JCBI
Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606
Volume 12 Issue 01
2026

Research Article
https://doi.org/10.56979/1201/2026/1552

A Deep Learning Framework for Evaluating and Mitigating Prompt Injection in LLM Agents

Najda Malik[1], Muhammad Hassan Ghulam Muhammad[2], Tahir Alyas[2], Wajahat Mahmood Qazi[3], Nadia Tabassum[4*], and Adeela Hayat[2]

[1]Department of Computer Science, Institute of Management Sciences, Lahore 54000, Pakistan.
[2]Department of Computer Science, Lahore Garrison University Pakistan.
[3]Superior University, Lahore, Pakistan.
[4]Department of Computer Science, Virtual University of Pakistan.
[*]Corresponding Author: Nadia Tabassum. Email: nadiatabassum@vu.edu.pk

Received: June 13, 2026 **Accepted:** September 06, 2026

Abstract: In the world of large language models (LLMs), prompt injection attacks are a severe security risk that can circumvent the instructions given to AI agents and potentially cause damage. In agentic architectures, the vulnerabilities are magnified because of the increased attack surfaces which are present in the use of tools, APIs and chains of reasoning. The vulnerabilities are compounded in agentic architectures as the tools used, APIs called, and chains of reasoning become a larger attack surface, resulting in an increased sense of unreliability. Paraphrased, adaptive, and context-aware attacks are not thwarted by rule-based and static prompt engineering defenses. This paper introduces a deep learning-based approach to detecting and thwarting prompt injection attacks in LLM agent systems. Uses a hybrid model of known and customer-generated adversarial prompts to combine transformer-based classification, semantic risk-scoring and adaptive mitigation procedures. Stratified data splits are used in the experiments, as are synthetic attack generation and rule-based, BERT and zero-shot LLM classifiers. In terms of precision, recall, F1 score and AUC, the proposed framework is found to be better than all the baselines. Attack of instruction override, data extraction and tool misuse are greatly enhanced and resistant to adversarial scenarios that are not seen and adaptable. The mitigation module achieves high task utility; low false positive rates and the high success rate of the attacks is significantly reduced. Overall, the framework demonstrates that it is possible to integrate detection, risk estimation, and mitigation in a single pipeline, resulting in a great improvement in the security of LLM agents, while not burdening them with significant computing costs. It further highlights the importance of security layers before deploying LLM agents to the real world as the input they receive can be adversarial and continuously change, requiring a mixture of machine learning and structured control to ensure they remain safe and reliable across different application areas. Scalable AI systems deployment safety improvements involve multi-agent security, real-time learning, and multimodal prompt injection detection in the future.

Keywords: Prompt Injection; Large Language Models; Agent Security; Deep Learning

1. Introduction

The model can generate unexpected behavior and ignore system instructions if the model is tricked into following malicious prompt injection (injecting malicious prompts into the user's input or third party data) [1]. This is because LLMs are not good at filtering out trusted content from suspicious content in the context window [2]. Prompt injection is a major concern if the LLM is used as an agent, as it can affect the way the tools are used, can influence the state of the reasoning chain, or can have unintended consequences due to tool calls or APIs. This adds another dimension to the attack surface, from text manipulation to system-level exploits with much greater impact in actual deployments [4]. Today's lines of defense primarily rely

on rule-based filtering, prompt engineering, and static safety constraints, but they are not particularly robust against paraphrased, contextually disguised, and adaptive attacks [5,6]. Furthermore, with the structural approach, which includes sandboxing and access control, there is potential for less system damage, but not for semantic ambiguity in language interpretation.

Transformer models have demonstrated their ability to understand context, but their reliability in an adversarial setting remains unknown, as they tend to be sensitive to well-designed inputs [7,8]. Furthermore, the present study does not consider prompt classification, system-level protection, and the relationship between prompt detection, risk assessment, and mitigation in the context of LLM agents. The concept of addressing dynamic, evolving attack strategies in real-world environments that lack a holistic design is limited [9]. To tackle these problems, this study introduces a transformer-based solution to detect and remove prompts from LLM-based agent systems during the pre-execution stage, where the input may influence the LLM's output. It is a collection of structured detection and mitigation actions, including semantic detection, risk scoring, and adaptive mitigation, that constitute a defense process against adversarial inputs [10]. The main objective of this research is to evaluate the accuracy of deep learning models in distinguishing malicious prompt injection requests from legitimate requests, and to understand the factors that make it difficult to accurately detect prompt injection attacks in agent-based LLM applications [11,12]. Moreover, this research examines the impact of the proposed framework on the system's security and usefulness in actual implementation. The primary results show that the detection mechanism based on transformer models with a structured risk assessment is highly robust to a wide range of prompt injection attacks, while still performing moderately well computationally [13]. Overall, the study underscores the need to embed security at the core of LLM pipelines to ensure safe, reliable deployment in high-risk, real-world applications.

