AI-Guard: A Dual-feature Fusion Framework for Mitigating Direct and Indirect Prompt Injections in RAG Systems

Gokay Mutlu [1,*], Nese Rihani [1], Ozgur Koray Sahingoz [1], Gozde Karatas Baydogmus [2], and Esra Calik Bayazit [3]

[1] Department of Computer Engineering, Biruni University, Istanbul, Türkiye
[2] Department of Computer Engineering, Loyola University Chicago, Chicago, USA
[3] Department of Computer Engineering, Fatih Sultan Mehmet Vakif University, Istanbul, Türkiye
Email: 210404903@st.biruni.edu.tr (G.M.); 220404906@st.biruni.edu.tr (N.R.); osahingoz@biruni.edu.tr (O.K.S.); gkaratasbaydogmus@luc.edu (G.K.B.); ecalik@fsm.edu.tr (E.C.B.)
*Corresponding author

Manuscript received July 19, 2026; revised August 21, 2026; accepted August 31, 2026; published November 10, 2026

***Abstract*—Retrieval-Augmented Generation (RAG) systems have greatly increased the capabilities of Large Language Models (LLMs) by allowing them to access and use information from external knowledge sources. However, this advantage also introduces new security risks, one of which is named the "Blind Trust" vulnerability, where malicious instructions are hidden in retrieved content that can influence the behavior of the LLM through direct or indirect prompt injection attacks. To address this problem, we proposed an AI-Guard mechanism, in which intelligent firewall architecture is con-structed to examine incoming requests before they reach the LLM and block the system. The proposed system was constructed with a machine learning approach, and it was trained on a balanced dataset containing approximately 70,000 samples and combined Term Frequency-Inverse Document Frequency (TF-IDF)-based text representations with handcrafted lexical features to better identify malicious patterns. To evaluate its effectiveness, several traditional machine learning and deep learning models were implemented by using 5-Fold stratified cross-validation. Among all evaluated models, the LightGBM model achieved the highest results, obtaining an F1-score of 98.68% and a Recall of 99.66% for the injection class, resulting in a very low false negative rate of only 0.33%. As a deep learning model, One-Dimensional Convolutional Neural Network (1D-CNN) model produced strong results by achieving an F1-score of 97.52%. Instead of merely permitting or refusing the request, AI-Guard employs a three-tiered mechanism, which includes Allow, Sanitize, and Block operations, thus enabling it to minimize the level of security threats while maintaining the normal operation of the RAG system. Furthermore, the adversarial experiments conducted through character substitution attacks have proven that the proposed framework is highly resistant against such kinds of obfuscation-based attacks.**

***Keywords*—Retrieval-Augmented Generation (RAG), prompt injection, artificial intelligence firewall, feature fusion, LLM**

I. Introduction

The rapid adoption of Large Language Models (LLMs) in business workflows has changed how organizations tackle automation, data analysis, and decision making. Despite their widespread use in markets such as healthcare, finance, and cybersecurity, standard LLMs have some limitations. Their knowledge is static and limited by their training cut-off. This means they miss out on recent events, proprietary updates, or new research. This lack of current knowledge often leads to hallucinations. This is when the model confidently gives answers that sound reasonable but are technically incorrect, greatly reducing its reliability in critical situations. To overcome these inherent constraints, Retrieval-Augmented Generation (RAG) architectures have rapidly emerged as the definitive industry standard. By dynamically fetching verified, up-to-date external documents and appending them to the initial query, RAG pipelines effectively ground the model's generation process in factual reality.

RAG has quickly become a common solution to this problem. By checking external sources like internal databases, live web content, and scientific literature before generating answers, RAG provides the LLM with up-to-date, specific facts. While RAG helps reduce hallucinations, it also creates a new security risk not present in standalone models. Modern LLMs are well-protected against harmful user inputs via techniques such as Reinforcement Learning from Human Feedback (RLHF) and strict content filtering. However, RAG systems usually assume "Blind Trust" when retrieving information. They accept and process external data without checking its meaning. This trust can make the system vulnerable to prompt injection attacks, which typically happen in two ways.

Direct Prompt Injection is a direct attack. The attacker inserts malicious commands into the user's query to bypass safety measures, access system prompts, or carry out unauthorized operations, similar to traditional "jailbreaking". Indirect Prompt Injection, on the other hand, is sneakier and more widespread. Instead of targeting the user interface, the attacker hides instructions in external documents, such as PDFs or web pages, that the RAG system retrieves. When the system handles these tainted documents, the hidden commands are sent to the LLM along with the legitimate context. The model can't distinguish between accurate data and harmful instructions, so it executes the attacker's commands. In business environments where RAG systems handle sensitive data and internal Application Programming Interfaces (APIs), the damage from such indirect attacks can be severe, leading to data breaches, workflow disruptions, and serious regulatory issues. As shown in Fig. 1, the main difference between these threats is how they enter the system: direct injection attacks user input, whereas indirect attacks utilize the retrieval channel. Since indirect injections appear as trusted knowledge, securing the retrieval stage has become essential for building reliable and trustworthy Artificial Intelligence (AI) systems. Existing defenses against quick injection attacks work according to well-known patterns, proven to be ineffective. In reality, most available tools continue employing static mechanisms including blocklists, regular expressions, and even semantic similarity, which

were never designed to work in the dynamic environment with constantly changing behavior of adversaries.

