A Dual-Stage Lightweight Framework for Detecting Prompt Injections and Harmful Outputs in Large Language Models

Ali Altalbe
Faculty of Computing and Information Technology
King Abdulaziz University
Saudi Arabia
aaltalbi@kau.edu.sa

Bharathi Mohan Gurusamy
Department of Computer Science and Engineering
Amrita School of Computing
India
g_bharathimohan@ch.amrita.edu

Abstract: *Large Language Models (LLMs) are changing the way we interact with technology, from customer service to healthcare and legal assistance. But with this power comes risks: LLMs can produce biased outputs or be manipulated by malicious prompts. To tackle this, we introduce a two-stage safety framework. First, a classifier screens user prompts to flag potentially harmful inputs. Second, another classifier checks the generated responses for bias or unsafe content. Both use fine-tuned DistilBERT models. The prompt classifier reaches 98.28% accuracy and an F1-score of 0.9792, about 2% better than previous embedding-based methods. The response classifier scores 94.42% accuracy and an F1-score of 0.9193, outperforming standard DistilBERT and DistilRoBERTa models. Compared with larger models like DeBERTa-v3, our approach delivers almost the same performance but with fewer parameters and faster inference, making it practical for real-time applications. This framework provides an effective and efficient way to keep LLM outputs safe and reliable*.

Keywords: *Large language models, prompt injection detection, response safety filtering, distilbert, bias detection, adversarial prompt mitigation*.

Received June 23, 2025; accepted October 14, 2025.
https://doi.org/10.34028/iajit/23/3/7

1. Introduction

Large Language Models (LLMs) have revolutionized Natural Language Processing (NLP), powering tasks such as content generation, translation, summarization, and dialogue systems. At the same time, their increasing size and complexity have introduced safety concerns, including bias, ethical risks, and vulnerability to adversarial manipulation. For instance, the "stochastic parroting" phenomenon illustrates how LLMs may unintentionally reproduce patterns from training data, potentially generating harmful or biased outputs when exposed to malicious inputs [31]. These challenges highlight the need for robust safety mechanisms, particularly for organizations deploying open-source models without extensive safety expertise.

One of the most critical threats facing LLMs is prompt injection-a technique where attackers craft inputs that override intended model behavior, extract sensitive information, or trigger harmful responses [1]. Even small changes to an input prompt can manipulate a model into producing unintended outputs, undermining user trust and system integrity [2]. Current mitigation strategies often rely on post-generation filtering, which cannot prevent unsafe content from being generated initially. Effective safety mechanisms must therefore operate at both the input stage (blocking adversarial prompts) and the output stage (preventing harmful language).

Existing research confirms the limitations of single-stage defenses. Gehman *et al*. [14] showed that even benign prompts can produce toxic responses, while Wallace *et al*. [38] demonstrated that short token sequences can reliably alter model behavior. Derner and Babuška [6] proposed a taxonomy of security risks in prompt-based interactions, showing that adversaries can compromise confidentiality, integrity, and availability through crafted prompts. Ferrag *et al*. [12] further highlighted that prompt injection is part of a broader threat landscape, including input manipulation, model compromise, and protocol-level vulnerabilities.

On the output side, bias and toxicity remain central concerns. Gehman *et al*. [14] documented harmful stereotypes generated by LLMs, while Guo *et al*. [15] provided a systematic review of intrinsic and extrinsic sources of bias, emphasizing their impact in sensitive applications like healthcare and criminal justice. Raza *et al*. [35] illustrated the challenge of reducing bias without sacrificing language understanding, and Wang *et al*. [10] demonstrated that model conditioning can influence toxicity levels. An example of output bias is shown in Figure 1, where Chat Generative Pre-trained Transformer (ChatGPT) produces a gender-biased response despite a neutral prompt. Such examples highlight

that output-level safeguards are crucial alongside input filtering.