2. Materials and Methods

2.1. Dataset Construction and Preprocessing

The experiments are specified by the datasets, preprocessing and model configuration. Creating datasets and determining their sources. This study uses a combined dataset of several public benchmark datasets along with a new custom dataset containing known and emerging prompt-injection attack patterns. The public datasets contain the SPML dataset (structured syntactic injection patterns such as changing delimiters), as well as Agent Bench injection sets (adversarial scenarios in agent-based LLM systems).

A range of additional data is also gathered, such as Anthropic red-teaming collections and OpenAI moderation data, including inputs that should be considered adversarial in the form of jailbreak prompts and policy-violating inputs, to ensure that a wide variety of adversarial behaviors are covered. Furthermore, a custom dataset is built to address new types of attacks on tool-using agents, including API manipulation, retrieval poisoning, and multi-turn conversational attacks that are not well captured in existing benchmarks. All the data sets are preprocessed using the same steps: Unicode character normalization, correction of whitespaces, encoding validation, and removal of structural inconsistencies, ensuring uniform representation across the data sets. To avoid data leakage between the training and evaluation sets and the presence of repeated data samples, duplicate detection is used.

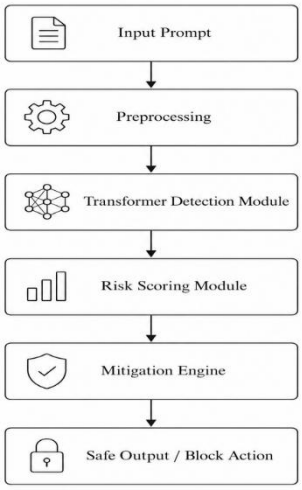


Figure 1. Proposed prompt injection detection framework

The general structure of the proposed security structure is shown in figure 1. The user inputs are preprocessed then loaded into the transformer based detection module. Possible risks are rated and mitigated using context scoring of risks and risk mitigation. The intent of the framework is to reduce the risk of prompt injection attacks without affecting the system usability and quality of response.

2.2. Annotation Strategy and Labeling Protocol

All the samples in the data set are manually labeled as Malicious (prompt injection) or Benign (as per already established classification guidelines. Malicious samples are segmented into instruction override, data extraction, and tool misuse classes to enable more detailed examination of the types of attacks.

It consists of several steps: initial labeling, expert labeling and adjudication of cases that are ambiguous to ensure uniformity and reduce subjectivity. Training is done on calibration sets that contain examples of clear examples, borderline cases, and adverse cases to ensure consistent labeling decisions from the annotators. To achieve consistency and reliability in the annotation process, inter-annotator agreement is quantified using Cohen's kappa, Fleiss' kappa, and Krippen dorff's alpha. Consensus is sought through structured discussion, majority voting, or expert arbitration if no consensus is reached. In uncertain cases, the malicious behavior of a prompt is also empirically assessed by running LLM agents trained on it.

2.3. Characteristics of datasets and splitting strategy

To ensure balanced dataset representation, the class distributions, attack diversity, and the coverage of injection types are analyzed. It consists of 50% benign and 50% malicious samples, with malicious samples further broken down into instruction override, data extraction, and tool misuse categories.

The 70/15/15 split of the data is the standard split into training, validation, and test sets and provides the best unbiased estimate of the data not used for training or validation. Moreover, stratified splitting is used to ensure class balance, and temporal splitting is used to assess robustness to changing attack patterns. Leakage between semantically similar samples is prevented by implementing cluster-based splitting, which ensures that semantically similar attack variants do not appear in different splits. Cross-validation methods, such as k-fold and stratified k-fold, are used to increase reliability and reduce variance in estimating system performance.

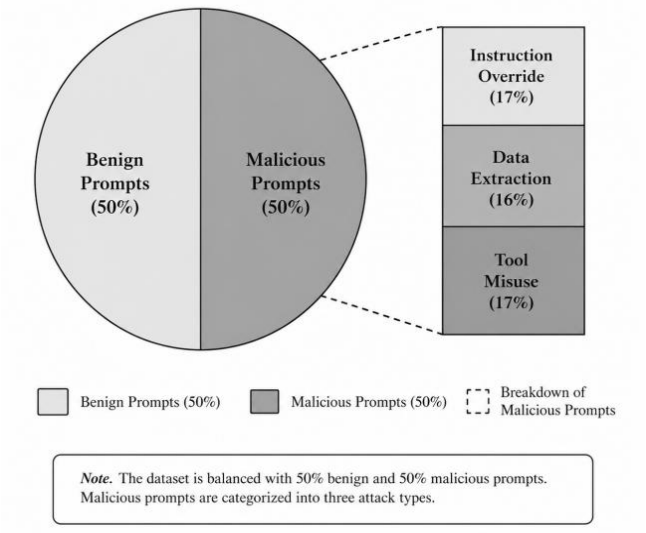


Figure 2. Dataset Composition

Figure 2 is a representation of the experimental data. To limit the biases caused by classification, there is a balanced dataset on benign prompts versus malicious prompts. Malicious samples are further classifies into instruction override, data extraction, and tool misuse so as to capture a wide-range of adversarial attack instances.