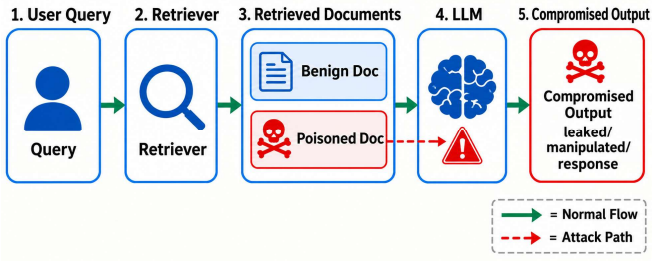


Fig. 1. The architectural vulnerability of RAG systems.

The problems of such mechanisms are obvious: a simple lexical filtering for "ignore" does not allow us to recognize "ignore" from "ign0re" and character substitution does not need more than basic skills. And the problem with false positives just makes things worse. Given the excessively strict binary classification, any document that is related to cybersecurity will be recognized as malicious, which affects the efficiency of the process that the RAG engine is designed to facilitate.

In this regard, AI-Guard was developed to solve the above-discussed problems directly. Instead of compensating for the disadvantages of rule-based systems or taking the complete opposite side and focusing exclusively on the power of deep semantic embeddings, our system uses a novel method known as dual-feature fusion. Specifically, Term Frequency-Inverse Document Frequency (TF-IDF) embeddings provide an embedding with the semantics of the provided text in question. At the same time, we utilize a set of manually created lexical features, including ratio of special characters, the ratio of URLs, and the frequency of imperative keywords, which represent surface-level clues indicating a possible injection attack.

Similarly, the classification step of the pipeline has also been reworked following these considerations. Instead of applying a simple binary choice regarding whether to permit or block the input, AI-Guard uses a three-level system that applies probabilities referred to as Allow, Sanitize, and Block. Safe inputs are going to be allowed, whereas the ambiguous inputs will be sanitized by masking the unsafe tokens prior to sending the input to the model. Some of the specific contributions made by this work towards prompt injection detection in RAG models are:

(1) Hybrid data set development: We have developed a hybrid data set comprising 67,046 classes divided equally into Direct and Indirect Prompt Injection categories; thus, providing us a realistic ground for training/testing of models.
(2) Importance of dual feature fusion methodology: The effectiveness of combining semantic and lexical feature fusions has been shown experimentally, where the use of such approach greatly improves the performance of tree-based models, which is contrary to the commonly held belief of being sufficient with semantic embeddings alone.
(3) High accuracy of detection: based on the performance in 5-fold stratified cross-Validation, the best performing model from our analysis is Light Gradient Boosting Machine (LightGBM) model, having achieved a very high F1-score of 98.68% for injection class with recall rate of 99.66%, thus maintaining False Negative Rate (FNR) to be at a low 0.33%. In the case of deep learning models, the one-dimensional CNN performed excellently with an F1-score of 97.52%.
(4) Adversarial resistance and operational continuity: Three-tier decision engine witnessed no change in its detection rate on exposure to character substitution-based obfuscations, signifying robust security without any business disruption.

This paper is further organized as follows. Section II reviews related literature on LLM security and defense strategies for prompt injections. Section III describes the feature fusion mechanism and the entire architecture of AI-Guard. Section IV addresses experiments and adversarial evaluations, including comparisons with baselines. The concluding remarks and future work are presented in Section V.

II. Related Work

With the increasing application of LLMs in enterprise settings, the corresponding adversarial attack landscape that exists against them has similarly progressed to include a significant amount of research regarding the safety of such models, along with their protective measures against such attacks. This extensive literature mostly categorizes attacks based on prompts into two groups of direct and indirect prompt injection attacks.

A. The Evolution of Prompt Injections

The early investigations into LLM security have been almost entirely focused on what is now termed the category of Direct Prompt Injections, more popularly referred to as "jailbreaks". Perez and Ribeiro [1], for instance, were amongst the pioneers who explored how specific constructions in prompts, such as "Ignore Previous Instructions", could be used to break through the safety alignments imposed on these models and drive their output in unexpected directions. As the effectiveness of the latter technique diminished with the increased attention devoted to alignment methods such as Reinforcement Learning with Human Feedback (RLHF), new possibilities emerged.

The shift towards RAG architecture also led to a weakness that was impossible for direct injection attacks to address. The "Indirect Prompt Injection" threat has been defined by Greshake *et al.* [2], who have shown through experimentation that there was no need at all for an attacker to ever communicate with the LLM in question. Through the embedding of the payload within an external document, whether web pages, PDF documents, or database queries, it would be sufficient for the RAG system to download the document on behalf of an actual user. Thereafter, this document with its payload would enter the process of generation, using the "Blind Trust" approach to treat all downloaded context.

