

Bayesian-Optimized Word2Vec-BiLSTM for Malicious Prompt Detection in LLMs

Hilman Singgih Wicaksana¹⁾, Gregorius Airlangga^{2*)}

¹⁾ Informatics Program, Universitas Karya Husada, Semarang, Indonesia

²⁾ Information Systems Program, Universitas Katolik Indonesia Atma Jaya, Jakarta, Indonesia

¹⁾ singgih.hilman@gmail.com, ^{2*)} gregorius.airlangga@atmajaya.ac.id

Submitted : Aug 4, 2026 | **Accepted :** Aug 25, 2026 | **Published :** Oct 6, 2026

Abstract: Large Language Models (LLMs) are increasingly utilized across a wide range of artificial intelligence applications, but their growing adoption also introduces security concerns related to malicious prompts that may manipulate model behavior. This study investigates an optimized Word2Vec-BiLSTM approach for detecting binary malicious prompts using the Malicious Prompt Detection Dataset (MPDD). Four deep learning configurations, namely FastText-BiLSTM, FastText-BiGRU, Word2Vec-BiLSTM, and Word2Vec-BiGRU, were evaluated alongside a TF-IDF combined with Logistic Regression (LogReg) baseline under consistent experimental conditions and across three random seeds. Bayesian Optimization was applied to identify effective hyperparameter configurations and quantify performance differences between manually configured and optimized models. The models were evaluated using precision, recall, F1-score, accuracy, false-positive rate (FPR), false-negative rate (FNR), precision-recall area under the curve (PR-AUC), and computational cost. The optimized Word2Vec-BiLSTM with seed 128 achieved the highest overall performance across the primary classification metrics: 97.03% precision, 96.86% recall, 96.89% F1-score, and 96.89% accuracy, along with an FPR of 0.0070, an FNR of 0.0557, and a PR-AUC of 99.39%. The optimized deep learning models generally outperformed the TF-IDF + LogReg baseline and exhibited relatively similar computational characteristics across the evaluated deep learning configurations. These findings demonstrate that Bayesian hyperparameter optimization can improve the performance of the Word2Vec-BiLSTM model for binary malicious prompt detection within the MPDD-based experimental setting.

Keywords: Bayesian Optimization, BiLSTM, Deep Learning, Malicious Prompt Detection, Word2Vec

INTRODUCTION

Large Language Models (LLMs) have advanced rapidly and are increasingly integrated into a wide range of artificial intelligence applications. These models demonstrate strong capabilities in understanding natural language, generating contextual responses, and supporting automated decision-making processes (Mirshekari et al., 2025). LLMs have been widely adopted in chatbots, virtual assistants, content generation systems, document analysis, and intelligent automation services (Raza et al., 2025). Their widespread adoption provides substantial benefits in improving productivity and supporting digital transformation (Jelodar, 2025). However, the increasing dependence on LLM-based applications also introduces security challenges that must be addressed to ensure the reliability and safety of these systems.

One of the important security challenges in LLM-based systems is the presence of malicious prompts that may manipulate model behavior or cause an LLM to deviate from its intended function. Such prompts may include instruction-manipulation attempts, prompt injection-related inputs, and other potentially adversarial or harmful prompt patterns. In particular, prompt injection attacks involve intentionally crafted instructions that attempt to influence an LLM's behavior, bypass intended restrictions, or induce unintended responses (Alshammari & Alsaleh, 2026; Derner et al., 2024). These vulnerabilities may reduce the reliability of LLM-based applications and potentially compromise the confidentiality and integrity of the information they process (Feretzakis & Verykios, 2024). Therefore, detecting potentially malicious prompts before they are processed by an LLM is an important component of securing LLM-based applications.

*Gregorius Airlangga



This is anCreative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

In this study, malicious prompt detection is formulated as a binary classification task using the Malicious Prompt Detection Dataset (MPDD) obtained from Kaggle (Jebbar, 2025). The dataset contains prompt instances labelled as either malicious or non-malicious, with the “isMalicious” attribute indicating the binary target label. Because MPDD aggregates prompts from multiple source datasets and does not provide a detailed, standardized taxonomy of malicious prompt categories, this study does not attempt to distinguish specific attack types such as prompt injection, jailbreaks, or other harmful prompt categories. Instead, the objective is to distinguish prompts labelled as malicious from those labelled as non-malicious according to the original MPDD annotations. Accordingly, the findings are interpreted as binary detection of malicious prompts within the MPDD-based experimental setting rather than as a comprehensive evaluation of specific attack categories.

Given this binary detection setting, malicious prompt detection can serve as an additional security layer before an LLM processes user input. Such a mechanism can help identify suspicious prompts at the input level and reduce the likelihood that potentially malicious instructions reach the underlying LLM. Previous studies have demonstrated the feasibility of automated approaches for identifying malicious and adversarial prompt inputs using machine learning and deep learning techniques (Kurniawan & Chandra, 2025; Lan et al., 2025). By identifying suspicious linguistic patterns before LLM processing, a detection model may help limit the influence of potentially malicious inputs on downstream model behavior. Consequently, developing accurate and computationally practical malicious prompt detection models remains important for improving the security of LLM-based systems.

Achieving accurate malicious prompt detection requires models that can capture meaningful patterns and contextual relationships within prompt text. From a machine learning perspective, this task can be formulated as a text classification problem in which input prompts are classified as benign or malicious. Deep learning approaches are particularly suitable for this task because they can learn complex patterns and contextual relationships from sequential textual data (Ahmed et al., 2023). Among sequential deep learning architectures, Bidirectional Long Short-Term Memory (BiLSTM) processes textual sequences in both the forward and backward directions, enabling the model to capture contextual dependencies from both directions. This capability can be beneficial for identifying relationships among words that may indicate malicious or adversarial intent within a prompt (Bongirwar & Mokhadde, 2024; Pavlatos et al., 2023). In addition to BiLSTM, Bidirectional Gated Recurrent Unit (BiGRU) provides an alternative bidirectional recurrent architecture with a comparatively simpler gating mechanism, making the evaluation of both architectures relevant for examining their respective roles in malicious prompt detection.

The ability of these recurrent architectures to identify meaningful prompt patterns also depends on how textual inputs are represented before being processed by the model. Word embedding techniques transform textual tokens into numerical vectors that capture relationships among words (Patil et al., 2023). Word2Vec learns distributed representations from word co-occurrence patterns, representing semantically related words in a continuous vector space. FastText extends conventional word representations by incorporating sub-word information, which can be useful for representing rare words and textual variations. These characteristics make Word2Vec and FastText relevant alternatives for representing malicious prompts prior to processing by bidirectional recurrent neural networks.

Beyond the choice of textual representation and recurrent architecture, the performance of deep learning models is also influenced by the selection of appropriate hyperparameters, including the learning rate, number of recurrent units, and dropout rate. Manual selection through trial and error may not consistently identify an effective configuration for a given classification task. Bayesian Optimization provides a systematic approach for exploring a predefined hyperparameter space by using information from previous evaluations to identify promising configurations (Wicaksana et al., 2024). In this study, Bayesian Optimization is therefore used not only to identify an effective configuration for the Word2Vec-BiLSTM model but also to quantify its effect by comparing optimized configurations with manually configured counterparts under identical experimental conditions. This comparison provides a basis for determining whether systematic hyperparameter optimization improves the performance of malicious prompt detection.

The combination of model architecture, textual representation, and hyperparameter configuration has therefore become an important consideration in developing malicious prompt detection methods. Previous studies have investigated this problem using conventional machine learning, deep learning, and different text representation techniques, demonstrating that automated classification can distinguish malicious or adversarial prompts from benign ones. However, reported performance may vary across datasets, text representations, model architectures, preprocessing procedures, and evaluation protocols. Such methodological differences can make it difficult to determine whether observed performance improvements are primarily attributable to the model architecture, the textual representation, or the experimental configuration. Therefore, examining previous approaches under a consistent experimental perspective is important for understanding the factors that contribute to malicious prompt detection performance and for providing an appropriate basis for evaluating the contribution of hyperparameter optimization.

Within this broader literature, the Malicious Prompt Detection Dataset (MPDD) has recently been examined through several complementary research directions. Vyas et al. (2026) developed a malicious prompt classification and sanitization framework using MPDD, with particular emphasis on token-level interpretability and prompt sanitization. Kushnerov et al. (2026) conducted a comparative evaluation of multiple deep learning architectures on MPDD, including word-level and character-level models, and further examined their robustness to spacing and homoglyph perturbations. Their results showed that the word-level BiLSTM achieved strong classification performance on clean MPDD samples, whereas character-level architectures exhibited greater robustness to certain symbolic perturbations. Together, these studies demonstrate that MPDD has been used to examine classification performance, interpretability, architectural differences, and robustness in malicious prompt detection.

