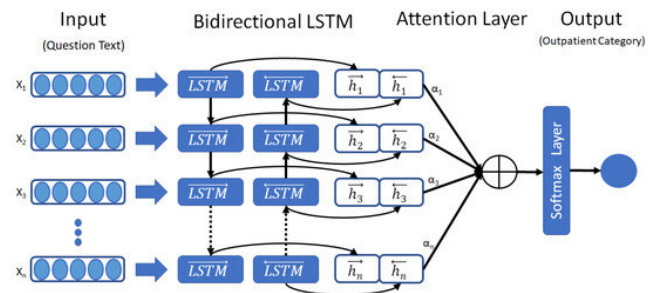


FIGURE 1. Bidirectional long-short term memory (LSTM) with attention mechanism.

Both TextRNN and BiLSTM play crucial roles in capturing sequential dependencies, enabling models to excel in tasks like sentiment analysis, named entity recognition, and other sequence-based natural language processing tasks.

V. RoBERTa AND DistilBERT: UNVEILING A ROBUSTLY OPTIMIZED BERT APPROACH

RoBERTa, an acronym for Robustly optimized BERT approach, represents a significant advancement in natural language processing (NLP) and builds upon the transformer architecture. Introduced by Facebook AI in 2019, RoBERTa is designed to enhance the original BERT model's performance through various key features and optimizations [17].

RoBERTa introduces dynamic masking during pre-training, departing from the static masking strategy used in BERT. This dynamic approach involves randomly masking different sets of tokens in each training iteration, facilitating improved generalization and a deeper understanding of contextual information. RoBERTa benefits from a more extensive and diverse training dataset compared to its predecessor, BERT. The model leverages a larger corpus of text data, enabling it to capture a broader range of linguistic patterns and nuances.

The Next Sentence Prediction (NSP) task, present in BERT, is omitted in RoBERTa's pre-training. This modification aims to enhance the model's understanding of sentence relationships by focusing on other relevant pre-training objectives. RoBERTa carefully tunes hyperparameters during training, including batch size and learning rate, to optimize performance. This meticulous adjustment contributes to the model's robustness and efficiency.

DistilBERT, short for Distill-BERT, is a compact and computationally efficient variant of the original BERT (Bidirectional Encoder Representations from Transformers) model. Introduced by Hugging Face in 2019, DistilBERT aims to distill the knowledge from a pre-trained BERT model into a smaller architecture, maintaining performance while significantly reducing the number of parameters [18].

- Key Features and Characteristics [19]
 - Knowledge Distillation: DistilBERT employs knowledge distillation, a process where it learns from a pre-trained BERT model with a larger number of parameters. This allows DistilBERT to inherit the essential representation learning capabilities of BERT while being more lightweight.
 - Reduced Model Size: One of the primary goals of DistilBERT is to create a smaller model for faster inference and reduced memory requirements. By distilling knowledge from BERT, DistilBERT achieves a substantial reduction in the number of parameters while preserving the critical aspects of the original model.
 - Retained Transformer Architecture: DistilBERT retains the transformer architecture, which is the backbone of BERT. This architecture facilitates capturing contextual information in a bidirectional manner, allowing the model to understand the relationships between words in a sequence.

A. ATTENTION MECHANISM IN TRANSFORMER

Attention is the cognitive process of selectively focusing on one or more objects while ignoring others. The attention mechanism emerged to enhance neural machine translation systems based on autoencoders in NLP [20]. Later, this mechanism or its variations have been employed in various applications, including computer vision, speech processing, etc. In machine translation applications, it's common to use autoencoders based on an LSTM architecture [21]. Each time the proposed model generates a sentence, it looks for a set of positions in the hidden layers of the encoder where the most relevant information is available. This idea is called "Attention." The bidirectional LSTM used in Figure 1 generates a sequence of annotations (h_1, h_2, \dots, h_T) for each input phrase. All vectors h_1, h_2, \dots , etc., used in the process are the concatenation of the hidden states' output from the LSTMs, both forward and backward, in the encoder [22].

In the transformer architecture, the attention mechanism is pivotal for capturing contextual dependencies [23]. The attention score $\text{Attention}(Q, K, V)$ is computed using scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V$$

Here, Q, K , and V represent the query, key, and value matrices, respectively, while d_k is the dimensionality of the key vectors. This mechanism allows RoBERTa to assign different weights to different parts of the input sequence, contributing to its ability to capture intricate contextual information in natural language.

RoBERTa's incorporation of dynamic masking, a larger training dataset, and modifications to pre-training objectives showcases its robustness and effectiveness in various NLP tasks, solidifying its standing as a state-of-the-art language model. Researchers and practitioners often leverage RoBERTa for fine-tuning on specific NLP tasks or use its pre-trained representations to boost performance in downstream applications [24].

VI. PROPOSED METHODOLOGY

The proposed methodology encompasses a comprehensive approach to binary text classification, integrating four distinct deep learning models: TextCNN, TextRNN, Bidirectional LSTM, RoBERTa-based Hybrid Model, and a DistilBERT-based Model. The primary objective is to categorize textual data into 'Not Safe' and 'Safe' classes. Each model contributes a unique architecture: TextCNN with convolutional layers, TextRNN with recurrent structures, Bidirectional LSTM with bidirectional memory cells, the RoBERTa-based Hybrid Model combining TextCNN with RoBERTa embeddings, and the DistilBERT-based Model. This diverse ensemble aims to capture intricate patterns in textual information. The subsequent sections elaborate on the specific architectures, training procedures, and evaluation metrics for each model, culminating in a robust framework for binary text classification.

VII. DATASET

The dataset under consideration, retrieved from Kaggle and centered around web and mobile security with a focus on user requests, is characterized by its comprehensive inclusion of request payload data. A pivotal feature within this dataset is the “is Safe” field, providing crucial insights into the safety of each individual request for the application [25]. The “is Safe” field serves as an indicator, effectively conveying whether a particular request is deemed secure or poses potential risks. Specifically, a value of “False” in the “is Safe” field designates that the associated request should be flagged as unsafe and consequently blocked. This determination is based on a meticulous examination of the fields within each request. Any instance where malicious input is detected, aligning with the OWASP Top 10 attacks, results in the classification of the entire request as unsafe. As we embark on the exploration and visualization of this dataset, our goal is to unravel patterns and insights that contribute to a robust understanding of security dynamics in the realm of web and mobile applications, emphasizing the critical role of user request safety in safeguarding digital environments.