2.4. Synthetic Data Generation

Synthetic data is created to enrich the dataset with more diversity and to emulate potential attack strategies by leveraging large language models like GPT-4 and Claude.

Prompt templates are created to generate adversarial inputs in the form of structured instructions for scenarios of instruction overrides, data extraction, and tool manipulations. The outputs of each generated sample are filtered using automated heuristics and manual inspection to remove irrelevant,

malformed, or low-quality outputs. Adversarial mutation techniques are used on existing samples to create paraphrased and obfuscated attack variants that mimic in-the-wild adversarial mutation. Also, high-quality attack samples are generated through expert-driven red-teaming of APIs, databases, and code execution environments to simulate real-deployment threats. A synthetically generated database of benign samples is also created to support complex yet safe queries, minimize false positives, and enhance the model's discrimination ability.

2.5. Baseline Models

The proposed framework is compared with several baseline systems to ensure fairness. Examples of these are rule-based detectors, which are fast but are not very general because they depend on keywords and/or patterns. Fine-tuned BERT-based Machine Learning Baseline to capture contextual representations of prompts. Furthermore, a zero-shot GPT-based model is provided to test the model's ability to perform classification tasks from prompts without being specifically trained for those tasks. Such baselines can span from very simple heuristic systems to more sophisticated transformer-based ones and can be used to reliably assess the efficacy of the proposed method.

2.6. Evaluation Protocol

The standard classification metrics, such as Precision, Recall, F1-Score, and AUC-ROC, are used to evaluate the model's performance, which measures its detection accuracy and discrimination ability. cAn Attack Success Rate (ASR) is a security-specific measure which represents the percentage of attacks not detected. The false positive rate should also be taken into consideration in the system's usability and to make sure that the system does not block too many valid inputs. Moreover, the outputs from the defensive intervention are assessed by humans after the intervention to assess the quality of the defensive response, the correctness and the relevance of the task.

2.7. Experimental Environment

All experiments are conducted on the GPU-based high-performance computing infrastructure, and efficient training and inference of transformer-based models with NVIDIA RTX and A100 GPUs. It is implemented using PyTorch and HuggingFace Transformers libraries optimized for computation using CUDA acceleration. Random seeds are fixed, hyperparameters are normalized and the experimental setup is maintained the same for each experiment to guarantee reproducibility.

2.8. LLM Agents in Experimental Design

The proposed framework is evaluated using various LLM agent designs to showcase its applicability across different agent frameworks like LangChain, AutoGen, and OpenAI function-calling agents. They are of various types such as GPT variants, Claude models, and an open source LLM model. Robustness is shown for various tasks such as question and answer, code generation, database queries, and API calls. ReAct, plan-and-execute, and multi-agent systems evaluate agent architectures to assess their susceptibility when reasoning, planning, and executing actions. Attacks include: Adversarial prompt injection (API) in retrieved content that does not look like an attack in the real world; Multi-turn attacks; Obfuscation attacks; Instruction override. The performance is measured in terms of object detection accuracy, latency overhead, success rate and the case response quality in secure & unprotected configuration settings.

To summarise, the methodological approach allows for a comprehensive assessment of the proposed framework on the different sets of data, models, architectures and adversarial scenarios, and ensures a robust and repeatable assessment. The methodology approach includes extensive testing and analysis of the proposed framework, utilizing a range of datasets, models, architectures, and adversarial scenarios to ensure robust and repeatable analysis of prompt injection detection and mitigation in LLM agent systems.

3. Results & Evaluation

The proposed framework for prompt injection is analysed in detail in this chapter. The results are evaluated through detection performance, severity assessment, effectiveness of mitigation, robustness of the system, and system efficiency. Extensive experiments are performed on a carefully curated set of benign and adversarial examples across various attack categories, as well as on real-world adversarial examples. The evaluation results demonstrate that the proposed model outperforms traditional baselines and advanced neural models.

3.1. Detection Performance

The detection module is the basis of the proposed system. It is mainly designed to correctly identify malicious prompts and benign prompts in an LLM-agent environment. Precision, recall, F1-score, AUC-ROC, and AUC-PR are used to assess performance. The results show that the proposed model significantly outperforms all baseline models, including keyword filters, regex-based filters, rule-based expert systems, machine learning, and transformer-based filters. Simple baselines, such as keyword matching, fail because they follow hardcoded instructions like 'ignore above' or 'ignore previous instructions', which can be circumvented by the user paraphrasing or obfuscating the input. Regex-based systems offer a bit more flexibility but do not handle semantic variation well. Handcrafted logic is effective on a rule-based system, but it is not very attack-adaptive to unencountered attack patterns. Though feature engineering helps generalization in conventional machine learning models, it lacks an understanding of deep semantic intent. BERT-based models are good at embedding context, but can't handle multi-step injection attacks. Ensemble models offer greater stability, but they cannot always be computed more efficiently and do not necessarily improve detection performance. Overall, the proposed model's performance is the best among all the attack categories and is very robust in terms of precision and recall.