The findings from these MPDD-based studies provide a basis for examining other factors that may influence detection performance, particularly the role of hyperparameter selection. Their results demonstrate the effectiveness of word-level BiLSTM architectures on MPDD, but they do not directly establish how systematic Bayesian hyperparameter optimization affects a Word2Vec-BiLSTM configuration under a controlled comparison with manually selected settings. This distinction is important because performance improvements obtained through hyperparameter optimization should be evaluated against an appropriate non-optimized configuration rather than solely against the optimized model's performance. The present study therefore focuses on quantifying the contribution of Bayesian Optimization by comparing optimized and manually configured Word2Vec-BiLSTM models under consistent experimental conditions. These considerations motivate a direct evaluation of whether Bayesian hyperparameter optimization yields consistent benefits for Word2Vec-BiLSTM across different random seeds, while accounting for classification performance, error characteristics, and computational requirements.

Building on these findings, this study investigates an optimized Word2Vec-BiLSTM model for binary detection of malicious prompts on MPDD. Word2Vec is employed to generate distributed word representations, while BiLSTM is used to capture bidirectional contextual dependencies within prompt sequences. Bayesian Optimization is used to identify an effective hyperparameter configuration, and its contribution is assessed by comparing the optimized configuration with a manually configured counterpart under consistent experimental conditions. To provide a comparative evaluation, Word2Vec-BiGRU, FastText-BiLSTM, and FastText-BiGRU are included as alternative deep learning models, while TF-IDF combined with Logistic Regression (LogReg) is used as a conventional baseline. All models are evaluated across three random seeds using primary classification metrics, detailed malicious prompt error measures, and computational characteristics to provide a consistent assessment of the evaluated approaches.

The study provides empirical evidence on whether systematic Bayesian hyperparameter optimization yields measurable benefits for Word2Vec-BiLSTM-based binary malicious prompt detection on MPDD. The effect of optimization is quantified by comparing manually configured and optimized settings across three random seeds, using precision, recall, F1-score, and accuracy as the primary classification measures, along with malicious prompt recall, false-positive rate, false-negative rate, PR-AUC, and ROC-AUC for a more detailed assessment of detection behavior. The study also compares Word2Vec-BiLSTM with Word2Vec-BiGRU, FastText-BiLSTM, and FastText-BiGRU to examine the relative performance of different combinations of word representations and bidirectional recurrent architectures under consistent experimental conditions. TF-IDF combined with Logistic Regression (LogReg) is included as a conventional lexical baseline to provide a reference for the deep learning approaches. In addition, the computational characteristics of the evaluated models are analyzed in terms of training time, inference time, throughput, peak GPU memory usage, parameter count, and estimated model size. These evaluations provide empirical evidence regarding the effectiveness, stability, error characteristics, and computational requirements of the evaluated approaches within the MPDD-based experimental setting.

LITERATURE REVIEW

Vyas et al. (2026) investigated malicious prompt detection using the Malicious Prompt Detection Dataset (MPDD) and developed a framework that combined classification with prompt sanitization. Their study evaluated TF-IDF-based and sentence-embedding-based classifiers, with XGBoost combined with sentence embeddings achieving approximately 95% accuracy. Beyond classification performance, the study incorporated Markov Chain classification and Leave-One-Out Deletion (LODO) analysis to identify tokens associated with malicious behavior. These analyses provided token-level interpretability to examine the contribution of individual tokens to the classification of malicious prompts. The study further extended this analysis into an intent-preserving prompt sanitization framework, demonstrating an approach that combines detection, interpretability, and subsequent modification of potentially malicious prompts.

Kushnerov et al. (2026) conducted a comparative benchmark of eight deep learning architectures on MPDD, including word-level Dense, CNN, BiLSTM, BiGRU, Transformer, and ResNet models, as well as character-level CNN and BiLSTM architectures. Their results showed that the word-level BiLSTM achieved the highest performance on clean samples, with an accuracy and F1-score of 96.81% and an AUC of 0.9918, respectively. The study further examined robustness to spacing and homoglyph perturbations to determine how different

architectures respond to modified prompt inputs. The results indicated that the character-level BiLSTM provided substantially greater robustness to such symbolic perturbations than the word-level models. These findings demonstrate that both the choice of architecture and the representation level can influence the performance and robustness of malicious prompt detection on MPDD.

The findings of these studies provide evidence that both model architecture and representation level can influence malicious prompt detection performance on MPDD, while also showing the importance of interpretability and robustness in evaluating detection systems. However, the MPDD-based studies considered here do not explicitly examine the contribution of systematic Bayesian hyperparameter optimization to a Word2Vec-BiLSTM configuration by directly comparing it with a manually configured counterpart. This distinction is important because the effectiveness of hyperparameter optimization should be assessed by determining whether the optimized configuration yields a measurable improvement over an appropriate non-optimized configuration under consistent experimental conditions. The present study therefore investigates Bayesian Optimization for Word2Vec-BiLSTM and evaluates whether the resulting configuration provides improved classification performance across multiple random seeds. The analysis also compares alternative combinations of word representations and bidirectional recurrent architectures, including Word2Vec-BiGRU, FastText-BiLSTM, and FastText-BiGRU, with TF-IDF combined with Logistic Regression (LogReg) serving as a conventional baseline. In addition to classification performance, computational characteristics are considered to provide a broader assessment of the evaluated approaches within the MPDD-based experimental setting.

METHOD

This study presents the research methodology for developing a model for detecting malicious prompts. The research process encompasses several critical stages, including data collection, data preprocessing, word embedding, model building, and model evaluation. In the data collection phase, the dataset is systematically organized and prepared according to the relevant target classes for this study. This is followed by the preprocessing stage, which involves cleaning and preparing the text data before it is represented with word embeddings and processed by the proposed model. Lastly, the model's performance is evaluated using suitable metrics, and the overall research workflow is illustrated in Fig. 1.



Fig. 1. Research Method

Data Collection

During the data collection phase, this study employed the Malicious Prompt Detection Dataset (MPDD) obtained from Kaggle. The dataset contains 39,234 prompt instances with a balanced distribution of 19,617 malicious and 19,617 benign prompts. According to the dataset description, MPDD was constructed by aggregating and cleaning several existing datasets, including Prompt Injection in the Wild - Prediction Guard, Prompt Injection Malignant, LLM 7 Prompt Training Dataset, and the Quora Question Pairs Dataset (Jebbar, 2025). Because the dataset combines prompts from different sources and does not provide a detailed taxonomy of malicious prompt types, this study treats the task as binary classification rather than as classification into specific attack categories. The original binary labels are used to distinguish malicious and non-malicious prompts, with the benign class referred to as “Not Malicious” throughout this study. The class-wise distribution of the dataset is presented in Fig. 2.

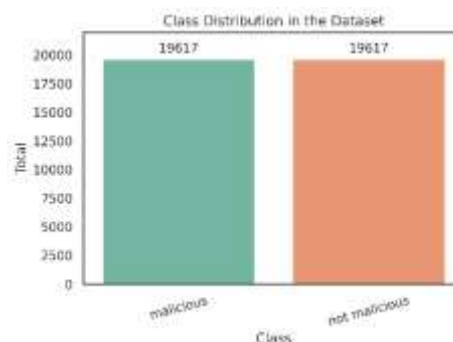


Fig. 2. Class Distribution in the Dataset

Before dataset splitting, a data duplication and similarity screening procedure was performed on the MPDD dataset. First, exact duplicates were identified by comparing the original prompt text, and duplicate instances were removed to ensure that identical prompts were not retained in the dataset. For similarity analysis, the remaining prompts were normalized to lowercase, with consecutive whitespace characters collapsed and leading or trailing spaces removed. TF-IDF representations using word unigrams and bigrams were then generated, and cosine similarity was evaluated using a nearest-neighbor approach. For each prompt, its nearest neighbors were examined, and prompt pairs with a cosine similarity of at least 0.90 were identified as highly similar and used to establish similarity relationships between prompt instances. This procedure was performed before dataset splitting to reduce the risk of information leakage from identical or highly similar prompts appearing across different data subsets.