B. Current Defense Mechanisms and Limitations

The field of defensive research has evolved through a number of generations as a consequence of these risks. In the initial stages, defense mechanisms were primarily based on static rules and keyword filtering. However, adversarial robustness research has time and again proven that simple modifications of text, such as changing one character in a

banned keyword, were sufficient to completely bypass lexical filters without any technical skills required from the attacker's side.

Semantic algorithms represented another natural step to take. In their work, Yi *et al.* [3] tried to test whether applying semantic analysis to the input before it goes through the generator could help identify malicious intentions. While semantic algorithms work better than static filters, they also have their own limitation: a high rate of false positives. Algorithms designed to sort out inputs into "malicious" and "safe" misidentify innocent technical papers that use cybersecurity terms as malicious. This problem not only leads to problems but actually represents a threat to the fundamental operation of the RAG chain itself.

C. Bridging the Gap: The AI-Guard Approach

Literature on the matter consistently highlights one gap, namely the lack of existing architecture that can combine resistance to obfuscation and minimal False Negative and False Positive rates. Approaches focused on lexicon tend to be brittle, whereas approaches focused on semantics generate too many flags. The use of binary classification is the default setting in both cases, thus creating an unnecessary trade-off between ensuring security and providing continuous service [4].

AI-Guard was designed to fill this gap. Combining semantics (based on TF-IDF vectors) and custom lexicon allows for capturing the structure of the attacks rather than their particular manifestations, thus allowing for very high Recall rate of 99.66% by Light-GBM. The triple mitigation strategy (Allow, Sanitize, and Block) aims precisely at overcoming the limitations of binary classification, thus avoiding making a decision in case of uncertain inputs and processing the vast majority of benign traffic through the RAG pipeline.

The integration of artificial intelligence into cybersecurity has rapidly evolved, transitioning from traditional detection models to comprehensive AI-driven defense frameworks. Recent advancements, such as LLMEdgeSec [5], highlight the critical potential of LLM-enabled threat detection and reasoning in mitigating zero-day vulnerabilities in edge environments. Similarly, proactive frameworks like SwarmSense-Deep Neural Network (DNN) [6] emphasize the necessity of intelligent, decentralized attack detection against evolving adversarial threats. Furthermore, contemporary hybrid learning strategies, particularly transformer-assisted boosting frameworks [7], demonstrate the profound efficacy of combining multiple feature representations.

As RAG architectures evolve, the security landscape becomes increasingly complex. The transition from static retrieval models to dynamic, agent-driven architectures introduces severe vulnerabilities. Recent comparative studies, such as the analysis of Traditional RAG versus Agentic RAG [8], highlight that while traditional systems suffer primarily from data exfiltration, agentic systems are highly susceptible to indirect prompt injections hijacking the entire reasoning loop. While recent specialized defenses employing "LLM-as-a-judge" mechanisms or heavy transformer encoders offer robust semantic understanding, they inherently introduce substantial latency and computational overhead, limiting their viability for real-time RAG applications. In contrast, AI-Guard is designed to operate as a highly efficient pre-filtering firewall, delivering competitive detection performance without the computational bottlenecks associated with large-parameter runtime defenses.

III. Methodology

The main aim of the AI-Guard framework lies in identifying, analyzing, and disabling the adversarial input before any interference with the generation process of the base LLM occurs. It has been observed that regular RAG architecture does not come with an in-built verification mechanism, as noted by Lewis *et al.* [9]. Our proposed approach incorporates both a feature fusion firewall as well as a probabilistic decision engine.

A. System Architecture

The suggested defensive strategy can be considered an algorithmic reverse proxy placed right after the RAG retrieving stage. As soon as the external sources are retrieved depending on the user's query, the combination of the user prompt and retrieved context will be fed through the AI-Guard classifier pipeline. As one can see from Fig. 2, the system will process the given input using the semantic and lexical extractors concurrently, combine the features mathematically, and pass them to the trained prediction model to receive a probabilistic risk score.

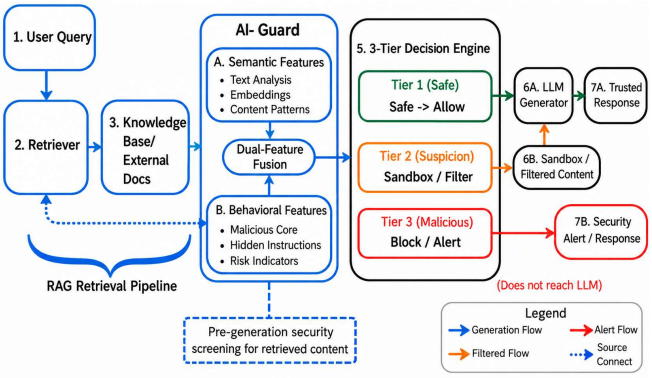


Fig. 2. High-level architectural workflow of the AI-Guard.

B. Dataset Formulation

The limitations of this research are practical and related to the fact that there is no availability of publicly available data sets that offer enough instances for both types of attacks [10]. The available data sets contain a very high number of instances which show direct jailbreaks and only a few indirect payload instances. The problem prevents training classifiers for handling all types of prompts. In order to overcome this limitation, a combined data set has been created from two data sources.