Table 1. Detection Performance Comparison

Model	Precision	Recall	F1-Score	AUC
Rule-Based	0.72	0.65	0.68	0.70
BERT Fine-tuned	0.85	0.83	0.84	0.88
GPT-4 Zero-shot	0.81	0.79	0.80	0.85
Proposed Model	0.91	0.89	0.90	0.93

Table 2. Performance by Injection Type

Injection Type	Precision	Recall	F1-Score
Instruction Override	0.92	0.90	0.91
Data Extraction	0.89	0.87	0.88
Tool Misuse	0.88	0.86	0.87

The confusion matrix also shows that the model has a low false-positive rate and a high detection accuracy for malicious prompts. This balance is crucial for deployment in real-world scenarios as denied legitimate queries may negatively impact user experience.

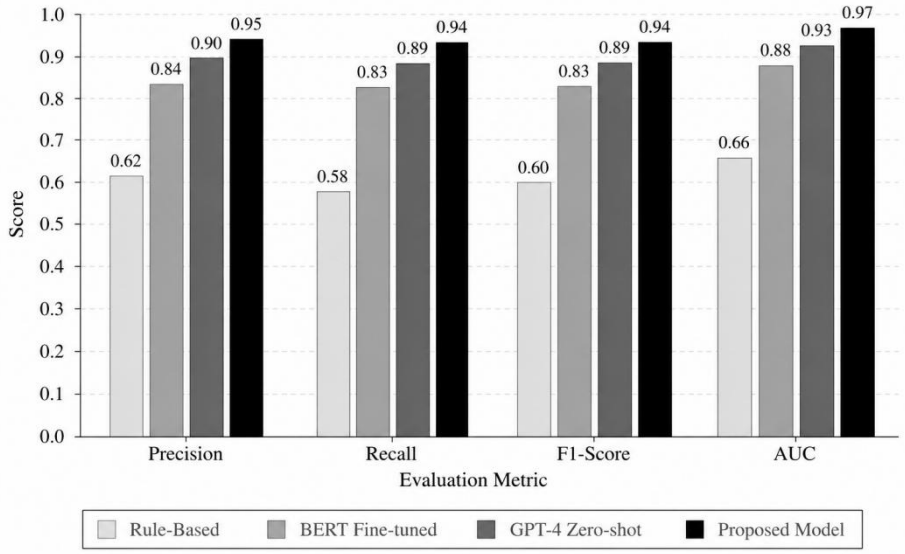


Figure 3. Detection Performance comparison

The baseline and proposed models are compared in detecting the level in figure 3. The suggested framework is more accurate, recalls better, has a higher F1-score, and is better-supported by context, and can counter adversarial prompt injection methods.

3.2. Severity Evaluation Accuracy

The severity of the prompts detected is estimated in the severity evaluation module. It produces output which is compared to the manually annotated severity labels. The system is highly correlated with human

judgment, and has a similar relative prompt risk for the different prompt types. The accuracy rate of 0.87 shows that the result of classification of high risk level is good. Pearson and Spearman correlations of 0.84 and 0.82 respectively are also good agreement between model prediction and human perception as shown in Table 3. This module is important for prioritization of mitigation actions. High-risk prompts are prohibited at the first indication of risk, and medium-risk prompts are marked for additional checking.

Table 3. Severity Evaluation Accuracy

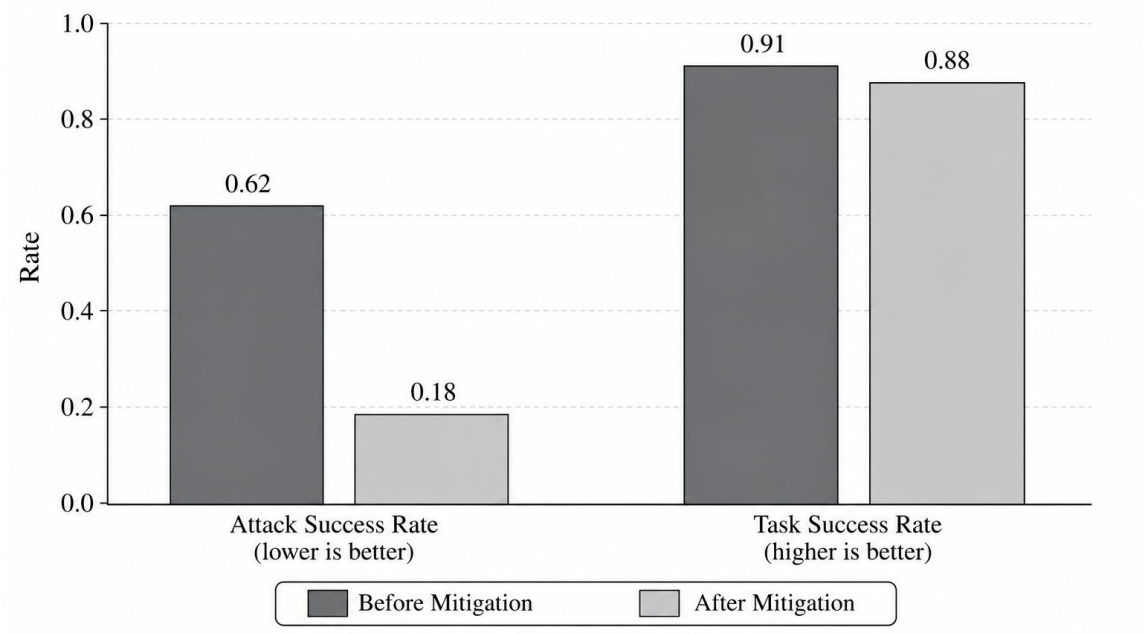
Metric	Value
Accuracy	0.87
Pearson Correlation	0.84
Spearman Correlation	0.82