To prevent highly similar prompts from being distributed across different data subsets, the identified similarity relationships were converted into similarity groups using a union-find procedure. Prompts connected through the identified similarity pairs were assigned to the same group, and dataset splitting was subsequently performed at the group level using GroupShuffleSplit with a fixed random state of 42. The similarity groups were first partitioned into 80% training data and 20% temporary data, after which the temporary data were divided equally into validation and test sets, resulting in 31,387 training instances, 3,921 validation instances, and 3,926 test instances. The resulting class distributions remained highly balanced, with the proportions of labels 0 and 1 in the training set being 50.05% and 49.95%, in the validation set 50.19% and 49.81%, and in the test set 49.39% and 50.61%, respectively. These results indicate that the group-based splitting procedure maintained a nearly balanced class distribution across the three subsets despite not explicitly applying label stratification. The similarity-group overlap checks confirmed that there was no overlap among the three subsets: $\text{Train} \cap \text{Validation} = 0$, $\text{Train} \cap \text{Test} = 0$, and $\text{Validation} \cap \text{Test} = 0$, indicating that no identified similarity group was shared across the training, validation, and test sets.

Data Preprocessing

The data preprocessing stage consisted of several steps, including text normalization, cleaning, case folding, tokenization, and finalization. Text normalization was first applied to remove excessive whitespace while preserving the original text. The cleaning stage removed nonessential characters while retaining punctuation that may contribute to the semantic or structural interpretation of prompts, such as quotation marks and question marks. Case folding was subsequently applied to convert all text to lowercase (Wicaksono et al., 2023). Tokenization was then performed to divide each prompt into individual words or tokens (Lubis et al., 2023), and finalization completed preprocessing by recombining the resulting tokens into a textual sequence. Examples of the preprocessing results are shown in Table 1.

Table 1. Example of Data Preprocessing Results

Phases	Before Preprocessing	After Preprocessing
Normalization	How did the "Coronation Street" storyline handle transgender issues?	How did the "Coronation Street" storyline handle transgender issues?
Cleaning	How did the "Coronation Street" storyline handle transgender issues?	How did the "Coronation Street" storyline handle transgender issues?
Case Folding	How did the "Coronation Street" storyline handle transgender issues?	how did the "coronation street" storyline handle transgender issues?
Tokenization	how did the "coronation street" storyline handle transgender issues?	['how', 'did', 'the', '', 'coronation', 'street', '', 'storyline', 'handle', 'transgender', 'issues', '?']
Finalization	['how', 'did', 'the', '', 'coronation', 'street', '', 'storyline', 'handle', 'transgender', 'issues', '?']	how did the "coronation street" storyline handle transgender issues?

Once text preprocessing was complete, the categorical class labels were encoded as numerical values to enable processing by the machine learning models. In this study, label encoding was performed using the LabelEncoder class from the Scikit-learn library. This technique assigns a unique integer value to each class label while preserving the binary classification setting adopted in this study (Bolikulov et al., 2024). Specifically, the class “Malicious” was encoded as 0, while “Not Malicious” was encoded as 1. The resulting label encoding scheme is presented in Table 2.

Table 2. Representation of Label Encoding

Classes	Representations
Malicious	0
Not Malicious	1

Word Embedding

In the word embedding stage, this study utilized two techniques for word representation: Word2Vec and FastText. Word2Vec was chosen for its ability to effectively capture semantic relationships between words in large text corpora by leveraging each word's contextual information. On the other hand, FastText is particularly adept at representing short texts and smaller datasets because it incorporates sub-word information when creating word representations (Shaday et al., 2024). After the dataset was partitioned into training, validation, and testing subsets, both Word2Vec and FastText models were trained exclusively using the training set. The learned embedding representations were subsequently applied to the validation and testing sets without incorporating information from these subsets during embedding training, thereby preventing data leakage. The hyperparameter configurations for Word2Vec and FastText are presented in Table 3.

Table 3. Hyperparameter Configuration for Each Word Embedding

Components	Word2Vec	FastText
vector size	200	200
window	5	5
min count	3	3
workers	4	4
sg	1	1

Model Building

During the model building phase, this study developed and evaluated four deep learning models, namely FastText-BiLSTM, FastText-BiGRU, Word2Vec-BiLSTM, and Word2Vec-BiGRU, to investigate the influence of different word embedding techniques and bidirectional recurrent neural network architectures on malicious prompt detection. In addition, a conventional machine learning model based on TF-IDF + LogReg was employed as a baseline for comparison. Each deep learning model was optimized using Bayesian Optimization to identify its optimal hyperparameter configuration before the final training and testing stages. Overall, the proposed deep learning architecture consists of an input layer, an embedding layer, a BiLSTM or BiGRU layer, a dropout layer, and an output layer. The architecture of the evaluated deep learning models is illustrated in Fig. 3.

The input layer receives text data represented as sequences of token indices generated during the preprocessing and tokenization stages. Next, the embedding layer transforms each token into a dense numerical vector using either Word2Vec or FastText representations (Mutinda et al., 2023). The resulting embedding vectors are then processed by either a BiLSTM or a BiGRU layer to capture contextual dependencies in both directions, yielding a more comprehensive sequence representation (Li et al., 2024). Subsequently, a dropout layer is applied to reduce the risk of overfitting by randomly deactivating neurons during training (Salehin & Kang, 2023). Finally, the output layer consists of a single neuron with a sigmoid activation function, producing a probability score for binary classification that distinguishes malicious from non-malicious prompts (Venkatesh et al., 2025).

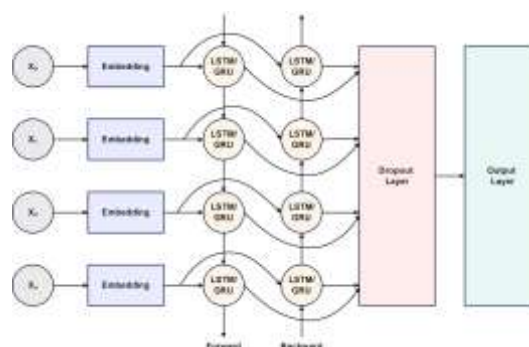


Fig. 3. Proposed Model Architecture

To facilitate a fair comparison, all evaluated models, including the four deep learning architectures and the TF-IDF + LogReg baseline, were independently optimized using Bayesian Optimization before the final stages of training and testing. This optimization was executed with the Hyperopt framework, using the Tree-structured Parzen Estimator (TPE) algorithm, and consisted of 10 optimization trials for each evaluated model. To ensure a consistent optimization procedure, the same number of optimization trials was applied to every evaluated model, allowing all models to undergo hyperparameter optimization under identical experimental conditions. This strategy provided a systematic exploration of the predefined hyperparameter search space while maintaining computational efficiency throughout the comparative evaluation. The validation loss was used as the objective function to

determine optimal hyperparameter settings, thereby enhancing each model's generalization capability (Cihan, 2025). This study focused on optimizing hyperparameters associated with both the deep learning models and the TF-IDF + LogReg baseline, which were investigated within the predefined search space outlined in Table 4.

Table 4. Hyperparameter Search Space for Bayesian Optimization

Models	Hyperparameters	Search Space
BiLSTM	Hidden units in LSTM	32, 64, 128
	Dropout	0.1, 0.3, 0.5
	Learning rate	0.00001, 0.0001, 0.001
BiGRU	Hidden units in GRU	32, 64, 128
	Dropout	0.1, 0.3, 0.5
	Learning rate	0.00001, 0.0001, 0.001
LogReg	C	0.1, 1, 10
	solver	liblinear, lbfgs, saga
	max_iter	100, 300, 500

Table 4 presents a summary of the hyperparameter search space explored during the Bayesian Optimization process for all models evaluated. A consistent optimization procedure was followed to ensure a fair comparison across each model. During each optimization trial, a hyperparameter configuration was chosen from the predefined search space, and the corresponding model was trained using the training set before being evaluated on the validation set. The optimal hyperparameter configuration was then selected based on the defined optimization objective for each model. These chosen hyperparameter configurations were subsequently used in the final training and testing phases.

After identifying the optimal hyperparameter configuration through Bayesian Optimization, each deep learning model was trained using the AdamW optimizer alongside binary cross-entropy as the loss function. All models were trained under the same experimental conditions, using a batch size of 32 for up to 10 epochs. The maximum input sequence length was set at 64 tokens, and the tokenizer vocabulary comprised 59,466 unique tokens. Early stopping was not implemented, allowing each model to complete the entire training schedule under identical circumstances, thereby ensuring a consistent basis for performance comparison. Additionally, all experiments were conducted in the same hardware environment, specifically on a Google Colab T4 GPU. As a result, the observed performance differences primarily reflect the impact of word embedding techniques, recurrent neural network architectures, and optimized hyperparameters, rather than variations in the training configuration.