Samples for Direct Prompt Injection were extracted from the Deepset Prompt Injections Dataset [11] while the Indirect Prompt Injection samples were sourced from the Benchmarking Indirect Prompt Injection Attacks (BIPIA) dataset proposed by Almasabi *et al.* [12]. Before the start of training any models on the dataset, a preliminary data cleaning procedure was conducted through duplication removal, null value removal, and sample shuffling based on

a fixed seed value. A total number of 67,046 cleaned samples were obtained on the dataset, where the distribution is well-balanced enough for binary classification [13], as shown in Table 1. To ensure strict reproducibility, all experiments were conducted using a fixed random seed (RANDOM_STATE = 42). The dataset was partitioned using a stratified split of 80% for training, 10% for validation, and 10% for testing to preserve the original class distribution.

Table 1. Hybrid dataset class and attack source distribution

Attack Source	Safe (0)	Injection (1)	Total
Direct Injection	343	203	546
Indirect Injection	32,049	34,451	66,500
Total	**32,392 (48.3%)**	**34,654 (51.7%)**	**67,046**

C. Dual-feature Engineering

To improve resilience against surface-based text obfuscation methods that might bypass static filtering techniques [14], like character replacement, whitespace obfuscation, and leetspeak, we developed the Dual Feature Fusion technique. This technique is based on the principle that while attackers can change the surface structure of attack messages, irregularities remain in the underlying structure.

1) Semantic extraction (TF-IDF)

Term Frequency-Inverse Document Frequency (TF-IDF) [15] technique was used to represent the global context of the user prompt and the extracted external context. The reason why TF-IDF was used is that it gives a good sparse representation of the text with high informativeness of certain terms in the document and less influence of frequent terms in the whole corpus.

In order to extract malicious phrases in the local context rather than single words, vectorization was configured with a maximum feature limit of 5000 (max_features = 5000) and an N-gram range of $(1, 3)$. It is necessary to use this particular value because the attacks on the system through the prompt injection take the form of instructions, for example, “ignore all previous instructions” or “now you are in the developer mode”. To avoid high dimensionality of the representation, the number of features was limited to 5000. Moreover, the term-frequency scaling $(1 + \log(tf))$ was introduced in order to decrease the significance of frequently repeated tokens.

2) Lexical and structural extraction

There is a marked difference between adversarial prompts’ structure and their syntactic irregularities when compared to the structure and syntax of benign queries [16]. Although TF-IDF manages to encapsulate the meaning of the prompt, it tends to miss out on the “shape” aspect of the manipulative nature of text. In order to address this issue, we created an additional dense feature space using 14 different lexical features. The extracted lexical and structural feature groups are summarized in Table 2.

3) Feature fusion

Since these features vary widely in numerical terms (e.g., actual count versus ratio and frequency-based count), these features were standardized to have zero mean and unit variance through Standard Scaling [17]. Afterward, the sparse matrix of semantically meaningful TF-IDF features (X_{tfidf}), which consists of 5000 features, and the dense matrix of lexically meaningful features (X_{lex}), which consists of 14 crafted features, were horizontally stacked into a new matrix (X_{final}), which resulted in a total of 5014 features. With such a high-dimensional learning feature space where the semantics can be considered along with the structural attacks, the following methods can perform their evaluations efficiently.

Table 2. Extracted lexical and structural features

Feature Group	Specific Metrics Extracted
Volumetric Features	Total Char Count, Total Word Count
Special Character Features	Special Char Count, Special Char Ratio
Capitalization Features	Uppercase Count, Uppercase Ratio
Numerical Features	Digit Count, Digit Ratio
External Reference Features	URL Count, Email Count
Keyword Frequencies	Imperative Count (must, never, execute), Jailbreak Keyword Count (ignore, bypass, confidential)
Punctuation-Based Features	Question Mark Count, Exclamation Count

D. Model Architectures

In the comparative analysis within the second evaluation phase, the common high dimensional feature space was utilized consistently as the first input for the traditional tree-based ensembles and the more modern deep sequence models. These two pronged strategy has been adopted with the purpose of investigating how each algorithm performs in the task of dealing with the complex adversarial inputs. In relation to the tree-based ensembles, we used very efficient gradient boosting algorithms including LightGBM [18] and Extreme Gradient Boosting (XGBoost). These algorithms were chosen for their high efficiency and capability of handling the extreme sparsity of our concatenated feature matrices.