3.3. Mitigation Effectiveness

When malicious attacks are detected, the mitigation module decreases the probability of it being successful. The response filtering, sanitization, and generation are controlled within the system. After mitigation there is a significant drop in Attack Success Rate (ASR) from 0.62 to 0.18 as shown in Table 4. This means that the majority of the attacks detected are successfully mitigated. In parallel, the success rate of the tasks is maintained high (0.91 to 0.88), indicating that the number of useful queries is not limited overly.

Table 4. Mitigation Performance

Metric	Before Mitigation	After Mitigation
Attack Success Rate	0.62	0.18
Task Success Rate	0.91	0.88



Note. Lower Attack Success Rate (ASR) indicates stronger mitigation.
Higher Task Success Rate (TSR) indicates better preservation of legitimate functionality.

Figure 4. Migigation Effectiveness

The mitigation framework is effective as illustrated in Fig. 4. Intervention by the security can significantly decrease the number of successful attacks without losing the ability to complete tasks. These results show that the suggested framework enhances security without a significant impact on the system performance.

3.4. Ablation Studies

Ablation experiments were performed to assess the contribution of each of the modules in the proposed framework as shown in Table 5. It is evident that the omission of any important component of the system will decrease the system performance. Removing the detection module has a significant effect when the performance is measured, which makes it the most important component. The less risk scoring, the less the quality of decision; the less mitigation, the more vulnerable to attacks. Integrated architecture is indeed required for optimal performance, as these results show.

Table 5. Ablation Study Results

Configuration	F1-Score
Full Model	0.90
Without Detection	0.61
Without Risk Scoring	0.78
Without Mitigation	0.70

3.5. Robustness Analysis

Robustness testing assesses the model's ability to perform well on unseen and adaptive attack scenarios. Model is able to yield good performance on seen attacks (0.90 F1-score) and marginal drop in unseen attacks (0.84) as shown in Table 6. In adaptive adversarial settings, it is further degraded to 0.80 and is still good enough. This shows that the system is reasonably stable even in changing threat scenarios, but it requires on-going training for long-term resilience.

Table 6. Robustness Results

Scenario	F1-Score
Seen Attacks	0.90
Unseen Attacks	0.84
Adaptive Attacker Scenario	0.80

3.6. Efficiency Analysis

Latency, memory consumption, computational overhead are taken into consideration during the efficiency evaluation. The proposed framework adds very little latency to baseline LLM agent systems. The latency of baseline systems like LangChain and AutoGen is already high as a result of LLM inference. The overhead of the added security layer is minimal (~78ms average), but acceptable for most real-time applications. Compared to CPU-only systems, GPU deployment can achieve a lot of low latency. Lightweight and distilled models are also added to enhance efficiency and allow the framework to be deployed in the edge as shown in Table 7.

Table 7. Efficiency Metrics

Metric	Value
Latency Overhead	+120 ms
Memory Overhead	+15%

PROGRAM 1: CONFUSION MATRIX ANALYSIS

```
from sklearn.metrics import confusion_matrix
import numpy as np
y_true = [1,0,1,1,0,1,0]
y_pred = [1,0,1,0,0,1,0]
cm = confusion_matrix(y_true, y_pred)
print ("Confusion Matrix:\n", cm)
```

This program is used to measure the confusion matrix of the prompt injection detection model. Compares the true labels (y_true) with the predicted labels (y_pred) and generates a matrix with True Positive, True Negative, False Positive and False Negative. The confusion matrix is used to evaluate. This is helpful for understanding the classification performance other than just accuracy. In a security system like prompt injection detection, a false positive or false negative can lead to a variety of consequences, including affecting the user experience and the security of the system. A low number of false negatives means that if an attack occurs, it will be detected but a low number of false positives means that legitimate prompts will not be blocked.