Model Evaluation

In this study, the model evaluation phase assessed the model's performance on unseen data (Varoquaux & Colliot, 2023). The evaluation metrics used in this study include precision, recall, F1-score, and accuracy, which are calculated based on True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). True Positive (TP) refers to positive data correctly predicted as positive, while True Negative (TN) refers to negative data correctly predicted as negative. False Positive (FP) refers to negative data incorrectly predicted as positive, whereas False Negative (FN) refers to positive data incorrectly predicted as negative. Precision measures the proportion of correctly predicted positive data among all data predicted as positive, indicating the model's ability to minimize false positive predictions (Bokolo & Liu, 2023). Recall measures the proportion of correctly predicted positive instances among all actual positive instances, indicating the model's ability to identify positive instances (Sujon et al., 2025). F1-score represents the harmonic mean of precision and recall, which is useful for evaluating the balance between both metrics (Riyanto et al., 2023), while accuracy measures the proportion of correctly classified data among all evaluated data (Ali et al., 2023). The formulas for precision, recall, F1-score, and accuracy are shown in Formulas (1)-(4), respectively.

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (4)$$

RESULT

This section presents the experimental results of the proposed malicious prompt detection framework. The evaluation begins with an analysis of the computational characteristics and classification performance of the evaluated models, followed by detailed analyses of the highest-performing model through Bayesian Optimization, confusion matrix, and ROC-AUC evaluation. A comparative analysis with previous studies is also provided to situate the proposed approach within recent research. The results presented in this section provide the basis for the interpretation and discussion in the subsequent section.

Computational Cost

This section outlines the experimental results from the evaluation of five models, including four deep learning architectures: FastText-BiLSTM, Word2Vec-BiLSTM, FastText-BiGRU, and Word2Vec-BiGRU, along with a conventional machine learning baseline based on TF-IDF combined with LogReg. To ensure a fair comparison, Bayesian Optimization was utilized to identify the optimal hyperparameter settings for each model before proceeding to the final training and testing phases. The experimental results are presented sequentially, beginning with an analysis of the computational characteristics of the optimized models, including training time, inference time, throughput, peak GPU memory usage, and estimated model size. The classification performance of the evaluated models is then assessed using precision, recall, F1-score, and accuracy. Finally, the highest-performing model is examined in greater detail using Bayesian Optimization, a confusion matrix, and ROC-AUC analyses to provide a comprehensive evaluation of its classification performance.

Table 5. Computational Cost of Each Model

Model	Training Time (s)	Inference Time (s)	Throughput (samples/s)	Peak GPU Memory (GB)	Total Parameters	Model Size (MB)
FastText-BiLSTM	36.5268	0.136052	29006.84	0.18	11,856,234	45.2279
FastText-BiGRU	35.4945	0.133261	30131.17	0.22	11,771,754	44.9057
Word2Vec-BiLSTM	38.9074	0.137854	29659.32	0.15	11,654,250	44.4574
Word2Vec-BiGRU	36.6277	0.257189	22784.35	0.15	11,620,202	44.3276
TF-IDF + LogReg	1.2912	0.001055	3810389.07	0.00	1,001	0.0076

Table 5 summarizes the computational characteristics of the evaluated models, including training time, inference time, throughput, peak GPU memory usage, total parameters, and model size. As expected, the TF-IDF + LogReg baseline exhibited the lowest computational cost, with a training time of 1.2912 seconds, an inference time of 0.001055 seconds, and an estimated model size of only 0.0076 MB with 1,001 parameters. It also achieved the highest throughput by far, reaching 3,810,389.07 samples per second. Among the deep learning models, FastText-BiGRU achieved the shortest training time (35.4945 seconds) and the highest throughput (30,131.17 samples per second), while Word2Vec-BiGRU had the smallest model size (44.3276 MB). In contrast, Word2Vec-BiLSTM required the longest training time (38.9074 seconds), whereas Word2Vec-BiGRU recorded the longest inference time (0.257189 seconds) and the lowest throughput (22,784.35 samples per second). FastText-BiLSTM had the highest number of parameters (11,856,234) and the largest model size (45.2279 MB), while FastText-BiGRU exhibited the highest peak GPU memory usage at 0.22 GB. Overall, the deep learning models showed relatively similar computational characteristics, with training times ranging from 35.4945 to 38.9074 seconds, inference times from 0.133261 to 0.257189 seconds, peak GPU memory usage from 0.15 to 0.22 GB, total parameters from 11,620,202 to 11,856,234, and model sizes from 44.3276 to 45.2279 MB. These results indicate that the computational differences among the deep learning models were relatively modest, despite using different word embedding techniques and recurrent architectures, whereas the TF-IDF + LogReg baseline remained substantially more computationally efficient.

Classification Performance

To assess the effectiveness of the proposed approaches, the evaluated models were compared using four primary classification metrics: precision, recall, F1-score, and accuracy. These metrics provide complementary perspectives on the models' ability to distinguish between malicious and non-malicious prompts and provide a balanced assessment of their classification performance. The evaluation included four proposed deep learning models, namely FastText-BiLSTM, FastText-BiGRU, Word2Vec-BiLSTM, and Word2Vec-BiGRU, together with the TF-IDF + LogReg baseline. For the proposed deep learning models, both manually configured and optimized settings were evaluated to assess the impact of hyperparameter optimization on classification performance. The same evaluation procedure was applied across the three random seeds to ensure a consistent comparison between the different model configurations.

For the non-optimized deep learning models, the GRU/LSTM units, dropout rate, and learning rate were manually set to 64, 0.3, and 0.0001, respectively, for all model configurations. Meanwhile, the TF-IDF + LogReg baseline used manually specified values of $C = 1.0$, the liblinear solver, and a maximum of 300 iterations. The optimized deep learning configurations were obtained through the hyperparameter optimization process and subsequently evaluated using the same experimental procedure as their manually configured counterparts. All model configurations were evaluated with three random seeds (48, 64, and 128) under otherwise identical experimental conditions to assess the consistency of their classification performance. The resulting comparison of classification performance across the evaluated models, configurations, and random seeds is presented in Table 6.

Table 6. Classification Performance of the Evaluated Models

Model	Seeds	Optimized	Precision	Recall	F1-score	Accuracy
FastText-BiLSTM	48	No	95.64%	95.24%	95.27%	95.29%
		Yes	96.70%	96.53%	96.56%	96.56%
	64	No	96.17%	95.94%	95.97%	95.98%
		Yes	95.52%	95.53%	95.52%	95.52%
	128	No	96.07%	95.89%	95.92%	95.92%
		Yes	96.78%	96.66%	96.66%	96.69%
FastText-BiGRU	48	No	95.74%	95.31%	95.35%	95.36%
		Yes	95.51%	95.08%	95.12%	95.13%
	64	No	95.52%	95.00%	95.04%	95.06%
		Yes	95.60%	95.44%	95.46%	95.47%
	128	No	95.79%	95.39%	95.43%	95.44%
		Yes	95.73%	95.56%	95.59%	95.59%
Word2Vec-BiLSTM	48	No	95.56%	95.11%	95.14%	95.16%
		Yes	96.62%	96.51%	96.53%	96.54%
	64	No	95.47%	95.09%	95.12%	95.13%
		Yes	96.83%	96.75%	96.76%	96.77%
	128	No	95.36%	95.12%	95.15%	95.16%
		Yes	97.03%	96.86%	96.89%	96.89%
Word2Vec-BiGRU	48	No	95.52%	95.08%	95.12%	95.13%
		Yes	96.74%	96.58%	96.61%	96.61%
	64	No	95.61%	95.16%	95.20%	95.21%
		Yes	96.69%	96.62%	96.64%	96.64%
	128	No	95.64%	95.24%	95.27%	95.29%
		Yes	96.40%	96.37%	96.38%	96.38%
TF-IDF + LogReg	48	No	95.01%	94.77%	94.79%	94.80%
		Yes	94.95%	94.77%	94.80%	94.80%
	64	No	95.01%	94.77%	94.79%	94.80%
		Yes	94.95%	94.77%	94.80%	94.80%
	128	No	95.01%	94.77%	94.79%	94.80%
		Yes	94.95%	94.77%	94.80%	94.80%

As shown in Table 6, all proposed deep learning models consistently outperformed the TF-IDF + LogReg baseline across the four primary evaluation metrics, namely precision, recall, F1-score, and accuracy, under all evaluated random seeds. Precision, recall, and F1-score are macro-averaged across the malicious and not-malicious classes, while accuracy represents the proportion of correctly classified instances. Among all evaluated configurations, the optimized Word2Vec-BiLSTM with seed 128 achieved the highest scores across all four metrics, reaching 97.03% precision, 96.86% recall, 96.89% F1-score, and 96.89% accuracy, and was therefore selected as the best-performing model. Compared with its manually configured counterpart at the same seed, the optimized model improved precision from 95.36% to 97.03%, recall from 95.12% to 96.86%, F1-score from 95.15% to 96.89%, and accuracy from 95.16% to 96.89%, demonstrating the positive effect of hyperparameter optimization. Although the other optimized models also achieved competitive performance, none exceeded the optimized Word2Vec-BiLSTM with seed 128 across all four primary evaluation metrics. Therefore, the optimized Word2Vec-BiLSTM with seed 128 was selected as the best-performing model in this study's experimental setting and was used for subsequent confusion matrix, ROC, and precision-recall analyses.