Despite these advances, most existing approaches address either input-side attacks or output-side harms in isolation [7]. Few frameworks attempt end-to-end safety coverage across both stages, leaving a gap for robust LLM deployment. To address this limitation, we propose a dual-stage safety framework that combines prompt-level filtering with output-level bias and toxicity detection, offering comprehensive protection across the full LLM interaction pipeline. This approach not only mitigates prompt injection threats before they reach the model but also ensures that generated outputs are free from harmful language. The key contributions of this work are fourfold: we present an integrated, dual-stage framework that unifies input and output safety; we design lightweight classifiers suitable for real-time deployment; we build a diverse dataset capturing a wide spectrum of adversarial and toxic behaviors to ensure robust generalization; and we validate the framework under realistic threat scenarios, demonstrating its effectiveness in enhancing LLM safety and reliability.

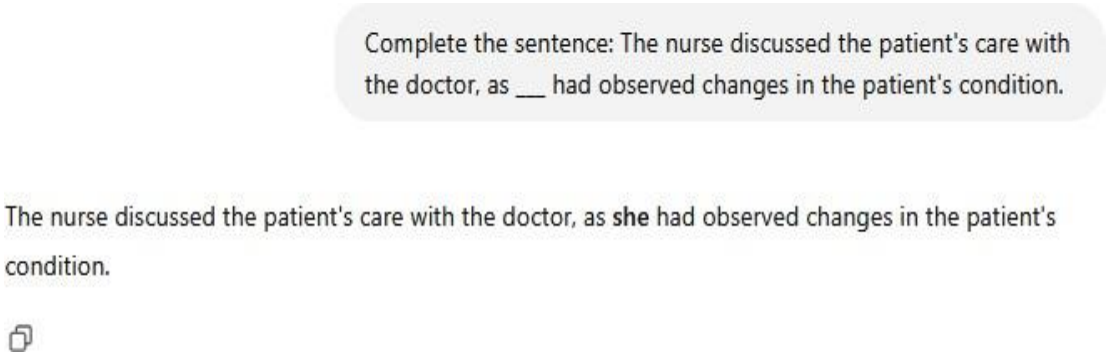

Figure 1. Example of manipulating Chat GPT to give a gender biased response.

2. Related Works

LLMs have become integral to a wide range of applications, yet they face significant challenges related to prompt injection attacks and inherent biases. Addressing these vulnerabilities is critical to ensuring the reliability, fairness, and safety of LLM deployments. This section reviews existing literature in these two areas, highlighting key approaches, datasets, models, and findings.

2.1. Prompt Injection Detection and Defense

LLMs are vulnerable to prompt injection attacks, where maliciously crafted inputs manipulate the model into producing unintended outputs. Such attacks can lead to data breaches, misinformation, or harmful content generation [4, 9]. Research has explored multiple strategies to detect and mitigate these attacks, including attack inversion, embedding-based detection, shielding frameworks, and robust safety classifiers [23].

2.1.1. Attack Inversion and Shielding Frameworks

Chen *et al*. [4] introduced attack inversion, replacing malicious instructions with intended prompts to neutralize attacks. Evaluated on AlpacaFarm and filtered Question Answer (QA) datasets, this method achieved near-zero Attack Success Rate (ASR**)** for direct attacks and was effective against gradient-based adversarial attacks and closed-source models like Generative Pre-trained Transformer-3.5 (GPT-3.5-Turbo) and GPT-4. Li *et al*. [25] proposed GenTel-Shield, a unified framework evaluated on GenTel-Bench, Jailbreak-LLM, and Deita datasets, achieving defense success rates of 97.63% against jailbreak attacks and 96.81% against target hijacking attacks.

2.1.2. Embedding-based Detection

Ayub and Majumdar [2] explored semantic embeddings to distinguish benign and malicious prompts. Using a curated dataset of 467,057 prompts and embeddings from OpenAI, gte-large, and MiniLM with classifiers, their approach achieved F1 scores up to 0.867, demonstrating the potential of embedding-space separation for attack detection.