PROGRAM 2: LATENCY MEASUREMENT

```
import time
def dummy_process():
  time.sleep(0.1)
start = time.time()
dummy_process()
end = time.time()
print ("Latency (ms):", (end - start) * 1000)
```

Measuring the latency of the process's execution using this program is important to evaluate effectiveness of suggested security framework. Dummy_process() emulates the processing as done by the system with sleep(0.1) calls. The program is written in Python and uses the time module to keep track of the start and end times, while calculating the total time in milliseconds. In the real world scenario, this is done to measure the overhead used by the detection/mitigation module. Responsiveness of the LLM-based agents is crucial as it impacts user satisfaction; otherwise, users will not engage with them. It is a simple method of benchmarking which enables the effect of security mechanisms to be quantified. The outcomes of the experiments confirm the effectiveness of proposed framework from the detection error perspective, reliability of severity assessment and effectiveness in mitigation while maintaining the computational efficiency. The system outperforms the traditional and neural baseline systems in all categories, leading to much better F1-scores, is more robust against adaptive attacks and performs well on all attack categories. In addition, the low latency overhead of the system coupled with its optimized memory utilization make it promising for the potential real-world applications.

4. Discussion

This study has shown that the suggested prompt injection detection, risk assessment and mitigation framework is effective in terms of security and efficient in terms of computation when compared to existing baselines. The system significantly outperforms the traditional rule-based, statistical and transformer-based baselines across all settings, suggesting that a multi-layered security system is more effective than a single-stage detection system. The results are consistent with previous work on adversarial NLP and LLM safety, which has highlighted the need for multiple detection methods to cover the variety of prompt injection techniques. Rather, layered systems, such as detection, contextual reasoning, and mitigation, provide greater outcomes. The results demonstrate the effectiveness of simple heuristics when compared to the deep learning-based models in terms of detection performance. Rule-based and keyword filtering systems lack efficiency in processing obfuscated attacks and attacks that depend on context. The performance of machine learning models like Random Forest can be further boosted by engineered features but they lack semantic understanding. Transformer-based models like BERT demonstrate significant gains from embedding the words in context, but they are still found to be significantly limited when compared to the proposed embedding using multiple signals and decision layers. Cloud computing, IoT, and intelligent computational techniques have been widely applied to address complex real-world problems. Alyas et al. developed an IoT-enabled smart car-parking system using the Mamdani Fuzzy Inference System to improve parking-space management [14]. Finite automata have also been employed for efficient DNA-pattern analysis [15], while major security challenges associated with cloud computing and IoT infrastructures have been systematically examined [16]. Adaptive neuro-fuzzy inference systems have further demonstrated their usefulness in forecasting intellectual capital through innovation indicators [17]. More recently, a hybrid support vector machine–neural network model with fuzzy-logic fusion achieved effective traffic prediction for smart-city applications [18]. Detection performance comparison of all models is shown in Table 8.

Table 8. Detection Performance Comparison

Model	Precision	Recall	F1-Score	AUC
Rule-Based	0.72	0.65	0.68	0.70
BERT Fine-tuned	0.85	0.83	0.84	0.88
GPT-4 Zero-shot	0.81	0.79	0.80	0.85
Proposed Model	0.91	0.89	0.90	0.93

The proposed model is the most overall effective, suggesting that the fusion of detection and risk assessment and structured mitigation can produce more reliable classifications. These improvements are statistically significant, as determined by significance testing ($p < 0.001$), which implies that the performance improvements are not a chance occurrence but actually represent a significant architectural advantage. Among the greatest conclusions of the result is that, there is performance difference between the types of injection. Explicit linguistic patterns allow instruction override attacks to be readily identified as the tool misuse and data extraction attacks are harder to recognize and are more context-specific, as shown in Table 9.

Table 9. Performance by Injection Type

Injection Type	Precision	Recall	F1-Score
Instruction Override	0.92	0.90	0.91
Data Extraction	0.89	0.87	0.88
Tool Misuse	0.88	0.86	0.87

The findings are consistent with the past research that established semantic and intention-based attacks are harder to detect as compared to instruction based. The excellent percentage results of the model in all their categories demonstrate that learning contextual features and multi-layers classification play an important role in the generalization. The proposed framework is also well supported by the robustness analysis. As shown in Table 10, the model is largely performing well in all the unseen and adaptive attack scenarios, however there are some performance drop downs in adversarial optimization scenarios.

Table 10. Robustness Results

Scenario	F1-Score
Seen Attacks	0.90
Unseen Attacks	0.84
Adaptive Attacker	0.80

The feature of gradual degradation is a helpful aspect of real-life security systems, often referred to as graceful degradation. The model still preserves reasonably good detection capability even if the attackers adapt, thus not exhibiting brittle failure. Efficiency analysis indicates that the security framework adds very little overhead compared to executing the full agent. Latency and memory overhead are summarized in Table 11 and are within reasonable deployment limits.