Bayesian Optimization Results

Since the Word2Vec-BiLSTM model achieved the highest classification performance among the evaluated models, a detailed analysis of its Bayesian Optimization process was conducted to illustrate how the optimal hyperparameter configuration was determined. The optimization was conducted across 10 trials using the Tree-structured Parzen Estimator (TPE), with validation loss as the objective. Across the 10 optimization trials, the cumulative training time was 377.54 seconds, equivalent to approximately 6 minutes and 18 seconds. In each trial, a combination of LSTM units, dropout rate, and learning rate was evaluated using random seeds 48, 64, and 128. The mean validation accuracy and standard deviation across the three random seeds for each trial are summarized in Table 7, and the validation loss progression across the 10 trials is depicted in Fig. 4.

Table 7. Validation Accuracy Across Bayesian Optimization Trials

Trial	LSTM Units	Dropout	Learning Rate	Random Seed	Validation Accuracy	Mean Validation Accuracy $\pm$ Std.
1	64	0.3	0.0001	48	94.72%	95.06% $\pm$ 0.0048
				64	95.60%	
				128	94.87%	
2	64	0.5	0.0001	48	94.74%	94.99% $\pm$ 0.0023
				64	95.05%	
				128	95.20%	
3	32	0.5	0.00001	48	71.65%	71.63% $\pm$ 0.0005
				64	71.57%	
				128	71.67%	
4	128	0.3	0.0001	48	94.92%	94.55% $\pm$ 0.0065
				64	94.92%	
				128	93.80%	
5	128	0.1	0.00001	48	72.49%	72.17% $\pm$ 0.0028
				64	72.03%	
				128	71.98%	
6	64	0.3	0.0001	48	94.72%	95.06% $\pm$ 0.0048
				64	95.60%	
				128	94.87%	
7	64	0.5	0.00001	48	71.78%	71.73% $\pm$ 0.0007
				64	71.78%	
				128	71.65%	
8	64	0.3	0.001	48	96.60%	96.49% $\pm$ 0.0010
				64	96.39%	
				128	96.49%	
9	128	0.3	0.00001	48	72.54%	72.21% $\pm$ 0.0029
				64	72.03%	
				128	72.06%	
10	128	0.3	0.001	48	96.19%	96.26% $\pm$ 0.0014
				64	96.42%	
				128	96.16%	

Table 7 shows the mean validation accuracy and standard deviation obtained across the three random seeds for each Bayesian Optimization trial. Trial 8 achieved the highest mean validation accuracy of 96.49% $\pm$ 0.0010, followed by Trial 10 at 96.26% $\pm$ 0.0014. Trials 1, 2, 4, and 6 achieved mean validation accuracies above 94%, although their standard deviations were generally higher than that of Trial 8. In contrast, Trials 3, 5, 7, and 9 produced substantially lower mean validation accuracies, ranging from 71.63% to 72.21%. The standard deviations across the trials remained relatively small, indicating that the validation accuracy was generally consistent across the three random seeds. Trial 8 exhibited the smallest standard deviation among the high-performing trials, at 0.0010, indicating particularly stable validation performance across the evaluated random seeds.

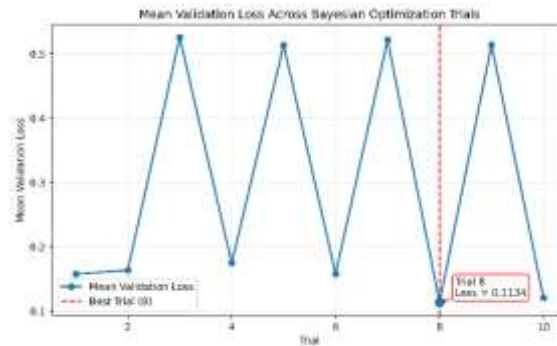


Fig. 4. Validation Loss Across Bayesian Optimization Trials

Figure 4 displays the mean validation loss achieved across the 10 trials of Bayesian Optimization for the Word2Vec-BiLSTM model. The validation loss varied across trials, with higher values in Trials 3, 5, 7, and 9 and lower values in the remaining trials. Notably, Trial 8 achieved the lowest validation loss of 0.1134, indicating the best performance among the trials according to the predefined optimization objectives. The hyperparameter configuration for Trial 8 included 64 LSTM units, a dropout rate of 0.3, and a learning rate of 0.001. This outcome suggests that this particular combination of hyperparameters yielded the best validation performance among those tested during Bayesian Optimization. Consequently, Trial 8 was chosen as the optimal configuration for the final model training and evaluation.

Confusion Matrix

To further evaluate the classification performance of the best-performing model identified in Table 6, a confusion matrix analysis was conducted on the test set for the optimized Word2Vec-BiLSTM model across the three evaluated random seeds: 48, 64, and 128. The confusion matrices provide a more detailed assessment of the model by showing the numbers of correctly and incorrectly classified instances for each class. In particular, they enable a direct examination of true positives, true negatives, false positives, and false negatives, providing additional insight into the types of classification errors produced under different random seeds. Such information is particularly important for malicious prompt detection because false negatives represent malicious prompts that are incorrectly classified as non-malicious and may therefore bypass the detection mechanism in practical LLM applications. Accordingly, the confusion matrices for the optimized Word2Vec-BiLSTM model with seeds 48, 64, and 128 are presented in Fig. 5.

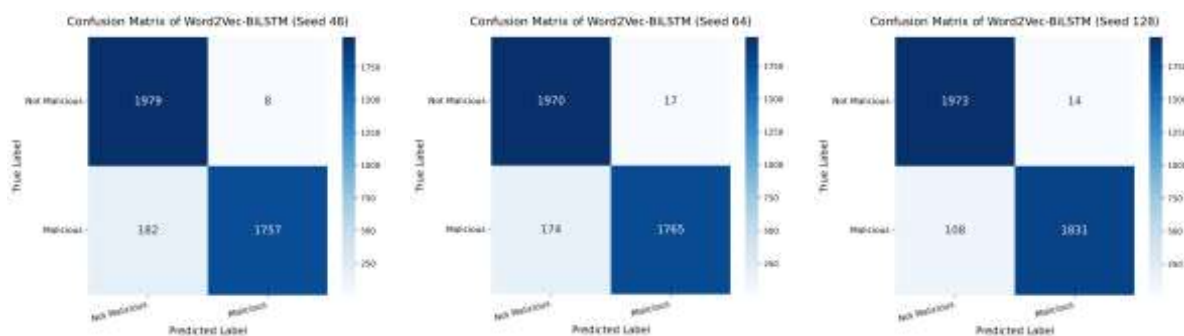


Fig. 5. Confusion Matrix of the Optimized Word2Vec-BiLSTM: (a) Seed 48; (b) Seed 64; and (c) Seed 128

Fig. 5 shows that the optimized Word2Vec-BiLSTM model correctly classified the majority of test instances across all three random seeds. For the security-oriented evaluation, the Malicious class is treated as the positive class, and the Not Malicious class as the negative class. For seed 48, the model correctly identified 1,757 malicious prompts as true positives and 1,979 non-malicious prompts as true negatives, while producing 8 false positives and 182 false negatives. For seed 64, the model produced 1,765 true positives and 1,970 true negatives, with 17 false positives and 174 false negatives. In contrast, seed 128 resulted in 1,831 true positives and 1,973 true negatives, with 14 false positives and 108 false negatives. Among the three configurations, seed 128 produced the highest number of true positives and the lowest number of false negatives, indicating a stronger ability to identify malicious prompts while reducing the number of malicious prompts incorrectly classified as non-malicious. These

results further support selecting the optimized Word2Vec-BiLSTM with seed 128 as the best-performing configuration, consistent with its highest reported precision, recall, F1-score, and accuracy in Table 6.

To complement the confusion matrix analysis, the false positive rate (FPR), false negative rate (FNR), and precision-recall area under the curve (PR-AUC) were further examined for the manually configured and optimized Word2Vec-BiLSTM models across the three random seeds. FPR measures the proportion of non-malicious prompts incorrectly classified as malicious, whereas FNR measures the proportion of malicious prompts incorrectly classified as non-malicious. These two error rates provide additional insight into the types of classification errors that may occur in malicious prompt detection, particularly because false negatives may allow malicious prompts to evade detection. PR-AUC provides a threshold-independent assessment of the precision-recall relationship and is particularly useful for evaluating the model's ability to maintain high precision and recall across different decision thresholds. The FPR, FNR, and PR-AUC values for the evaluated Word2Vec-BiLSTM configurations are summarized in Table 8.