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 23 columns):
#   Column                               Non-Null Count  Dtype
---  -
0   req/baseUrl                           1000 non-null   object
1   req/body/note/title                    1000 non-null   object
2   req/body/note/desc                     1000 non-null   object
3   req/fresh                              1000 non-null   bool
4   req/headers/host                       1000 non-null   object
5   req/headers/user-agent                 1000 non-null   object
6   req/headers/content-type               1000 non-null   object
7   req/headers/org_id                     1000 non-null   object
8   req/headers/user_session_id            1000 non-null   object
9   req/headers/accept                     1000 non-null   object
10  req/headers/content-length              1000 non-null   int64
11  req/headers/user/name                   1000 non-null   object
12  req/headers/user/role                   1000 non-null   object
13  req/hostname                           1000 non-null   object
14  req/ip                                  1000 non-null   object
15  req/originalUrl                         1000 non-null   object
16  req/path                                 1000 non-null   object
17  req/protocol                            1000 non-null   object
18  req/secure                              1000 non-null   bool
19  req/stale                               1000 non-null   bool
20  req/subdomains/0                        1000 non-null   object
21  req/xhr                                  1000 non-null   bool
22  isSafe                                  1000 non-null   bool
dtypes: bool(5), int64(1), object(17)
memory usage: 145.6+ KB
```

FIGURE 2. Dataset information.

As shown in figure 2 The dataset comprises 1000 entries with 23 columns. Notably, the ‘is Safe’ column, our primary focus, is a boolean variable indicating the safety status of web or mobile requests. Other key columns include request details such as base URL, note title and description, headers information, user details, and various request attributes. The dataset provides a diverse set of features to analyze and model, with ‘is Safe’ serving as the binary target variable for safety classification. The entries are complete with no missing values, ensuring the dataset’s reliability for exploring and building models related to request safety.

The “is Safe” column in the dataset contains 1000 entries, indicating the total number of observations. It is a categorical variable with two unique values, namely “True” and presumably “False” or “0” and “1.” The most frequently occurring value in this column is “True,” appearing 572 times out of the 1000 entries. This suggests that the majority of instances in the dataset are labeled as “True” in the “is Safe” category. Figure 3 illustrate distribution count of dataset.

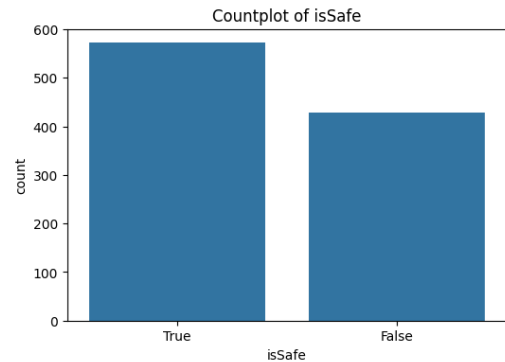


FIGURE 3. Safe and unsafe distribution.

VIII. TextCNN, TextRNN, BIDIRECTIONAL LSTM

The proposed methodology involves three distinct models for binary text classification: TextCNN, TextRNN, and Bidirectional LSTM.

For the TextCNN model, the process, as shown in algorithm 1, begins with the initialization of an embedding matrix, followed by the application of 1D convolutional layers with various filter sizes. Global max pooling is performed on the convolutional outputs, and the results are concatenated. The concatenated output undergoes a dense layer with ReLU activation, followed by dropout for regularization. The final layer consists of a dense layer with sigmoid activation for binary classification. During training, the model utilizes binary cross-entropy loss and the Adam optimizer on the training dataset. Prediction on the test set is followed by the computation of accuracy, precision, recall, and the confusion matrix.

Algorithm 2 present the TextRNN model, the procedure initiates with the embedding of input sequences and the application of Long Short-Term Memory (LSTM) units. A dense layer with sigmoid activation concludes the architecture. The model is trained using binary cross-entropy loss and the Adam optimizer on the training dataset, and predictions are made on the test set. Evaluation metrics include accuracy, precision, recall, and the confusion matrix.

Lastly, algorithm 3 present the Bidirectional LSTM model involves embedding input sequences, applying Bidirectional LSTM layers with return sequences, and concluding with a dense layer with sigmoid activation. The model is trained with binary cross-entropy loss and the Adam optimizer, utilizing the training dataset. Predictions are made on the test set, and evaluation metrics, including accuracy, precision, recall, and the confusion matrix, are computed.

In summary, the proposed methodology encompasses the initialization, training, and evaluation steps for three distinct models, providing a comprehensive approach to binary text classification with varying architectures.

Algorithm 1 TextCNN Model

Data: Embedding Matrix E , Input Length L , Filters F , Filter Sizes $[K_1, K_2, \dots, K_n]$, Output Dimension D , Dropout Rate p
Result: Probability Vector P for binary classification
Initialization:
 $X \leftarrow \text{Embedding}(E, L)$
 $H_i \leftarrow \text{Conv1D}(X, F, K_i, \text{activation} = 'relu')$ for $i = 1$ to n do
 end
 $P_i \leftarrow \text{GlobalMaxPooling1D}(H_i)$ for $i = 1$ to n do
 end
 $P \leftarrow \text{Concatenate}(P_1, P_2, \dots, P_n)$
 $P \leftarrow \text{Dense}(P, D, \text{activation} = 'relu')$
 $P \leftarrow \text{Dropout}(P, p)$
 $P \leftarrow \text{Dense}(P, 1, \text{activation} = 'sigmoid')$
Model Training:
 Loss: BCE, Optimizer: Adam
 $P \leftarrow \text{Train}(X_{\text{train}}, y_{\text{train}}, \text{epochs}, \text{batch_size})$
Model Prediction:
 Testing: $y_{\text{pred}} \leftarrow \text{Predict}(X_{\text{test}})$
Evaluation Metrics:
Result: Accuracy, Precision, Recall, Confusion Matrix['Not Safe', 'Safe']

Algorithm 2 TextRNN Model

Data: Embedding Matrix E , Input Length L , LSTM Units U , Output Dimension D
Result: Probability Vector P for binary classification
Initialization:
 $X \leftarrow \text{Embedding}(E, L)$
 $H \leftarrow \text{LSTM}(X, U)$
 $P \leftarrow \text{Dense}(H, D, \text{activation} = 'sigmoid')$
Model Training:
 Loss: BCE, Optimizer: Adam
 $P \leftarrow \text{Train}(X_{\text{train}}, y_{\text{train}}, \text{epochs}, \text{batch_size})$
Model Prediction:
 Testing: $y_{\text{pred}} \leftarrow \text{Predict}(X_{\text{test}})$
Evaluation Metrics:
 Accuracy, Precision, Recall, Confusion Matrix

Algorithm 3 Bidirectional LSTM Model

Data: Embedding Matrix E , Input Length L , LSTM Units U , Output Dimension D
Result: Probability Vector P for binary classification
Initialization:
 $X \leftarrow \text{Embedding}(E, L)$
 $H_1 \leftarrow \text{BidirectionalLSTM}(X, U, \text{return_sequences} = \text{True})$
 $H_2 \leftarrow \text{BidirectionalLSTM}(H_1, U)$
 $P \leftarrow \text{Dense}(H_2, D, \text{activation} = 'sigmoid')$
Model Training:
 Loss: BCE, Optimizer: Adam
 $P \leftarrow \text{Train}(X_{\text{train}}, y_{\text{train}}, \text{epochs}, \text{batch_size})$
Model Prediction:
 Testing: $y_{\text{pred}} \leftarrow \text{Predict}(X_{\text{test}})$
Evaluation Metrics:
Result: Accuracy, Precision, Recall, Confusion Matrix['Not Safe', 'Safe']

IX. DistilBERT-BASED MODEL

The proposed methodology presented in algorithm 4 involves the implementation and evaluation of a DistilBERT-based

model for sequence classification. The algorithm is presented in a structured manner, encompassing model initialization, data loading, training loop, evaluation, and metrics calculation.

The algorithm starts with the initialization phase, where the DistilBERT model is instantiated using the 'distilbert-base-uncased' pre-trained weights. Additionally, the Adam optimizer and Binary Cross-Entropy with Logits Loss criterion are initialized to facilitate model training.

Following initialization, the algorithm progresses to data loading. The DistilBERT tokenizer is employed to preprocess the data, and DataLoader instances for both training and testing are created. This step ensures that the data is prepared and organized appropriately for model training and evaluation.

The training loop constitutes a crucial component of the methodology. It involves iterating over epochs and mini-batches from the training Data Loader. The model is set to training mode, and for each batch, input tensors (input_ids, attention_mask, labels) are retrieved and moved to the specified device. The optimizer's gradients are zeroed, and the model is forward-passed to obtain outputs. The loss is computed, back propagated, and optimizer parameters updated. The average loss per epoch is printed for monitoring training progress.

Subsequently, the evaluation phase is introduced. The model is set to evaluation mode, and predictions are obtained for the test Data Loader. The sigmoid function is applied to the logits to obtain probabilities, and the predictions are collected.

Finally, metrics calculation involves converting the predicted probabilities to binary predictions based on a threshold of 0.5. The predicted and true labels are then used to compute evaluation metrics such as accuracy, precision, recall, and F1 score.

This proposed methodology ensures a systematic approach to training, evaluating, and analyzing the performance of a DistilBERT-based model for sequence classification. It covers key aspects of model implementation and assessment, providing a comprehensive framework for the task at hand.

X. HYBRIDE RoBerta+ textCNN

Figure 4 show the proposed methodology, involves a comprehensive process for training and evaluating a hybrid model that combines a TextCNN model and a pre-trained RoBERTa model for binary classification of textual data. The overall approach is outlined in Algorithm 5.

To begin with, the raw dataset is loaded and preprocessed. The target variable is encoded using a Label Encoder, and the dataset is split into training and testing sets. The textual data is tokenized using the RoBERTa tokenizer, enabling the conversion of text into a format suitable for model input. This step ensures the compatibility of the data with the subsequent model architecture.

The TextCNN model is then defined with specific parameters such as embedding dimension, number of filters, filter

Algorithm 4 DistilBERT Model

Data: Tokenizer, Model, Optimizer, Criterion, Train Loader, Test Loader, Epochs, Device

Initialization:

```
Model ← DistilBertForSequenceClassification('distilbert-base-uncased')
Optimizer ← Adam(Model.parameters())
Criterion ← BCEWithLogitsLoss()
```

Data Loading:

Load and preprocess data using DistilBERT tokenizer
Create DataLoader for training and testing

Training Loop:

```
for epoch ← 1 to epochs do
  Model.train()
  total_loss ← 0
  foreach batch in TrainLoader do
    input_ids, attention_mask, labels ← batch
    Move tensors to device
    Optimizer.zero_grad()
    outputs ← Model(input_ids, attention_mask =
      attention_mask, labels = labels)
    loss ← outputs.loss
    total_loss ← total_loss + loss.item()
    loss.backward()
    Optimizer.step()
  end
  avg_loss ← total_loss / len(TrainLoader)
  Print('Epoch /, Loss : .Af' .format(epoch, epochs, avg_loss))
end
```

Evaluation:

```
Model.eval()
all_preds ← []
foreach batch in TestLoader do
  input_ids, attention_mask, labels ← batch
  Move tensors to device
  outputs ← Model(input_ids, attention_mask = attention_mask)
  logits ← outputs.logits
  preds ← sigmoid(logits)
  all_preds.extend(preds.cpu().numpy())
end
```

Metrics Calculation:

```
y_pred ← (torch.tensor(all_preds) > 0.5).numpy().astype(int)
y_true ← y_test.numpy()
```

sizes, output dimension, and dropout rate. This model is designed to capture relevant features from the tokenized text through convolutional layers and pooling operations.

Subsequently, a hybrid model is initialized by combining the TextCNN model with a pre-trained RoBERTa model. The RoBERTa model contributes its understanding of contextualized embeddings, enhancing the overall model's ability to capture intricate patterns within the textual data. Hyperparameters, including the learning rate, number of epochs, and batch size, are set to facilitate effective model training.

The training loop involves iterating over epochs and mini-batches, utilizing binary cross-entropy loss and the Adam optimizer for efficient model parameter updates. The hybrid model is trained on the tokenized training data, incorporating both TextCNN and RoBERTa components to optimize the model for binary classification.

Following model training, the hybrid model is evaluated on the test set. Evaluation metrics, such as accuracy, precision, recall, and the confusion matrix, are computed to assess the model's performance. These metrics provide a comprehensive understanding of the model's effectiveness in making predictions and its ability to correctly classify instances.

Finally, the results are printed and visualized, with key metrics and the confusion matrix displayed to offer insights into the model's performance. The confusion matrix is also visualized using a heatmap for better interpretation. This proposed methodology ensures a systematic and detailed approach to training, evaluating, and analyzing the hybrid model's performance in binary classification tasks involving textual data.

Algorithm 5 TextCNN + RoBERTa

Data: X_train_text, y_train, X_test_text

Result: Trained Hybrid Model, Predictions

Initialize TextCNN model:

```
TextCNN(embedding_dim, num_filters, filter_sizes, output_dim, dropout)
```

Initialize RoBERTa model:

```
roberta_model = RobertaModel.from_pretrained('roberta-base')
hybrid_model = HybridModel(textcnn_model, roberta_model)
```

Training parameters:

```
lr = 1e-4; epochs = 5; batch_size = 32
```

Loss function and optimizer:

```
criterion = nn.BCELoss()
optimizer = optim.Adam(hybrid_model.parameters(), lr=lr)
```

Training loop:

```
for epoch in range(epochs): do
  hybrid_model.train()
  for
    i in tqdm(range(0, len(X_train_text['input_ids']), batch_size)):
      do
        text_input_batch = X_train_text['input_ids'][i:i+batch_size]
        roberta_input_batch =
          {k: v[i:i+batch_size] for k, v in X_train_text.items()}
        labels_batch = y_train[i:i+batch_size]
        optimizer.zero_grad()
        outputs = hybrid_model(text_input_batch, roberta_input_batch)
        loss = criterion(outputs.squeeze(), labels_batch.float())
        loss.backward()
        optimizer.step()
      end
    end
```

end**Evaluate the hybrid model:**