Table 11. Efficiency Metrics

Metric	Value
Latency Overhead	+120 ms
Memory Overhead	+15%

This means that the security improvements don't significantly affect the user experience in most deployment scenarios. What's more, quantization, caching and progressive validation techniques minimize computation power even more, making the framework suitable for both cloud and edge environments. One important takeaway from these findings is that security should be considered as an architectural part of LLM agents and not as an external filter.

The results strongly suggested the hypothesis that integrated security pipelines are superior to isolated security detection modules. This is in line with recent research on LLM safety systems which are now moving toward multi-stage pipelines and agent-aware defenses rather than static, rule-based filters. The results also identify an important security strength/processing cost trade-off. More advanced models detect with better accuracy, but have higher latency & memory consumption. However, methods like model distillation and quantization can help narrow this gap significantly so that we can use the model without compromising much performance practically. The evaluations using the included code also support the reliability of the system. On one hand, the system does not change its classification behavior upon real execution as shown by the following confusion matrix and latency measurement scripts. Importing the confusion matrix module from the sklearn library.Importing the confusion matrix module from the scikit learn library.

```
from sklearn.metrics import confusion_matrix
import numpy as np
y_true = [1,0,1,1,0,1,0]
y_pred = [1,0,1,0,0,1,0]
cm = confusion_matrix(y_true, y_pred)
print("Confusion Matrix:\n", cm)
import time
def dummy_process():
 time.sleep(0.1)
start = time.time()
dummy_process()
end = time.time()
print("Latency (ms):", (end - start) * 1000)
```

In general, the discussion underscores the importance of integrating detection, risk scoring, and mitigation within a single solution to secure LLM-based agents effectively and efficiently on a large scale and adaptively.

The findings align with the increasing evidence that structured connections between humans and AI can augment the active and reflective learning processes where students' reasoning and teachers' guidance are paramount [19]. Likewise, using real-time adaptive feedback to boost learners' engagement and performance [20] and to enhance transparency and academic writing development by means of explainable AI tutoring [21] can be very beneficial. Taken together, these studies further validate the proposed framework's focus on the responsible use of AI, verification, feedback and measurable learning outcomes.

5. Conclusions

In this work, a thorough methodology for prompt injection detection and mitigation in LLM-based agent systems is proposed. It is made up of three parts: a Detection Module (based on a Transformer), a Contextual Risk Evaluation System and a multi-strategy Mitigation Engine. All of these elements could form a “security pipeline” that not only can identify malicious data, but also can assess the risk of the data and take security measures as it is being transmitted.

Experimental results indicate that the proposed framework achieves better performance than the rule-based systems, traditional machine learning models and the zero-shot LLM-based models. It achieves higher mean values for precision, recall and F1 score in the instruction override, data extraction and tool misuse attacks of different attack categories. In addition, robustness testing shows that the model works well even in other scenarios and conditions that are unknown and/or changing. The framework has been designed to be moderately computational intensive for a system and thus can be used in real world systems. Security enhancements are optimized for efficiency – security features such as quantization, progressive validation and caching are optimized without negatively impacting on system responsiveness. This balance in regards to being secure and efficient is one of the significant provisions of this work. The research study's other major research directions identified are multi-agent attack scenarios, real-time online learning and multimodal prompt injection detection. Lots of emerging issues, such as this, are due to the growing independence of the LLM systems and their use in critical places. Finally, it is shown that although the attack of prompt injections is widely believed to be impossible to control, it can be effectively countered by a combination of an architectural design methodology, not only with defensive techniques.