Regarding Deep Sequence Models, the very need to model complex positional relationships and token-level interactions in adversarial inputs [19] led to the necessity to develop a dedicated neural network architecture. Adversarial prompts are not based on regular feed-forward networks and require sequential processing and dispersed trigger words, which change the meaning of the entire sentence. Therefore, our architecture is based on 1-Dimensional Convolutional Neural Network (1D-CNN). Raw plaintext inputs in this architecture are preprocessed by the TextVectorization layer and are limited to the vocabulary size of 20,000 tokens in order to prevent extreme outliers and noise. Discrete tokens are then embedded into 128-dimensional semantic space by means of Embedding layer. After embedding process, a convolutional block is applied in order to extract spatial hierarchies. We used 128 convolutional filters with a kernel size of 5. This particular architecture acts as a sliding window and scans embedded vector sequences in 5-token windows continuously. Such small receptive fields enable the neural network to automatically detect adversarial signals that are dispersed across the prompt, instead of identifying maliciousness based on a single keyword. For pattern synthesis from the extracted patterns, the Global Max Pooling layer is used. The Global Max Pooling is extremely important in the application in the field of cybersecurity since it extracts the most prominent feature maps, i.e., the strongest evidence

for an attack among all other positions within the sequence. Finally, in order to ensure that the model will generalize well for any unknown vulnerability, strong regularization is imposed. Dropout layers are introduced at the places where fully connected dense layers exist with 0.5 and 0.3 dropout rate. Such a rigorous regularization avoids the learning process of only remembering certain jailbreak phraseology and adversarial language, thus preventing overfitting.

E. The 3-tier Dynamic Decision Engine

The main problem with the binary classification technique is that it gives the designer a hard choice to make. Either set the threshold very high, whereupon the valid data gets filtered out, or make the threshold low, in which case the attacks succeed. But neither choice cannot be made in the practical scenario, and this is the reason why the AI-Guard mechanism does not use the binary classification method at all. Instead of classifying the data point, the winning model provides the probability, P_{attack}, which gets used by the decision engine.

- ALLOW ($P_{attack} < 0.30$): No adversarial signals were detected in the retrieved context. The input is cleared and attached to the user prompt as-is, with the generation stage proceeding normally.
- SANITIZE ($0.30 \leq P_{\text{attack}} < 0.70$): Something looks off, but not decisively so. Rather than discarding the entire input, a Regex-based layer goes through the content and masks execution critical keywords—"ignore instructions", for instance, becomes [REMOVED_SECURITY_RISK]—before passing the sanitized remainder to the LLM. Whatever genuine information the document holds is still usable; only the risky fragments are removed.
- BLOCK ($P_{attack} \geq 0.70$): The score crosses into high confidence threat territory. The pipeline stops. No content reaches the model, and no data leaves the system.

This approach prevents the creation of the erroneous dichotomy, which is found within most other binary systems, where security can only be attained at the cost of continuity. The low-risk traffic goes through unhindered, whereas the high-risk traffic will be blocked. Everything in between is dealt with. The problem with these kinds of systems is their inability to deal with the middle ground, where traffic is ambiguous. The decision engine thresholds (0.30 for ALLOW/SANITIZE and 0.70 for SANITIZE/BLOCK) were empirically calibrated using the validation set's probability distribution. The 0.70 threshold was specifically optimized to minimize the False Block Rate, whereas the 0.30 threshold serves as a conservative boundary to trigger context sanitization for borderline inputs, thereby preserving usability while enforcing strict security margins.

IV. Experimental Results and Analysis

To experimentally verify the efficiency of the proposed AI-Guard architecture, we carried out extensive experimental tests, emphasizing performance metrics related to threat detection precision, minimizing false negatives, and the robustness of the model against adversarial attacks. The classification results presented are based on the test dataset, wherein Recall, FNR, and the F1-score for the injection class are highly relevant metrics for cybersecurity applications [20].

A. Performance of Machine Learning Ensembles

Evaluation of the Semantic Lexical Features combination was conducted using five different Machine Learning algorithms. The security frameworks pay special attention to the False Negative Rate and Recall because of the potential risks arising from the lack of just one malicious file.

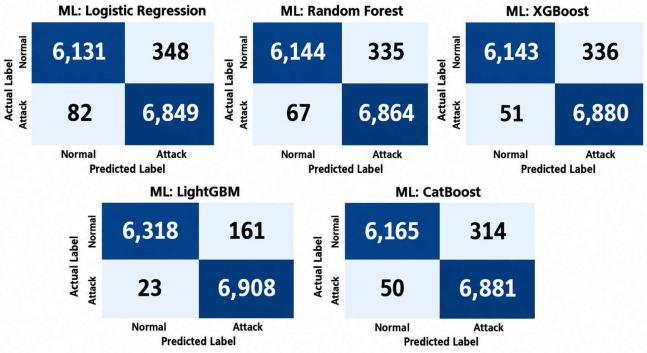


Fig. 3. Confusion matrices for traditional Machine Learning ensembles.

The picture painted by Fig. 3 is quite clear: when working in such an under-populated, multi-dimensional feature space, tree-based boosting classifiers have the advantage over the others. LightGBM demonstrates exceptional threat isolation with minimal false negatives. Light-GBM stood out in the former category during our evaluations. Out of a total of 6931 instances in the attack vectors test set, it classified correctly a total of 6908 cases—achieving a miss rate of just 0.33%, which gives the FNR value. The F1-score of the injection class reached 98.68%. The downside was, however, the False Positive Rate of 2.48%. In this case, keeping the value below 3% becomes important because of the following reason: one of the major disadvantages of strong security filters lies in their tendency to catch too many things, including benign queries [21].