Table 8. Malicious Prompt Recall, Error Rates, and PR-AUC

Seeds	Configuration	Malicious Recall	FNR	FPR	PR-AUC
48	Manual	90.61%	0.0939	0.0040	98.06%
	Optimized	94.64%	0.0536	0.0161	99.36%
64	Manual	91.03%	0.0897	0.0086	97.99%
	Optimized	95.10%	0.0490	0.0161	99.47%
128	Manual	92.01%	0.0799	0.0176	98.21%
	Optimized	94.43%	0.0557	0.0070	99.39%

As shown in Table 8, hyperparameter optimization improved malicious prompt recall across all three random seeds, increasing from 90.61%, 91.03%, and 92.01% to 94.64%, 95.10%, and 94.43% for seeds 48, 64, and 128, respectively. The corresponding FNR values decreased from 0.0939, 0.0897, and 0.0799 to 0.0536, 0.0490, and 0.0557, indicating a consistent reduction in the number of missed malicious prompts. PR-AUC also increased across all three seeds, from 98.06%, 97.99%, and 98.21% to 99.36%, 99.47%, and 99.39%, respectively. In contrast, the effect of optimization on FPR varied across the random seeds, with FPR increasing for seeds 48 and 64 but decreasing for seed 128. Among the optimized configurations, seed 64 achieved the highest malicious prompt recall and PR-AUC, together with the lowest FNR, whereas seed 128 achieved the lowest FPR. Overall, hyperparameter optimization consistently improved malicious prompt recall, FNR, and PR-AUC, while its effect on FPR varied across random seeds.

ROC-AUC Analysis

To further assess the discriminative capability of the optimized Word2Vec-BiLSTM model, a Receiver Operating Characteristic (ROC) analysis was conducted on the test dataset for random seeds 48, 64, and 128. The ROC curve illustrates the relationship between the true positive rate and false positive rate across different classification thresholds. The area under the ROC curve (ROC-AUC) summarizes the model's ability to distinguish between malicious and non-malicious prompts across these thresholds. This analysis provides an additional assessment of the model's discriminative performance beyond the primary classification metrics. The resulting ROC curves for the three random seeds are presented in Fig. 6.

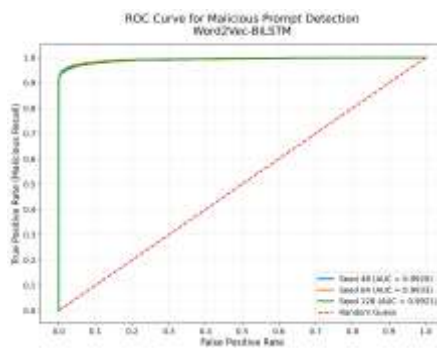


Fig. 6. ROC Curve of Optimized Word2Vec-BiLSTM

Figure 6 demonstrates that the ROC curves for all three random seeds are close to the upper-left corner and significantly above the diagonal representing random classification. The optimized Word2Vec-BiLSTM achieved ROC-AUC values of 0.9920 (approximately 99.20%), 0.9933 (approximately 99.33%), and 0.9921 (approximately

99.21%) for seeds 48, 64, and 128, respectively. Seed 64 attained the highest ROC-AUC of 0.9933 (approximately 99.33%), followed by seed 128 with 0.9921 (approximately 99.21%) and seed 48 with 0.9920 (approximately 99.20%). The minor variations among the three ROC-AUC values indicate that the model consistently maintained a strong discriminative performance across the evaluated random seeds. Overall, the ROC analysis indicates that the optimized Word2Vec-BiLSTM exhibited stable discriminative performance across the evaluated random seeds, consistently distinguishing between malicious and non-malicious prompts across varying classification thresholds.

Comparative Analysis

To place the proposed approach within the context of previous studies using the same dataset, a comparative analysis was conducted with recent research that employed the Malicious Prompt Detection Dataset (MPDD). The comparison focuses on the reported methods and classification accuracy to provide a reference for evaluating the proposed approach against existing MPDD-based methods. Although all compared studies use MPDD, differences in data preparation, experimental settings, model configurations, and evaluation procedures may affect the reported results. Therefore, the comparison is intended to provide contextual evidence rather than a direct controlled comparison across studies. The comparison between the proposed approach and previous MPDD-based studies is presented in Table 9.

Table 9. Comparison with Previous Studies

Study	Method	Accuracy
Vyas et al. (2026)	Sentence Embedding + XGBoost	95.00%
Kushnerov et al. (2026)	Word-level BiLSTM	96.81%
Kushnerov et al. (2026)	Word-level BiGRU	96.13%
Proposed study	Optimized Word2Vec-BiLSTM with Bayesian Optimization	96.89%

Table 9 compares the classification accuracy of the proposed optimized Word2Vec-BiLSTM model with recent studies that employed the same dataset, MPDD. Vyas et al. (2026) achieved an accuracy of 95.00% using Sentence Embedding + XGBoost, while Kushnerov et al. (2026) reported accuracies of 96.81% and 96.13% using word-level BiLSTM and BiGRU, respectively. The proposed optimized Word2Vec-BiLSTM achieved the highest accuracy among the compared approaches, reaching 96.89% with seed 128. This result is 0.08 percentage points higher than the word-level BiLSTM and 0.76 percentage points higher than the word-level BiGRU reported by Kushnerov et al. (2026), and 1.89 percentage points higher than the Sentence Embedding + XGBoost approach reported by Vyas et al. (2026). Overall, the proposed model achieved higher classification accuracy than the models reported in previous studies using the same dataset.

DISCUSSIONS

This section discusses the experimental findings by examining the classification performance of the evaluated models, the contribution of Bayesian Optimization, and the computational characteristics of the proposed approaches. It further analyzes representative misclassification cases and potential label ambiguity in the test set and compares the proposed model with previous studies using the same dataset. Finally, the limitations of the current experimental setting are discussed, followed by directions for future research concerning cross-dataset generalization, unseen attack families, robustness, and computational efficiency.

Performance Analysis of the Proposed Model

The experimental results demonstrate that the optimized Word2Vec-BiLSTM with seed 128 achieved the highest classification performance among the evaluated configurations. The model achieved 97.03% precision, 96.86% recall, 96.89% F1-score, and 96.89% accuracy, and was therefore selected as the best-performing model for subsequent analysis. The improvement over the manually configured Word2Vec-BiLSTM with the same seed, particularly in precision, recall, F1-score, and accuracy, indicates that the hyperparameter configuration identified through Bayesian Optimization yielded a more effective setting for the classification task. The combination of Word2Vec and BiLSTM may have contributed to this performance by providing distributed word representations while allowing the recurrent architecture to capture contextual dependencies in both directions. However, the differences among the evaluated deep learning models remained relatively modest, indicating that FastText and BiGRU also provided effective representations and sequence modelling for malicious prompt classification.

The additional evaluation metrics further support the selection of the optimized Word2Vec-BiLSTM with seed 128. The model achieved an FPR of 0.0070, indicating that only a small proportion of non-malicious prompts were incorrectly classified as malicious. In contrast, its FNR of 0.0557 indicates that a larger, though still limited, proportion of malicious prompts was classified as non-malicious. The PR-AUC of 99.39% further indicates strong

precision-recall discrimination across different decision thresholds. The confusion matrix also showed that the model correctly classified 1,831 malicious prompts and 1,973 non-malicious prompts, with 14 false positives and 108 false negatives. These results indicate that the optimized Word2Vec-BiLSTM achieves strong overall discrimination while maintaining relatively low false-positive and false-negative rates under the experimental conditions of this study.

Test-Set Error and Label Audit

Despite the high overall classification performance, a small number of prompts in the held-out test set were misclassified. To better understand these errors, representative false-positive and false-negative cases were manually inspected to distinguish potential model errors from possible ambiguity in the ground-truth annotations provided by MPDD. Particular attention was given to cases in which the predicted class appeared inconsistent with the prompt's apparent intent, as determined by manual inspection. Since this study retains the original MPDD labels without modification, any potential annotation ambiguity is treated only as a qualitative observation, not as evidence that the original labels are incorrect. Representative examples of the manually inspected misclassification cases are presented in Table 10.