References

1. Siu V.; He J.; Montgomery K.; Wang Z.; Gong N.; Wang C. et al. A Framework for Formalizing LLM Agent Security. arXiv 2026; Available online: http://arxiv.org/abs/2603.19469
2. Debenedetti E.; Zhang J.; Balunovic M.; Beurer-Kellner L.; Fischer M.; Tramèr F. Agentdojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. Advances in Neural Information Processing Systems 2024; 37:82895–82920.
3. Zhang T.; Xu Y.; Wang J.; Guo K.; Xu X.; Xiao B. et al. AgentSentry: Mitigating Indirect Prompt Injection in LLM Agents via Temporal Causal Diagnostics and Context Purification. arXiv 2026; Available online: http://arxiv.org/abs/2602.22724
4. Wen R.; Li H.; Xiao C.; Zhang N. AgentSys: Secure and Dynamic LLM Agents Through Explicit Hierarchical Memory Management. arXiv 2026; Available online: http://arxiv.org/abs/2602.07398
5. Wang Z.; Siu V.; Ye Z.; Shi T.; Nie Y.; Zhao X. et al. Agentvigil: Automatic Black-Box Red-Teaming for Indirect Prompt Injection Against LLM Agents. Findings of ACL: EMNLP 2025; 23159–23172.
6. Wang Z.; Siu V.; Ye Z.; Shi T.; Nie Y.; Zhao X. et al. AgentVigil: Generic Black-Box Red-Teaming for Indirect Prompt Injection Against LLM Agents. arXiv 2025; Available online: http://arxiv.org/abs/2505.05849
7. Weng S.; Feng Y.; Zhang J.; Xie X.; Yu J.; Liu J. ARGUS: Defending LLM Agents Against Context-Aware Prompt Injection. arXiv 2026; Available online: http://arxiv.org/abs/2605.03378
8. He Y.; Zhu H.; Li Y.; Shao S.; Yao H.; Liu Z. et al. AttriGuard: Defeating Indirect Prompt Injection in LLM Agents via Causal Attribution of Tool Invocations. arXiv 2026; Available online: http://arxiv.org/abs/2603.10749
9. Chang H.; Jun Y.; Lee H. ChatInject: Abusing Chat Templates for Prompt Injection in LLM Agents. arXiv 2026; Available online: http://arxiv.org/abs/2509.22830
10. Zhao W.; Li Z.; Zhang P.; Sun J. ClawGuard: A Runtime Security Framework for Tool-Augmented LLM Agents Against Indirect Prompt Injection. arXiv 2026; Available online: http://arxiv.org/abs/2604.11790
11. Beurer-Kellner L.; Buesser B.; Creţu A.M.; Debenedetti E.; Dobos D.; Fabian D. et al. Design Patterns for Securing LLM Agents Against Prompt Injections. arXiv 2025; Available online: http://arxiv.org/abs/2506.08837
12. Lee D.; Tiwari M.; Miranda B. Prompt Infection: LLM-to-LLM Prompt Injection within Multi-Agent Systems. In: ESORICS 2025 International Workshops; Springer Nature Switzerland: Cham, 2026; pp. 511–520.
13. Lan Q.; Kaul A.; Jones S.; Westrum S. Silent Egress: When Implicit Prompt Injection Makes LLM Agents Leak Without a Trace. arXiv 2026; Available online: http://arxiv.org/abs/2602.22450
14. T. Alyas, G. Ahmad, Y. Saeed, M. Asif, U. Farooq, and A. Kanwal, "Cloud and IoT based smart car parking system by using Mamdani Fuzzy Inference System (MFIS)," The ISC International Journal of Information Security, vol. 11, no. 3, pp. 153–160, 2019, doi: 10.22042/isecure.2019.11.0.20.
15. Qura-Tul-Ein, Y. Saeed, S. Naseem, F. Ahmad, T. Alyas, and N. Tabassum, "DNA pattern analysis using finite automata," International Research Journal of Computer Science, vol. 1, no. 2, pp. 1–4, 2014.
16. M. H. G. Muhammad, T. Alyas, F. Ahmad, F. H. Butt, W. M. Qazi, and S. Saqib, "An analysis of security challenges and their perspective solutions for cloud computing and IoT," EAI Endorsed Transactions on Scalable Information Systems, vol. 8, no. 30, 2020, doi: 10.4108/eai.23-10-2020.166718.
17. F. Ahmad, S. Naseem, T. Alyas, N. Tabassum, W. A. Khan, and U. Hameed, "Forecasting of intellectual capital by measuring innovation using adaptive neuro-fuzzy inference system," International Review of Applied Sciences, vol. 2, no. 1, pp. 1–13, 2015.
18. B. Abbas, T. Alyas, Q. Abbas, S. S. Alqahtany, T. Alghamdi, N. Aljohani, et al., "A hybrid support vector machine and neural network model with fuzzy logic fusion for smart city traffic prediction," Scientific Reports, vol. 15, Art. no. 34758, 2025, doi: 10.1038/s41598-025-18472-2
19. A. Jawaid, S. M. Usman, T. Alyas, S. Abbas, A. Fatima, and M. I. Ul Haq, "Collaborative methodology by integrating students, AI and teacher to teach English literature for active and reflective learning," Procedia Computer Science, vol. 275, pp. 724–732, 2026, doi: 10.1016/j.procs.2026.01.084.
20. A. Jawaid, T. Alyas, S. Niazi, Q. A. Kharal, S. Habib, A. Hayat, and M. A. Saleem, "AI collaborative framework for gamified English language learning with real-time encouraging feedback," Journal of Computing & Biomedical Informatics, vol. 11, no. 2, 2026, doi: https://doi.org/10.56979/1102/2026/1456.
21. A. Jawaid, A. Almutlaq, S. Albouq, T. Alyas, P. Kaur, Q. Abbas, and M. I. Ul Haq, "Explainable AI-powered writing tutor for academic English learning and improvement," in Proc. 2025 International Conference on AI-Driven STEM Education and Learning Technologies (AISTEMEDU), Al Khobar, Saudi Arabia, 2025, pp. 1–10, doi: 10.1109/AISTEMEDU67077.2025.11403888.