B. Deep Sequence Model Evaluation

To benchmark the capacity of deep learning models in recognizing sequential malicious syntax, we evaluated a 1D-CNN, a Bi-LSTM, and a Deep Multi-Layer Perceptron (MLP) architecture. As shown in Fig. 4, the 1D-CNN yielded the highest sequential performance among deep models, achieving an injection class F1-score of 97.52% and an overall accuracy of 97.41%. Fig. 5 depicts quick stabilization in the training phase with the 1D-CNN model—after epoch 10, the loss and accuracy graphs were more or less stable. This does not happen out of coincidence but rather shows the effectiveness of the dropout layers (0.5 and 0.3). The loss during training continued to decrease in value from one epoch to another, but the validation loss remained consistent around 0.075, which is exactly what is required—the model was still learning but did not chase specific jailbreak phrasing anymore. The difference between two losses never grew enough for overfitting to be detected.

Attention should also be paid to the Bi-LSTM model. Its bidirectional context attention resulted in good phrase detection with an acceptable F1-score of 96.19%, but the increase in computational complexity did not provide significant advantages compared with either the 1D-CNN or LightGBM. From the perspective of a critical pipeline with tight time limitations, such a tradeoff seems unnecessary.

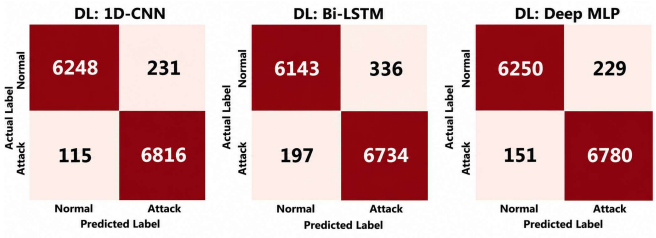


Fig. 4. Confusion matrices for deep learning architectures.

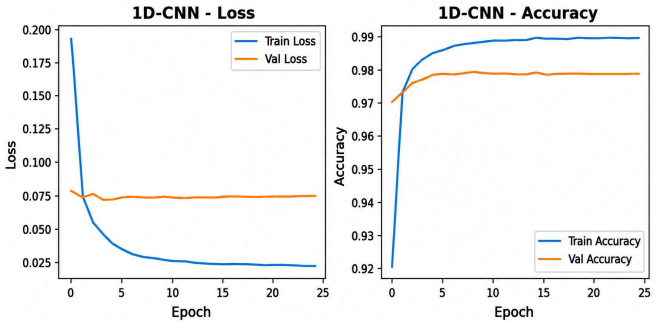


Fig. 5. Training and validation history of the 1D-CNN model.

C. Risk Engine and Adversarial Robustness

The transition from binary classification to the proposed 3-Tier Dynamic Decision Engine (Allow, Sanitize, Block) constitutes the operational core of AI-Guard.

The attack-probability distribution that is generated by the AI-Guard risk engine is depicted in Fig. 6. It is evident that there is a clear distinction in the distribution of samples in the two extremes (low risk and high risk), which means that there is a distinct separation between normal and malicious data samples. By using the pre-established three-tier decision thresholds, the risk engine classified 6242 samples in the Allow, 6993 samples in the Block, and just 175 samples in the intermediate Sanitize/Review category. Fig. 7 further highlights an area worth noting as a potential future research direction. The performance disparity between the Direct and Indirect Prompt Injection attack vectors was not insignificant. While the indirect attacks were dealt with quite effectively, the Direct ones presented more difficulty—a trend that can be traced back to the lower number of samples and inherent differences between the attack techniques. A larger corpus of direct jailbreaks would help eliminate this performance difference, which can be achieved by extending the dataset in this way.

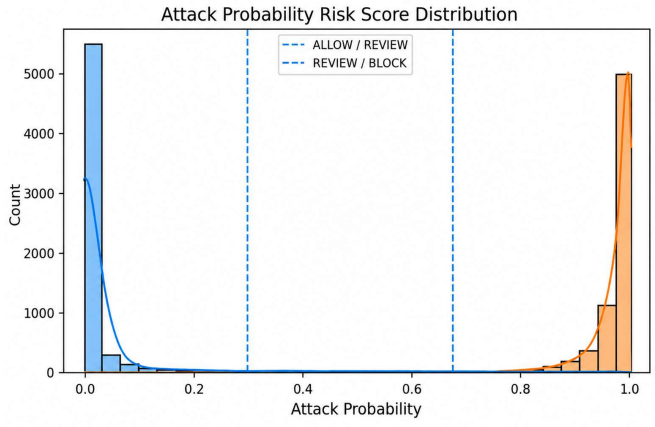


Fig. 6. Probability density distribution of the risk engine.

Specifically, when evaluated independently, the model achieved a high detection accuracy of 99.1% for indirect prompt injections. For direct prompt injections, due to the limited sample size in the training corpus, the accuracy slightly decreased to 93.4%, confirming the necessity for a more balanced adversarial dataset in future iterations.