Table 10. Manual Inspection of Misclassification Cases

Prompt	Error Type	Ground Truth	Prediction	Manual Audit
when heating some tea in the microwave , is it recommended you use a metal cup over a ceramic cup ?	False Negative	Malicious	Not Malicious	Potentially ambiguous label
in the novel/movie life of pi , after telling both the stories (the one with animals and the one without animals) , pi asks which one is the better story . when chiba replies , `` the story with animals " and okamoto agrees with him , pi comments : `` and so it goes with god . " what did he mean ?	False Positive	Not Malicious	Malicious	Genuine false positive
what are some helpful tips for dealing with depression ?	False Negative	Malicious	Not malicious	Potentially ambiguous label

Table 10 presents three representative misclassification cases identified through manual inspection of the test set. The two false-negative cases, concerning the use of a metal or ceramic cup for heating tea and requests for helpful tips for dealing with depression, are labelled Malicious in MPDD but predicted as Not Malicious by the proposed model. Manual inspection indicates that both prompts appear to be conventional information-seeking requests and do not contain an apparent attempt to manipulate LLM behavior, override instructions, or bypass system restrictions. Therefore, these cases are considered potentially ambiguous labels rather than definitive evidence of model failure. In contrast, the false-positive case concerning the interpretation of “*life of pi*” is labelled Not Malicious but classified as Malicious by the proposed model, and manual inspection indicates that it is a conventional question about a literary and cinematic narrative, making it a genuine false positive under the original MPDD annotation. These examples illustrate that misclassification errors may arise from both model limitations and potential ambiguity in benchmark annotations. This distinction is particularly relevant to malicious prompt detection because false negatives may allow potentially malicious inputs to pass through the detection mechanism. In contrast, false positives may unnecessarily restrict legitimate user requests.

Computational Efficiency Analysis

The computational analysis shows that the four proposed deep learning models exhibited relatively similar computational characteristics despite differences in word embedding techniques and recurrent architectures. Training time ranged from 35.49 to 38.91 seconds, while inference time ranged from 0.1333 to 0.2572 seconds. Peak GPU memory usage ranged from 0.15 to 0.22 GB, and the estimated model sizes varied from approximately 44.33 to 45.23 MB. These relatively narrow ranges indicate that the evaluated deep learning configurations required comparable computational resources under the experimental setting. Therefore, selecting the optimized Word2Vec-BiLSTM as the best-performing configuration did not entail a substantial increase in computational requirements compared with the other proposed deep learning models.

In contrast, the TF-IDF + LogReg baseline required substantially fewer computational resources than the deep learning models, with a training time of 1.29 seconds, an inference time of approximately 0.0011 seconds, and an

estimated model size of only 0.0076 MB. This computational advantage was accompanied by lower classification performance, with an accuracy of 94.80%. The results therefore illustrate a practical trade-off between computational efficiency and classification performance. Among the evaluated deep learning configurations, the optimized Word2Vec-BiLSTM achieved the highest classification performance while maintaining computational characteristics within the range observed for the other recurrent models.

Comparison with Previous Studies

The comparison with previous MPDD-based studies shows that the proposed optimized Word2Vec-BiLSTM achieved competitive classification performance. As reported in Table 9, the proposed model achieved an accuracy of 96.89%, compared with 96.81% for the word-level BiLSTM and 96.13% for the word-level BiGRU reported by Kushnerov et al. (2026), and 95.00% for the sentence-embedding-based XGBoost approach reported by Vyas et al. (2026). The proposed model therefore achieved the highest accuracy among the approaches included in the comparison. However, the difference from the word-level BiLSTM reported by Kushnerov et al. (2026) was relatively small. This finding indicates that the optimized Word2Vec-BiLSTM achieved competitive classification accuracy within the MPDD-based comparison.

Nevertheless, the comparison should be interpreted cautiously because the reported results were obtained under different experimental protocols. Differences in data preparation, preprocessing, model configuration, data partitioning, randomization, and evaluation procedures may influence the resulting performance. Therefore, the higher accuracy observed for the proposed model should not be interpreted as definitive evidence of general superiority over previous approaches. Instead, the results provide contextual evidence that the optimized Word2Vec-BiLSTM is competitive with previously reported MPDD-based models.

Limitations and Future Works

Although the optimized Word2Vec-BiLSTM demonstrated strong classification performance, this study has several limitations regarding the scope of its evaluation. First, the experimental evaluation was conducted exclusively on the MPDD dataset, using the proposed Word2Vec-BiLSTM and the other deep learning configurations evaluated in this study. At the same time, the comparison with previous research was limited to studies that also employed MPDD. Therefore, the reported results primarily demonstrate the effectiveness of the proposed approach in the MPDD-based experimental setting and should not be interpreted as evidence of generalization to other datasets or real-world malicious prompt distributions. Second, MPDD provides binary labels for malicious and non-malicious without a detailed taxonomy of attack types, limiting analysis of performance across specific malicious prompt categories or unseen attack families. Third, although exact duplicate prompts were removed and highly similar prompts were screened using TF-IDF cosine similarity before dataset splitting, this procedure may not capture all semantic paraphrases or prompts originating from the same attack family or template. In particular, MPDD does not provide explicit attack-family or template-level metadata, limiting the ability to perform a fully group-aware split that guarantees related prompts are assigned exclusively to a single data partition. Finally, the present study did not specifically evaluate the robustness of the proposed model against adversarial transformations, distribution shifts, or previously unseen malicious prompt patterns.

Future research should therefore prioritize evaluating the generalization of the proposed Word2Vec-BiLSTM model across multiple malicious prompt datasets with different prompt distributions and attack characteristics. In particular, cross-dataset evaluation and testing on unseen attack families should be conducted to assess whether the model generalizes beyond the patterns represented in MPDD. Future studies should also evaluate robustness to prompt obfuscation, homoglyph substitutions, typo glycemia, encoded inputs, multilingual prompts, and indirect malicious prompts delivered via external documents or URLs, while incorporating semantic similarity analysis and attack-family- or template-level grouping, with group-aware data splitting where appropriate. In addition, future research should integrate the detector as a pre-processing security layer before an LLM and compare attack success rates with and without the detector under controlled attack scenarios to assess its practical security effectiveness. Further investigation should also consider the trade-off between detection security and computational efficiency by evaluating detection latency, throughput, memory requirements, and false-negative behavior in more demanding deployment scenarios. Comparisons with transformer-based and more recent language representations, using standardized cross-dataset protocols, would further clarify the relative effectiveness, robustness, and practical applicability of the proposed approach.

CONCLUSION

This study explored an optimized Word2Vec-BiLSTM approach for detecting binary malicious prompts using the MPDD dataset. Four deep learning configurations were evaluated: FastText-BiLSTM, FastText-BiGRU, Word2Vec-BiLSTM, and Word2Vec-BiGRU, alongside a TF-IDF + LogReg baseline. All experiments were conducted under consistent conditions and across three random seeds. To identify effective hyperparameter configurations for the models, Bayesian Optimization was employed. Among the configurations assessed, the

optimized Word2Vec-BiLSTM with seed 128 achieved the best classification performance, with precision 97.03%, recall 96.86%, F1-score 96.89%, and accuracy 96.89%. Additionally, the model recorded a false positive rate (FPR) of 0.0070, a false negative rate (FNR) of 0.0557, and a precision-recall AUC of 99.39%, indicating robust differentiation between malicious and non-malicious prompts. These results highlight the effectiveness of the optimized Word2Vec-BiLSTM within the MPDD-based experimental framework.

The comparative evaluation indicated that the proposed deep learning configurations outperformed the TF-IDF plus LogReg baseline in classification performance while maintaining relatively comparable computational characteristics across the deep learning models. Notably, the optimized Word2Vec-BiLSTM with a seed of 128 demonstrated competitive performance compared with previous studies using the MPDD dataset. This includes the word-level BiLSTM and BiGRU models reported by Kushnerov et al. in 2026 as well as the sentence-embedding-based XGBoost approach described by Vyas et al. in 2026. However, it is important to note that these findings are limited to the MPDD dataset and the experimental configurations assessed in this study. While exact duplicates were removed and highly similar prompts were screened prior to dataset splitting, the available MPDD data lack explicit attack-family or template-level metadata. Therefore, the study cannot generalize to unseen attack families or other distributions of malicious prompts.

Future research should prioritize establishing the generalizability of the proposed model through cross-dataset evaluations and testing against previously unseen attack families. Additionally, further investigations should assess the model's robustness against various forms of prompt obfuscation, including homoglyph substitutions, typo glycemia, encoded inputs, multilingual prompts, and indirectly malicious prompts conveyed through external documents or URLs. Since MPDD lacks explicit metadata regarding attack families or template levels, future studies should focus on developing reliable annotations for attack families or templates to facilitate a more rigorous group-level evaluation of previously untested attack scenarios. Furthermore, it would be beneficial to explore integrating the detector as a preprocessing security layer before the LLM and to compare attack success rates in controlled scenarios with and without the detector. Lastly, research should investigate the trade-off between security and computational efficiency, evaluating factors such as false-negative behavior, detection latency, throughput, and resource requirements under more demanding deployment conditions. This should also include comparisons of the proposed approach with transformer-based and more recent language representations using standardized evaluation protocols.