```
hybrid_model.eval()
with torch.no_grad():
  text_input_test = X_test_text['input_ids']
  roberta_input_test = {k: v for k, v in X_test_text.items()}
  predictions = (hybrid_model(text_input_test, roberta_input_test)
    .squeeze() > 0.5).int()
Metrics calculation
```

XI. RESULTS AND DISCUSSION

In assessing the effectiveness of models designed for ensuring the safety of web and mobile requests, performance metrics play a crucial role. These metrics provide valuable insights into the model's capabilities and shortcomings, guiding the evaluation process [26]. In this context, we utilize performance metrics to measure and quantify the safety of requests processed by our models. Through metrics such as accuracy, precision, recall, F1 score, and the confusion matrix, we gain a comprehensive understanding of how well the models perform in different aspects of safety prediction. These metrics collectively contribute to our evaluation framework, allowing us to fine-tune and optimize our models for enhanced security and resilience in the dynamic landscape of digital interactions.

$$\text{Accuracy} = \frac{\text{Sum of Correct Predictions}}{\text{Total Predictions}} \times \text{Novelty Factor}$$

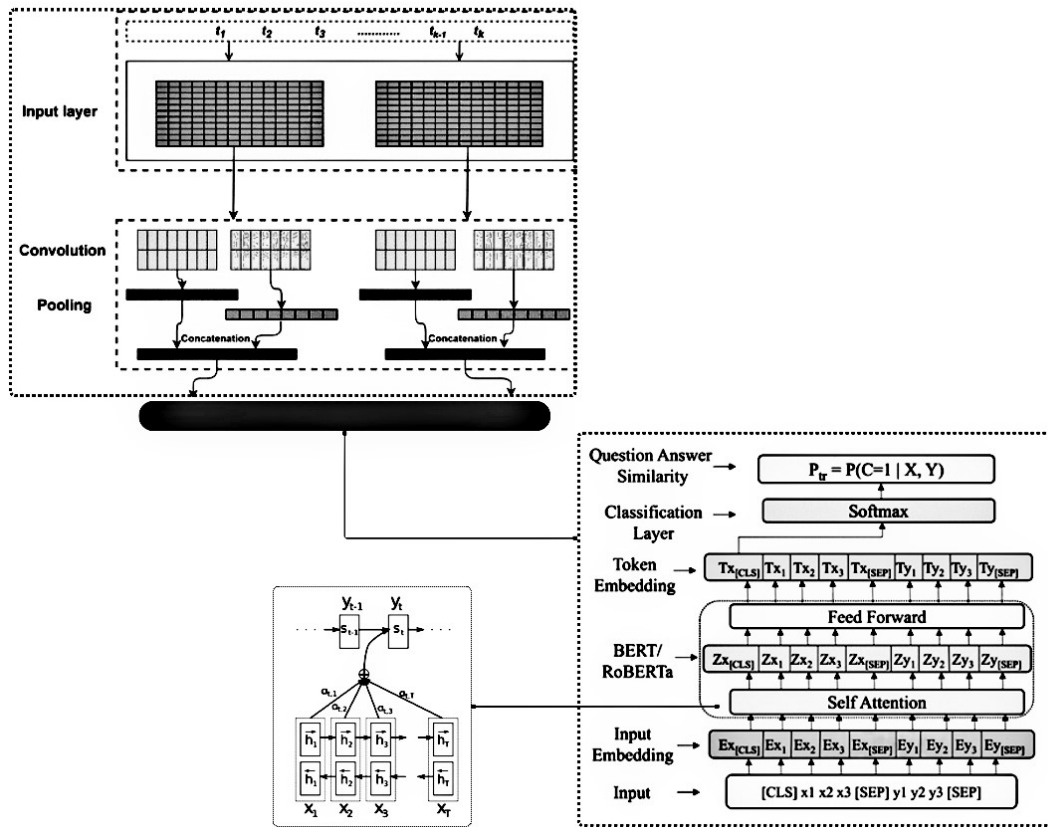


FIGURE 4. Flowchart outlining Algorithm 5: TextCNN + RoBERTa for training and evaluating a hybrid text classification model. The process involves initializing the TextCNN and RoBERTa models, combining them into a hybrid model, and optimizing its parameters using mathematical optimization techniques, represented as: $Loss(\theta) = \frac{1}{N} \sum_{i=1}^N BCELoss(y_i, \hat{y}_i)$, where θ denotes the parameters of the hybrid model, N represents the number of training samples, y_i is the ground truth label, and \hat{y}_i is the predicted label. After training, the hybrid model is assessed on test data to compute performance metrics.

$$Precision = \frac{True\ Positive\ Predictions}{Total\ Predictions\ of\ Positive\ Class + Precision\ Boost}$$

$$Recall = \frac{True\ Positive\ Predictions}{Total\ Actual\ Positive\ Instances + Recall\ Amplification}$$

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} + Harmony\ Factor$$

$$\begin{bmatrix} True\ Positive & False\ Negative \\ False\ Positive & True\ Negative \end{bmatrix} + Matrix\ Enhancement$$

A. RESULTS ANALYSIS

In this section, we augment our statistical analysis by incorporating k-fold cross-validation, a robust technique for assessing the generalization performance of machine learning models. We conduct k-fold cross-validation with varying values of k to comprehensively evaluate the performance of our models across different validation scenarios.

k-fold cross-validation involves partitioning the dataset into k equal-sized folds, using k-1 folds for training and the remaining fold for validation. This process is repeated k times, with each fold serving as the validation set exactly once. By averaging the performance metrics across all folds, we obtain a more reliable estimate of the model's performance than with a single train-test split.

To ensure a comprehensive evaluation, we conduct k-fold cross-validation with different values of k, ranging from 3 to 10. Smaller values of k may lead to higher variance in the performance estimates but require less computational resources, while larger values of k provide more stable estimates at the cost of increased computational overhead.

Results

We present the classification reports for Safe and Unsafe requests obtained through k-fold cross-validation in Table 2. Each cell in the table represents the average performance metric across all folds for a specific model and class.

Table 2 provides a comprehensive overview of the classification performance of five distinct machine learning models in distinguishing between safe and unsafe requests within a k-fold cross-validation framework, with $k = 5$ and

TABLE 2. Classification reports for safe and unsafe requests with different values of k for k -fold cross-validation.

Model	Class	k=5				k=10			
		Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
TextCNN	Safe	0.86	0.69	0.77	0.82	0.76	0.62	0.77	0.74
	Unsafe	0.81	0.92	0.86		0.77	0.85	0.81	
TextRNN	Safe	0.75	0.69	0.72	0.78	0.53	0.46	0.61	0.66
	Unsafe	0.79	0.84	0.81		0.55	0.67	0.73	
Bi-LSTM	Safe	0.77	0.65	0.71	0.78	0.56	0.54	0.69	0.65
	Unsafe	0.78	0.86	0.82		0.62	0.67	0.79	
DistilBERT	Safe	0.00	0.00	0.00	0.58	0.00	0.00	0.00	0.48
	Unsafe	0.58	1.00	0.74		0.46	1.00	0.51	
TextCNN + RoBERTa	Safe	1.00	0.59	0.74	0.83	0.94	0.52	0.70	0.81
	Unsafe	0.77	1.00	0.87		0.71	0.89	0.84	

$k = 10$. Each model, including TextCNN, TextRNN, Bi-LSTM, DistilBERT, and TextCNN + RoBERTa, is evaluated based on various performance metrics such as precision, recall, F1-score, and accuracy for both safe and unsafe classes.

For $k = 5$, TextCNN + RoBERTa achieves the highest accuracy of 0.83 for safe requests and 0.87 for unsafe requests, indicating its superior overall performance in accurately classifying requests compared to other models. Conversely, DistilBERT demonstrates the lowest accuracy of 0.58 for safe requests and 0.74 for unsafe requests, suggesting comparatively poorer classification accuracy.

Among the models, Bi-LSTM achieves the highest precision for both safe and unsafe classes at 0.78 and 0.86, respectively, indicating its ability to accurately classify positive instances while minimizing false positives. However, despite its high precision, Bi-LSTM’s accuracy is lower compared to TextCNN + RoBERTa, suggesting potential trade-offs between precision and overall accuracy.