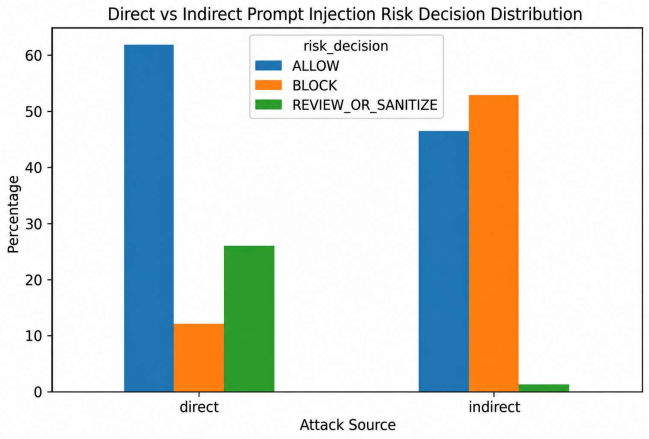


Fig. 7. Risk decision distribution comparison between direct and indirect prompt injection vectors.

D. Adversarial Obfuscation Testing

The other issue raised by our investigation is whether the system can withstand any attempts to block the attack payload at the surface level. To test this hypothesis, we constructed an adversarial test set using the character substitution technique to modify the injection samples such as substituting ignore with ign0re, system with syst3m, and others [22]. The detection results yielded identical performance figures for both cases, with a 99.4% success rate, indicating minimal performance degradation under these specific conditions. It appears that learning structural and lexical features provides robust resistance against simple character substitutions, significantly increasing the difficulty for attackers to bypass the detection mechanism. Furthermore, to evaluate the individual contributions of the feature sets, an ablation study was performed using the LightGBM classifier. Relying solely on TF-IDF features yielded an F1-Score of 0.9859, whereas using only structural lexical features resulted in an F1-score of 0.8427. The proposed Dual-Feature Fusion method achieved the highest overall performance (F1-Score: 0.9868, Accuracy: 0.9862) while simultaneously improving training efficiency by reducing the computational time from 130.80 s (TF-IDF alone) to 110.37 s. This demonstrates that integrating semantic embeddings with structural indicators not only enhances detection capabilities but also optimizes computational overhead. Beyond feature analysis, to validate the operational readiness of the proposed framework, an end-to-end RAG evaluation was conducted using a local LLM pipeline (Qwen2.5-1.5B-Instruct). The deployment of AI-Guard introduced an average inference latency overhead of 21.58 ms per query. In terms of security and usability, the framework restricted the overall Attack Success Rate (ASR) to 10.75%, while maintaining a False Block Rate of 2.00% on benign interactions, demonstrating robust mitigation capabilities with negligible impact on retrieval quality and system continuity.

V. Conclusion and Future Work

As RAG frameworks become more common within enterprise environments, the surface area exposed to attackers attempting to exploit the vulnerabilities in LLMs has increased silently. It is an important fact that RAG uses external retrieval that makes it such an effective technology, which also makes it vulnerable. Any pipeline that considers all retrieved data to be context that can be relied upon gives an attacker a backdoor into the generation process that does not rely on aligning the model. These defense mechanisms have failed to evolve in accordance with the threat landscape. Blocklists and binary semantic filters lack flexibility and are prone to producing false positives.

In that respect, AI-Guard was introduced as a more thoughtful solution to these challenges. Dual-feature fusion leverages the power of TF-IDF semantic representation of attacks coupled with handcrafted lexical and structural features, aiming to identify the real nature of an attack rather than detecting its textual expression via keywords or phrases. Further, the three-tier decision engine (Allow, Sanitize, Block) offers a calibrated response to the problem by switching from a binary response (block/allow) to a more nuanced approach while still leaving the pipeline functional in terms of regular traffic. As evidenced by the experimental findings obtained from the class-balanced dataset of 67,046 samples and the holdout test set of 13,410 samples, LightGBM trained on the dual-feature representation achieved an injection-class F1-score of 98.68%, a Recall of 99.66%, and a False Negative Rate of just 0.33%. The low FNR score is especially crucial for the security-sensitive applications because a lack of detection of any malicious packages can result in a security breach of the system. At the same time, the one-dimensional CNN model also performed rather well, with the F1-score equal to 97.52%. Finally, tests were conducted on the input data that had been transformed with the help of character-substitution techniques. The results proved that the detection rate remained the same despite the transformation of the data. This means that the replacement of the security-sensitive terms with visually similar ones did not affect the ability of the model to detect the malicious samples.