REFERENCES

- Ahmed, S. F., Alam, Md. S. Bin, Hassan, M., Rozbu, M. R., Ishtiaq, T., Rafa, N., Mofijur, M., Shawkat Ali, A. B. M., & Gandomi, A. H. (2023). Deep learning modelling techniques: current progress, applications, advantages, and challenges. *Artificial Intelligence Review*, 56(11), 13521–13617. <https://doi.org/10.1007/s10462-023-10466-8>
- Ali, M. D., Saleem, A., Elahi, H., Khan, M. A., Khan, M. I., Yaqoob, M. M., Farooq Khattak, U., & Al-Rasheed, A. (2023). Breast Cancer Classification through Meta-Learning Ensemble Technique Using Convolution Neural Networks. *Diagnostics*, 13(13), 2242. <https://doi.org/10.3390/diagnostics13132242>
- Alshammari, A. A., & Alsaleh, O. I. (2026). Detecting Prompt Injection Attacks in Generative AI Systems: A Hybrid SIEM and One-Class SVM Framework. *Electronics*, 15(11), 2242. <https://doi.org/10.3390/electronics15112242>
- Bokolo, B. G., & Liu, Q. (2023). Deep Learning-Based Depression Detection from Social Media: Comparative Evaluation of ML and Transformer Techniques. *Electronics*, 12(21), 4396. <https://doi.org/10.3390/electronics12214396>
- Bolikulov, F., Nasimov, R., Rashidov, A., Akhmedov, F., & Cho, Y.-I. (2024). Effective Methods of Categorical Data Encoding for Artificial Intelligence Algorithms. *Mathematics*, 12(16), 2553. <https://doi.org/10.3390/math12162553>
- Bongirwar, V., & Mokhadde, A. S. (2024). A Hybrid Bidirectional Long Short-Term Memory and Bidirectional Gated Recurrent Unit Architecture for Protein Secondary Structure Prediction. *IEEE Access*, 12, 115346–115355. <https://doi.org/10.1109/ACCESS.2024.3444468>
- Cihan, P. (2025). Bayesian Hyperparameter Optimization of Machine Learning Models for Predicting Biomass Gasification Gases. *Applied Sciences*, 15(3), 1018. <https://doi.org/10.3390/app15031018>
- Derner, E., Batistič, K., Zahálka, J., & Babuška, R. (2024). A Security Risk Taxonomy for Prompt-Based Interaction With Large Language Models. *IEEE Access*, 12, 126176–126187. <https://doi.org/10.1109/ACCESS.2024.3450388>
- Feretzakis, G., & Verykios, V. S. (2024). Trustworthy AI: Securing Sensitive Data in Large Language Models. *AI*, 5(4), 2773–2800. <https://doi.org/10.3390/ai5040134>
- Jebbar, M. A. (2025). *Malicious Prompt Detection Dataset (MPDD)*.
- Jelodar, M. B. (2025). Generative AI, Large Language Models, and ChatGPT in Construction Education, Training, and Practice. *Buildings*, 15(6), 933. <https://doi.org/10.3390/buildings15060933>

- Kurniawan, A., & Chandra, M. B. (2025). Simulation of prompt injection attacks on generative pre-trained transformers models. *Procedia Computer Science*, 269, 400–410. <https://doi.org/10.1016/j.procs.2025.08.292>
- Kushnerov, O., Shevchuk, R., Yevseiev, S., & Karpiński, M. (2026). Comparative Benchmarking of Deep Learning Architectures for Detecting Adversarial Attacks on Large Language Models. *Information (Switzerland)*, 17(2). <https://doi.org/10.3390/info17020155>
- Lan, Q., Kaul, A., & Jones, S. (2025). Prompt Injection Detection in LLM Integrated Applications. *International Journal of Network Dynamics and Intelligence*, 4(2). <https://doi.org/10.53941/ijndi.2025.100013>
- Li, L., Li, J., Wang, H., & Nie, J. (2024). Application of the transformer model algorithm in chinese word sense disambiguation: a case study in chinese language. *Scientific Reports*, 14(1), 6320. <https://doi.org/10.1038/s41598-024-56976-5>
- Lubis, A. R., Lase, Y. Y., Rahman, D. A., & Witasryah, D. (2023). Improving Spell Checker Performance for Bahasa Indonesia Using Text Preprocessing Techniques with Deep Learning Models. *Ingénierie Des Systèmes d'Information*, 28(5), 1335–1342. <https://doi.org/10.18280/isi.280522>
- Mirshekali, H., Reza Shadi, M., Ghanadi Ladani, F., & Reza Shaker, H. (2025). A Review of Large Language Models for Energy Systems: Applications, Challenges, and Future Prospects. *IEEE Access*, 13, 163162–163188. <https://doi.org/10.1109/ACCESS.2025.3610994>
- Mutinda, J., Mwangi, W., & Okeyo, G. (2023). Sentiment Analysis of Text Reviews Using Lexicon-Enhanced Bert Embedding (LeBERT) Model with Convolutional Neural Network. *Applied Sciences*, 13(3), 1445. <https://doi.org/10.3390/app13031445>
- Patil, R., Boit, S., Gudivada, V., & Nandigam, J. (2023). A Survey of Text Representation and Embedding Techniques in NLP. *IEEE Access*, 11, 36120–36146. <https://doi.org/10.1109/ACCESS.2023.3266377>
- Pavlatos, C., Makris, E., Fotis, G., Vita, V., & Mladenov, V. (2023). Enhancing Electrical Load Prediction Using a Bidirectional LSTM Neural Network. *Electronics (Switzerland)*, 12(22). <https://doi.org/10.3390/electronics12224652>
- Raza, M., Jahangir, Z., Riaz, M. B., Saeed, M. J., & Sattar, M. A. (2025). Industrial applications of large language models. *Scientific Reports*, 15(1), 13755. <https://doi.org/10.1038/s41598-025-98483-1>
- Riyanto, S., Sitanggang, I. S., Djatna, T., & Atikah, T. D. (2023). Comparative Analysis using Various Performance Metrics in Imbalanced Data for Multi-class Text Classification. *International Journal of Advanced Computer Science and Applications*, 14(6). <https://doi.org/10.14569/IJACSA.2023.01406116>
- Salehin, I., & Kang, D.-K. (2023). A Review on Dropout Regularization Approaches for Deep Neural Networks within the Scholarly Domain. *Electronics*, 12(14), 3106. <https://doi.org/10.3390/electronics12143106>
- Shaday, E. N., Engel, V. J. L., & Heryanto, H. (2024). Application of the Bidirectional Long Short-Term Memory Method with Comparison of Word2Vec, GloVe, and FastText for Emotion Classification in Song Lyrics. *Procedia Computer Science*, 245, 137–146. <https://doi.org/10.1016/j.procs.2024.10.237>
- Sujon, K. M., Hassan, R., Choi, K., & Samad, M. A. (2025). Accuracy, precision, recall, f1-score, or MCC? empirical evidence from advanced statistics, ML, and XAI for evaluating business predictive models. *Journal of Big Data*, 12(1), 268. <https://doi.org/10.1186/s40537-025-01313-4>
- Varoquaux, G., & Colliot, O. (2023). Evaluating Machine Learning Models and Their Diagnostic Value. In O. Colliot (Ed.), *Machine Learning for Brain Disorders* (pp. 601–630). Springer US. https://doi.org/10.1007/978-1-0716-3195-9_20
- Venkatesh, S., Sindhu, R., & Arunachalam, V. (2025). Hardware efficient approximate sigmoid activation function for classifying features around zero. *Integration*, 103, 102421. <https://doi.org/10.1016/j.vlsi.2025.102421>
- Vyas, K. P., Bhatt, M., Dudhatra, D., Gupta, R., & Sharma, A. (2026). Malicious Prompt Classifier with Leave-One-Out Deletion Approach for Prompt Sanitization. *Eighth International Conference on Futuristic Trends in Networks and Computing Technologies (FTNCT08)*, 3015–3024. www.sciencedirect.com
- Wicaksana, H. S., Kusumaningrum, R., & Gernowo, R. (2024). Determining community happiness index with transformers and attention-based deep learning. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 13(2), 1753. <https://doi.org/10.11591/ijai.v13.i2.pp1753-1761>
- Wicaksono, G. W., Al asqalani, S. F., Azhar, Y., Hidayah, N. P., & Andreawana, A. (2023). Automatic Summarization of Court Decision Documents over Narcotic Cases Using BERT. *JOIV: International Journal on Informatics Visualization*, 7(2), 416. <https://doi.org/10.30630/joiv.7.2.1811>