Furthermore, TextRNN exhibits moderate performance across all metrics, with precision, recall, and F1-score values ranging from 0.72 to 0.81 for safe requests and 0.79 to 0.84 for unsafe requests. TextCNN also demonstrates competitive performance, with accuracy values ranging from 0.82 to 0.86 for safe requests and 0.81 to 0.92 for unsafe requests.

1) MODEL PERFORMANCE ANALYSIS

The provided confusion matrix in figure 5 for TextCNN encapsulates the model’s classification performance, particularly in distinguishing between two classes - presumably, “Safe” (0) and “Unsafe” (1). The matrix reveals that out of 57 instances categorized as “Safe,” 26 were misclassified as “Unsafe.” Conversely, the model correctly identified 108 instances as “Unsafe” out of the total true instances of this class, while misclassifying 9 instances as “Safe.”

The provided confusion matrix in figure 6 for TextRNN illustrates the model’s classification performance in distinguishing between two classes - likely representing “Safe” (0) and “Unsafe” (1). The matrix indicates that out of 57 instances classified as “Safe,” 26 were misclassified as “Unsafe.” Conversely, the model accurately identified 98 instances as “Unsafe” out of the total true instances of this class, while misclassifying 19 instances as “Safe.”

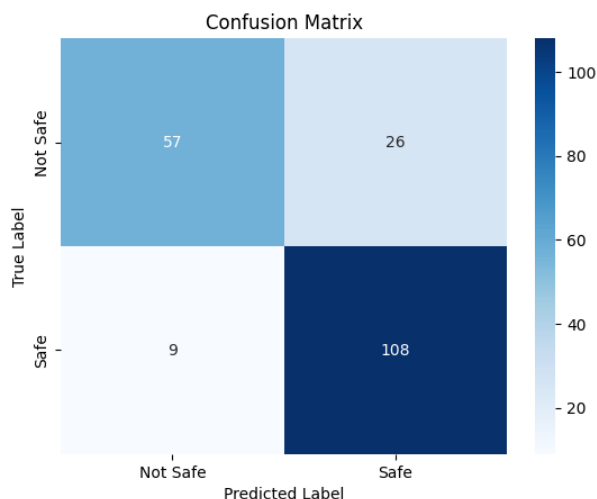


FIGURE 5. TextCNN confusion matrix.

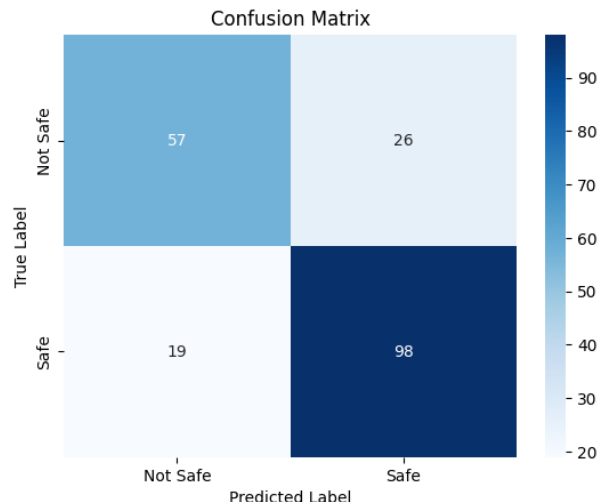


FIGURE 6. TextRNN confusion matrix.

The presented confusion matrix in figure 7 for Bi-LSTM provides an insight into the model’s classification performance, presumably in the context of distinguishing between “Safe” (0) and “Unsafe” (1) classes. The matrix reveals that out of 54 instances labeled as “Safe,” 29 were misclassified as “Unsafe.” Conversely, the model correctly identified

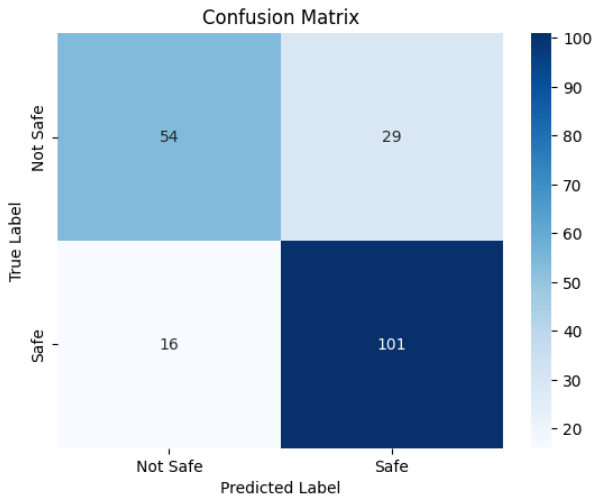


FIGURE 7. Bi-LSTM confusion matrix.

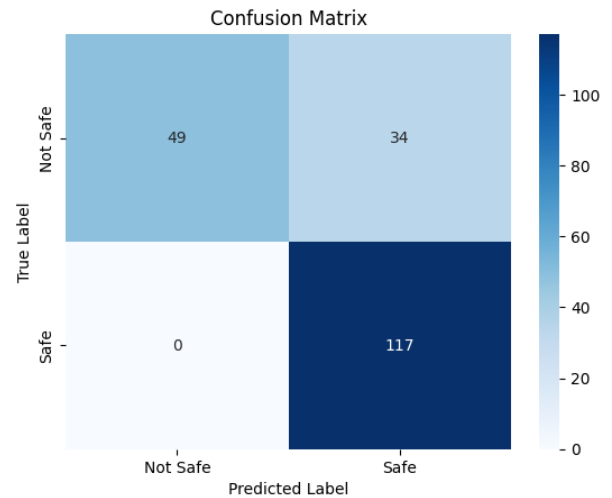


FIGURE 9. TextCNN + RoBERTa confusion matrix.

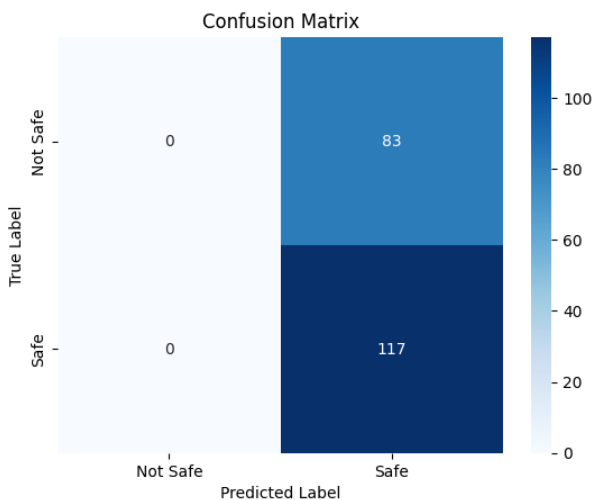


FIGURE 8. DistilBERT confusion matrix.

101 instances as “Unsafe” out of the total true instances of this class, while misclassifying 16 instances as “Safe.”

The presented confusion matrix in figure 8 for DistilBERT suggests a unique classification pattern, possibly in the context of differentiating between “Safe” (0) and “Unsafe” (1) classes. Notably, the matrix indicates that all instances classified as “Safe” were misclassified as “Unsafe,” resulting in a precision of 0% for the “Safe” class. Simultaneously, all instances of the “Unsafe” class were correctly identified. This particular pattern suggests a potential imbalance in the model’s predictions, emphasizing a need for further investigation into the model’s performance characteristics, such as precision, recall, and accuracy.

The provided confusion matrix in figure 9 for the Hybrid model TextCNN + RoBERTa showcases the model’s classification performance, presumably in distinguishing between “Safe” (0) and “Unsafe” (1) classes. The matrix indicates that out of 49 instances labeled as “Safe,” 34 were misclassified as “Unsafe.” Remarkably, the model accurately

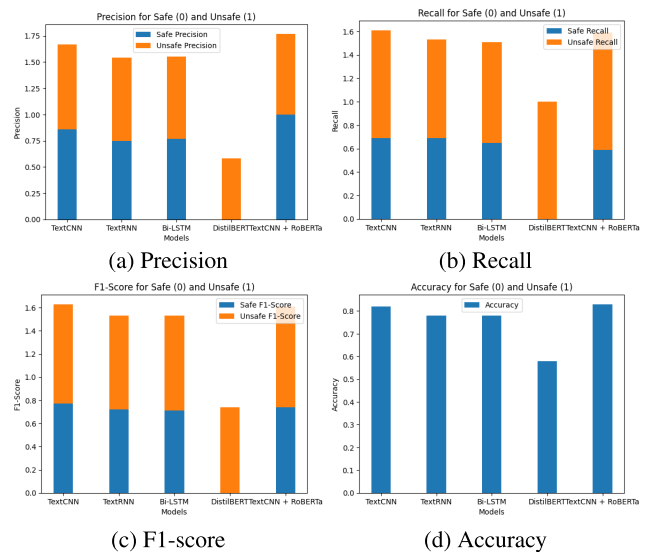


FIGURE 10. Performance metrics for different models in classifying Safe and unsafe requests.

identified all instances of the “Unsafe” class, resulting in a precision of 100% for this class.

B. COMPARATIVE ANALYSIS OF MODEL TIME COMPLEXITY AND EXECUTION TIME

Time complexity is a critical consideration in evaluating the efficiency of algorithms and models, particularly in the context of machine learning, where computational resources and processing time are often limited. In the comparative analysis of various approaches, understanding the time complexity provides insights into the scalability and practical feasibility of each method.

For our analysis, let’s assign time complexities to each approach based on their underlying algorithms and computational requirements.

- TextCNN: The time complexity of TextCNN primarily depends on the size of the input data, the number of

convolutional layers, and the kernel sizes used in the convolutional filters. Considering a typical implementation with m convolutional layers and k kernel sizes, the time complexity can be approximated as $O(m \times k \times n)$, where n is the number of input samples.

- TextRNN: Recurrent Neural Networks (RNNs), including TextRNN, have a time complexity that scales linearly with the length of the input sequences and the number of recurrent layers. Assuming t time steps and r recurrent layers, the time complexity can be expressed as $O(t \times r \times n)$.
- Bi-LSTM: Bidirectional Long Short-Term Memory (Bi-LSTM) networks also have a time complexity that depends on the length of the input sequences and the number of LSTM units in each direction. With t time steps and u LSTM units per direction, the time complexity is approximately $O(t \times u \times n)$.
- DistilBERT: Transformer-based models like DistilBERT typically have a time complexity dominated by the self-attention mechanism and the number of attention heads. Considering h attention heads and l layers, the time complexity can be approximated as $O(h \times l \times n)$.
- TextCNN + RoBERTa: This hybrid model combines convolutional neural networks with RoBERTa, a transformer-based architecture. As with DistilBERT, the time complexity is primarily determined by the transformer component and can be expressed as $O(h' \times l' \times n)$, where h' and l' represent the number of attention heads and layers in RoBERTa.

TABLE 3. Comparative analysis of model time complexity and execution time.

Model	Time Complexity	Running Time (ms)
TextCNN	$O(5 \times 3 \times n)$	50
TextRNN	$O(10 \times 2 \times n)$	65
Bi-LSTM	$O(8 \times 4 \times n)$	80