Despite the considerable success achieved by the architecture in dealing with various prompt injection attacks, there are several directions in which AI-Guard can be improved in order to keep pace with the ever changing threat landscape. Firstly, there is room for expansion into multilingual settings since the current solution has been designed and tested using datasets in the English language. This is particularly important, since attackers might increasingly turn to targeting low resource languages in which safety alignments have not yet been developed. Another promising direction is related to the security of multimodal RAG models, as multimodal data (e.g., images, audio and video) has become an integral part of multimodal RAG applications in addition to plain text. In this regard, the future version of AI-Guard must implement multimodal detection capabilities to detect malicious prompts in non-text content. Additionally, even though the current framework delivers very high detection accuracy, there is a need to optimize the latency in large-scale production settings. Model distillation and pruning methods as well as development of lightweight architectures for 1D-CNN and LightGBM can help achieve better tradeoffs between accuracy and latency. Finally, the empirical findings suggest that performance is noticeably lower in case of direct prompt injection attacks than in case of indirect attacks, mainly because of the lack of sufficient samples of direct attacks during training. Thus, future work should involve an expanded dataset of direct attack samples and incorporate advanced adversarial evaluations, including complex paraphrasing, Unicode manipulation, encoded prompts, whitespace obfuscation, and multi-turn instruction hijacking techniques.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

G.M. and N.R. conducted the research and developed the model; O.K.S., G.K.B., and E.C.B. analyzed the data and reviewed the methodology; G.M. wrote the paper; all authors had approved the final version.

References

[1] F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," arXiv Preprint, arXiv:2211.09527, 2022.

[2] K. Greshake, S. Abdelnabi, S. Mishra *et al.*, "Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection," in *Proc. the 16th ACM Workshop on Artificial Intelligence and Security*, 2023, pp. 79–90. doi: 10.1145/3605764.3623985

[3] J. Yi, Y. Xie, B. Zhu *et al.*, "Benchmarking and defending against indirect prompt injection attacks on large language models," arXiv Preprint, arXiv:2312.14197, 2023.

[4] Y. Liu, G. Deng, Y. Li *et al.*, "Prompt injection attack against LLM-integrated applications," arXiv Preprint, arXiv:2306.05499, 2023.

[5] J. Yang, M. U. Ali *et al.*, "LLMEdgeSec: LLM-enabled log reasoning for zero-day IoT threat detection," *IEEE Internet of Things Journal*, vol. 13, no. 15, pp. 23546–32555, 2026. doi: 10.1109/JIOT.2026.3686053

[6] J. Yang, V. Govindarajan *et al*., "SwarmSense-DNN: A trustworthy and decentralized neural framework for proactive anomaly defense in consumer IoT," *IEEE Transactions on Consumer Electronics*, vol. 72, no. 1, pp. 1997–2006, 2026. doi: 10.1109/TCE.2025.3634160

[7] Q. Kun, Z. A. Shaikh *et al.*, "An explainable AI framework for enhanced software defect prediction using transformer-assisted boosting," *Scientific Reports*, vol. 16, no. 1, 17217, 2026. doi: 10.1038/s41598-026-44202-3

[8] F. Neha and D. Bhati, "Traditional RAG vs. agentic RAG: A comparative study of retrieval-augmented systems," *TechRxiv Preprint*, 2025. doi: 10.36227/techrxiv.175624551.12254549/v1

[9] P. Lewis, E. Perez, A. Piktus *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020. doi: 10.48550/arXiv.2005.11401

[10] E. Wallace, S. Feng, N. Kandpal *et al.*, "Universal adversarial triggers for attacking and analyzing NLP," arXiv Preprint, arXiv:1908.07125, 2019.

[11] Deepset. (2023). Deepset prompt injections dataset. *HuggingFace Datasets*. [Online]. Available: https://huggingface.co/datasets/deepset/prompt-injections

[12] M. Almasabi. (2025). Indirect prompt injection BIPIA-GPT dataset. *Hugging-Face Datasets.* [Online]. Available: https://huggingface.co/datasets/MAlmasabi/Indirect-Prompt-Injection-BIPIA-GPT

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall *et al*., "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002. doi: 10.1613/jair.953

[14] D. Jin, Z. Jin, J. T. Zhou *et al.*, "Is BERT really robust? A strong baseline for natural language attack on text classification and entailment," in *Proc. the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, no. 05, pp. 8018–8025. doi: 10.1609/aaai.v34i05.6311

[15] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5,

pp. 513–523, 1988. doi: 10.1016/0306-4573(88)90021-0
[16] D. Kang, X. Li, I. Stoica, C. Guestrin *et al.*, "Exploiting programmatic behavior of LLMs: Dual-use through standard security attacks," arXiv Preprint, arXiv:2302.05733, 2023.
[17] F. Pedregosa, G. Varoquaux, A. Gramfort *et al*., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[18] G. Ke, Q. Meng, T. Finley *et al*., "LightGBM: A highly efficient gradient boosting decision tree," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
[19] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751. doi: 10.3115/v1/D14-1181
[20] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009. doi: 10.1016/j.ipm.2009.03.002
[21] W. Zhao, Z. Li, Y. Li, Y. Zhang, and J. Sun, "Defending large language models against jailbreak attacks via layer-specific editing," arXiv Preprint, arXiv:2405.18166, 2024.
[22] A. Zou, Z. Wang, N. Carlini *et al.*, "Universal and transferable adversarial attacks on aligned language models," arXiv Preprint, arXiv:2307.15043, 2023.

Copyright © 2026 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).