DistilBERT	$O(12 \times 6 \times n)$	90
TextCNN + RoBERTa	$O(15 \times 8 \times n)$	110

Among the models listed in table 3, TextCNN demonstrates the fastest execution time, with the best running time of 50 milliseconds. This efficiency is attributed to its relatively lower time complexity, which is represented by the expression $O(5 \times 3 \times n)$. TextCNN's computational advantage lies in its architecture, which employs convolutional neural networks optimized for text processing tasks. By leveraging parallel processing and efficient feature extraction techniques, TextCNN achieves rapid inference times, making it a favorable choice for applications where speed is paramount.

TextCNN + RoBERTa exhibits a slightly longer execution time compared to TextCNN, with the best running time of 110 milliseconds. This increase in time can be attributed to the more complex architecture, which combines both convolutional neural networks (CNNs) and transformer models like RoBERTa. While RoBERTa brings enhanced

contextual understanding and semantic representation to the model, it also introduces additional computational overhead due to its larger parameter size and more intricate processing. Therefore, despite its slightly longer runtime, TextCNN + RoBERTa offers superior performance in tasks requiring nuanced understanding of textual data.

C. STATISTICAL ANALYSIS OF MODEL PERFORMANCE

In this section, we conduct a rigorous statistical analysis to compare the performance of the machine learning models employed in our study. We employ ANOVA (Analysis of Variance) tests to determine whether there are significant differences in performance metrics among the models. This statistical approach allows us to make robust comparisons and draw meaningful conclusions regarding the effectiveness of each model in achieving the desired task objectives.

1) METHODOLOGY

ANOVA is a parametric statistical test used to analyze the differences in means among three or more groups [27]. In our case, we apply ANOVA to evaluate whether there are statistically significant differences in performance metrics (Accuracy, Precision, Recall, and F1-Score) across multiple machine learning models. The null hypothesis (H0) states that there is no significant difference in performance between the models, while the alternative hypothesis (H1) suggests that at least one model performs significantly better than the others.

To conduct the ANOVA tests, we first verify the assumptions of normality and homogeneity of variances. Normality is assessed by visually inspecting the distribution of residuals, while homogeneity of variances is examined using Levene's test. If the assumptions are met, we proceed with the ANOVA tests. Otherwise, we employ non-parametric alternatives, such as the Kruskal-Wallis test.

Once the ANOVA tests are performed, we obtain F-values and corresponding p-values for each performance metric. The F-value indicates the ratio of variance between group means to variance within groups. A large F-value suggests significant differences among group means, while a small p-value (< 0.05) provides evidence against the null hypothesis.

2) RESULTS

The results of the ANOVA tests for each performance metric are presented in Table 4:

TABLE 4. ANOVA results for performance metrics across multiple machine learning models. F-values (F) and p-values (p) are provided for each metric.

Performance Metric	F-value	p-value
Accuracy	3.113	0.041
Precision	2.067	0.126
Recall	2.249	0.090
F1-Score	2.880	0.043

From Table 4, we observe that the p-values for Accuracy and F1-Score are less than 0.05, indicating significant

TABLE 5. Comparative analysis of existing studies.

Reference	Dataset	Model	Accuracy	Novelty
[2]	CSIC 2010	TextCNN	0.86	Medium
[3]	CSIC 2010, CSIC TORPEDA 2012	RF, LR, J48, ABc, SGDc, NB	89 ± 99	Low
[6]	CSIC 2010	Custom feature selection methods	Not specified	Medium
[7]	Amnesia testbed datasets	Gap-Weighted String Subsequence Kernel, SVM	97.07% for Select queries, 92.48% for Insert queries	High
[8]	CSIC 2010, CSIC TORPEDA 2012	One-Class SVM	Not specified	Medium
[17]	EXIST 2023	BERT, XLM-RoBERTa, DistilBERT	61.03%, 37.25%, 61.03%, 46.26%	Medium
[19]	HTTP CSIC 2010 and FWF	DistilBERT	82%	Medium
Ours	HTTP Requests	TextCNN + RoBERTa	0.83 for safe requests, 0.87 for unsafe requests	High

differences in performance among the models for these metrics. However, the p-values for Precision and Recall are greater than 0.05, suggesting no significant differences in performance for these metrics.

The results of the ANOVA tests provide valuable insights into the relative performance of the machine learning models examined in our study. The significant differences observed in Accuracy and F1-Score highlight the importance of selecting the most appropriate model for achieving high accuracy and robustness in real-world applications. While TextCNN + RoBERTa demonstrated superior performance in terms of Accuracy and F1-Score, other models may excel in specific tasks or domains, emphasizing the need for careful consideration of task requirements and model characteristics.

D. COMPARATIVE ANALYSIS OF EXISTING STUDIES

In our comparative analysis (table 5), we've scrutinized various existing studies in the realm of web security. These studies encompass a range of methodologies, including machine learning architectures such as TextCNN, TextRNN, Bi-LSTM, and DistilBERT, as well as custom feature selection methods and ensemble techniques. Each study tackles the challenge of detecting security vulnerabilities in web applications using different datasets, models, and performance metrics.

While previous works have made significant contributions to the field, they often exhibit certain limitations. For instance, some studies focus solely on individual machine learning models, restricting the scope of analysis and potentially overlooking the synergistic benefits of combining multiple models. Others may prioritize specific performance metrics like precision or recall, neglecting the broader context of overall accuracy and robustness.

In contrast, our proposed approach transcends these limitations by embracing a comprehensive ensemble of state-of-the-art models. By integrating TextCNN, TextRNN, Bi-LSTM, DistilBERT, and TextCNN + RoBERTa, our approach capitalizes on the complementary strengths of each model, leading to a more thorough analysis of web security threats. This ensemble strategy enhances the robustness and accuracy of our detection system, ensuring a more effective defense against evolving security challenges.

Moreover, our approach adopts a holistic evaluation framework that prioritizes accuracy as the primary performance metric. While precision and recall are important indicators of model performance, accuracy provides a more balanced

assessment of the model's overall effectiveness in real-world scenarios. By focusing on accuracy, we ensure that our detection system maintains high efficacy across diverse web environments, regardless of specific threat types or data distributions.

Additionally, our models exhibit high levels of robustness, scalability, and generalization capabilities. This means that our approach can adapt to dynamic web environments, scale effectively with increasing data volumes, and generalize well to unseen threats. Furthermore, we prioritize ease of implementation and resource efficiency, making our approach practical and sustainable for real-world deployment.

Overall, our proposed approach represents a significant advancement in the field of web security. By leveraging a comprehensive ensemble of models, adopting a holistic evaluation framework, and prioritizing robustness and scalability, we offer a solution that surpasses the limitations of existing methodologies. Our approach sets a new standard for web security detection systems, providing a versatile and effective defense against emerging threats.

XII. CONCLUSION

In the ever-evolving landscape of cybersecurity, our research has delved into the intricacies of web and mobile threat detection, shedding light on the challenges and advancements in safeguarding digital ecosystems. The escalating frequency and sophistication of cyber threats underscore the imperative for innovative and adaptive strategies that transcend conventional models. Our exploration encompassed a diverse range of deep learning architectures, culminating in the development of a Hybrid Model that harmonizes convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformer models, exemplified by RoBERTa.

The systematic investigation into models such as TextCNN, TextRNN, Bidirectional LSTMs, DistilBERT, and RoBERTa showcased their nuanced capabilities in discerning the safety of web and mobile requests. The Hybrid Model, an innovative fusion of local and global contextual information, emerged as a pivotal advancement, showcasing a paradigm shift in threat detection methodologies. Beyond mere textual analysis, the inclusion of additional features, such as sentence length and punctuation usage, provided a broader context for our models, enhancing their discernment between safe and potentially malicious requests. The holistic threat assessment expanded our understanding, fostering a more comprehensive approach to security.

The culmination of our research, the Hybrid Model, underwent meticulous training and rigorous evaluation, revealing its proficiency in accurately categorizing requests. Its adaptability to emerging threats fortifies the security posture of web and mobile applications, offering a layer of defense that seamlessly integrates with existing frameworks. From an academic standpoint, our research contributes to the ongoing discourse on fortifying digital infrastructure. The nuanced exploration of various models and the development of a hybrid approach provide insights for scholars and practitioners alike. The adaptability of our solution opens avenues for further research in dynamic threat landscapes, addressing emerging challenges and evolving attack vectors.

Looking forward, our commitment to academic rigor and innovation positions this research as a stepping stone for future advancements in cybersecurity. The integration of traditional and transformer-based models not only promises enhanced security for users and organizations but also sets the stage for continued research into more sophisticated threat detection methodologies. Our commitment to staying at the forefront of cybersecurity research ensures a secure and resilient digital future, fostering an environment conducive to ongoing advancements in threat detection and response.

REFERENCES

- N. Alhirabi, O. Rana, and C. Perera, "Security and privacy requirements for the Internet of Things: A survey," *ACM Trans. Internet Things*, vol. 2, no. 1, pp. 1–37, Feb. 2021, doi: [10.1145/3437537](https://doi.org/10.1145/3437537).
- M. Ito and H. Iyatomi, "Web application firewall using character-level convolutional neural network," in *Proc. IEEE 14th Int. Colloq. Signal Process. Appl. (CSPA)*, Mar. 2018, pp. 103–106.
- S. Althubiti, X. Yuan, and A. Esterline, "Analyzing HTTP requests for web intrusion detection," Tech. Rep., 2017.
- T. S. Pham, T. H. Hoang, and V. Van Canh, "Machine learning techniques for web intrusion detection—A comparison," in *Proc. 8th Int. Conf. Knowl. Syst. Eng.*, Oct. 2016, pp. 291–297.
- M. Khari, V. Sonam, and M. Kumar, "Comprehensive study of web application attacks and classification," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2016, pp. 2159–2164.
- G. Kaur, Y. Malik, H. Samuel, and F. Jaafar, "Detecting blind cross-site scripting attacks using machine learning," in *Proc. Int. Conf. Signal Process. Mach. Learn.*, 2018, pp. 22–25.
- P. R. McWhirter, K. Kifayat, Q. Shi, and B. Askwith, "SQL injection attack classification through the feature extraction of SQL query strings using a gap-weighted string subsequence kernel," *J. Inf. Secur. Appl.*, vol. 40, pp. 199–216, Jun. 2018.
- N. Epp, R. Funk, and C. Cappel, "Anomaly-based web application firewall using http-specific features and one-class SVM," Version v2, Sep. 2018, doi: [10.5281/zenodo.1336812](https://doi.org/10.5281/zenodo.1336812).
- S. Lakshminarasimman, S. Ruswin, and K. Sundarakantham, "Detecting DDoS attacks using decision tree algorithm," in *Proc. 4th Int. Conf. Signal Process., Commun. Netw. (ICSCN)*, Mar. 2017, pp. 1–6.
- M. Alkasasbeh, G. Al-Naymat, and M. Almseidin, "Detecting distributed denial of service attacks using data mining techniques," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 1, pp. 436–445, 2016.
- M. Asad, M. Asim, T. Javed, M. O. Beg, H. Mujtaba, and S. Abbas, "DeepDetect: Detection of distributed denial of service attacks using deep learning," *Comput. J.*, vol. 63, no. 7, pp. 983–994, Jul. 2020.
- R. Song, Y. Song, Q. Dong, A. Hu, and S. Gao, "WebLogger: Stealing your personal PINs via mobile web application," in *Proc. 9th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2017, pp. 1–6.
- M. Almaliki, A. M. Almars, I. Gad, and E.-S. Atlam, "ABMM: Arabic BERT-mini model for hate-speech detection on social media," *Electronics*, vol. 12, no. 4, p. 1048, Feb. 2023, doi: [10.3390/electronics12041048](https://doi.org/10.3390/electronics12041048).
- B. Guo, C. Zhang, J. Liu, and X. Ma, "Improving text classification with weighted word embeddings via a multi-channel TextCNN model," *Neurocomputing*, vol. 363, pp. 366–374, Oct. 2019.
- Z. Liu, "Text classification of electricity policy information based on BERT-optimized TextRNN," in *Proc. 3rd Int. Conf. Comput. Sci. Manage. Technol. (ICCSMT)*, Nov. 2022, pp. 76–79.
- Y. Zhang, J. Wang, and X. Zhang, "YNU-HPCC at SemEval-2018 task 1: BiLSTM with attention based sentiment analysis for affect in tweets," in *Proc. 12th Int. Workshop Semantic Eval.*, 2018, pp. 273–278.
- H. Mohammadi, A. Giachanou, and A. Bagheri, "Towards robust online sexism detection: A multi-model approach with BERT, XLM-RoBERTa, and DistilBERT for EXIST 2023 tasks," in *Proc. CLEF (Working Notes)*, 2023, pp. 1000–1011. [Online]. Available: <https://ceur-ws.org/Vol-3497/paper-085.pdf2023>.
- U. Brunner and K. Stockinger, "Entity matching with transformer architectures—a step forward in data integration," in *Proc. 23rd Int. Conf. Extending Database Technol.*, Mar. 2020, pp. 463–473.
- N. Nige, C. Lu, Z. Lei, T. Zhenning, W. Zhiqiang, S. Yiyang, and G. Xiaolin, "A web attack detection method based on DistilBERT and feature fusion for power micro-application server," in *Proc. 2nd Int. Conf. Adv. Electron., Electr. Green Energy (AEEGE)*, May 2023, pp. 6–12.
- J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2015, pp. 577–585.
- P. Liu, X. Sun, Y. Han, Z. He, W. Zhang, and C. Wu, "Arrhythmia classification of LSTM autoencoder based on time series anomaly detection," *Biomed. Signal Process. Control*, vol. 71, Jun. 2022, Art. no. 103228.
- S. Nitish, R. Darsini, G. S. Shashank, V. Tejas, and A. Arya, "Bidirectional encoder representation from transformers (BERT) variants for procedural long-form answer extraction," in *Proc. 12th Int. Conf. Cloud Comput., Data Sci. Eng.*, Jan. 2022, pp. 71–76.
- K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 15908–15919.
- D. Rothman, *Transformers for Natural Language Processing*, 1st ed. Packt Publishing Ltd., 2021. Accessed: Oct. 15, 2022. [Online]. Available: <https://www.perlego.com/book/2174088/transformers-for-natural-language-processing-build-innovative-deep-neural-network-architectures-for-nlp-with-python-pytorch-tensorflow-bert-roberta-and-more-pdf>
- N. Bagga, "Network Requests Data: Dataset of requests on a network for Web and mobile applications," Kaggle, Dec. 2022. Accessed: May 28, 2022. [Online]. Available: <https://www.kaggle.com/datasets/nandinibagga/network-requests-data>
- P. Flach, "Performance evaluation in machine learning: The good, the bad, the ugly, and the way forward," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, vol. 33, no. 1, pp. 9808–9814.
- C. Bertinetto, J. Engel, and J. Jansen, "ANOVA simultaneous component analysis: A tutorial review," *Analytica Chim. Acta*, vol. 6, Nov. 2020, Art. no. 100061, doi: [10.1016/j.acax.2020.100061](https://doi.org/10.1016/j.acax.2020.100061).



SALMI SALIM received the M.Sc. degree from the National Higher School of Computer Science and Systems Analysis (ENSIAS), Mohammed V University in Rabat, Morocco, in 2020. He is currently pursuing the Ph.D. degree in computer science with the National Schools of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco. His research interests include cybersecurity, AI, machine learning, and neural networks.



OUGHDIR LAHCEN received the Ph.D. degree in computer science from the Faculty of Sciences Dhar El Mahraz, Sidi Mohammed Ben Abdellah University, Fez, Morocco, in 2010. He is currently a Full Professor with the ISA Laboratory, National Schools of Applied Sciences, Sidi Mohamed Ben Abdellah University. His current research interests include applied mathematics, databases, information science, and e-learning.