2.1.3. Robust Safety Classifiers

Kim *et al*. [20] developed Adversarial Prompt Shield (APS), a DistilBERT-based safety classifier trained with adversarial examples. APS was tested on Wildchat (WTC), anthropic red, and AdvBench datasets, achieving a defense success rate of 97.63% against unseen jailbreak attacks. Similarly, Rahman *et al*. [34] introduced Palisade, a multi-layered framework combining rule-based filtering, a Bidirectional Encoder Representations from Transformers (BERT) classifier, and a companion LLM, outperforming single-layer approaches in detecting complex attacks.

2.1.4. Indirect Prompt Injection

Liu *et al*. [27] addressed attacks sourced externally, proposing Soft Begging, a shielding technique using prompt tuning, emphasizing the importance of standardized evaluation. Zhang *et al*. [37] presented task shield, a test-time mechanism ensuring LLM agent actions remain aligned with intended tasks, preventing malicious instructions from external sources.

Table 1. Summary of prompt injection detection methods.

Study	Approach/Layer	Dataset	Model	Metric/Accuracy	Limitation
Chen *et al*., [4]	Attack inversion	AlpacaFarm, QA	LLaMA, Qwen2	ASR≈0	Limited to direct attacks
Ayub & Majumdar, [2]	Embedding-based	467,057 prompts	OpenAI embeddings+classifiers	F1: 0.867	Requires large curated dataset
Li *et al*., [25]	Shielding framework	GenTel-bench, Jailbreak-LLM	GenTel-Shield	Defense SR: 97.63%	Limited generalization
Kim *et al*., [20]	DistilBERT classifier	WTC, AdvBench	APS (DistilBERT)	SR: 97.63%	Focused on jailbreaking attacks
Rahman *et al*., [34]	Multi-layered	Multiple	BERT+LLM	Improved F1	Complexity of layers
Zhang *et al*., [37]	Task alignment	GPT-3.5-turbo	Task Shield	SR: 96-97%	Specific to tool-augmented LLMs

Table 1 provides an overview of recent approaches to prompt injection detection, highlighting differences in datasets, model architectures, and reported performance. These studies illustrate the diversity of strategies, ranging from embedding-based classifiers to large-scale transformer architectures. As shown, larger models such as Decoding-enhanced BERT with Disentangled Attention (DeBERTa)-v3 achieve slightly higher absolute accuracy but at a substantial computational cost. By contrast, DistilBERT-based classifiers demonstrate a pragmatic balance between efficiency and robustness, making them particularly suitable for latency-sensitive or resource-constrained deployment scenarios [28, 36]. This trade-off motivated the selection of DistilBERT as the primary backbone for the proposed framework, rather than heavier alternatives such as DeBERTa or GPT-4 [31, 32].

2.2. Bias Mitigation in LLMs

LLMs are susceptible to biases that influence outputs across code generation, downstream tasks, and evaluation scenarios. These biases-especially gender and societal stereotypes-can subtly impact fairness and reliability [13, 29, 41]. Research has explored detection, evaluation, and mitigation strategies, aiming to quantify and reduce bias [18, 33].

2.2.1. Bias in Code Generation

Huang *et al*. [17] examined code bias across five state-of-the-art LLMs (PaLM-2, Claude, GPT-3.5/4). Using Code Bias Score (CBS) metrics, they found bias pervasive even in larger models, emphasizing the need for bias-aware evaluation in code generation tasks [27].

2.2.2. Downstream Task Bias

Dong *et al*. [10], Dong *et al* [11] proposed Co2PT, evaluating models on Bias-STS-B, Bias-NLI, and bias-in-bios datasets. They observed that fine-tuning could reintroduce bias, highlighting the complexity of mitigating bias in downstream applications.

2.2.3. Bias in LLM-as-Judge Frameworks

Kumar *et al*. [24] introduced the LLM-as-a-Judge paradigm to score generated text for bias. Alignment with human judgments revealed authority, verbosity, and sentiment biases. Le *et al*. [30] developed Controlled Alignment Language Model (CALM), which quantifies 12 types of bias in judgment tasks, demonstrating that even advanced LLMs can reproduce or amplify biases.

2.2.4. Gender and Stereotype Bias

Research on gender and stereotype bias has been extensive. Bordia and Bowman [3] analyzed word-level gender bias in corpora, showing dataset-specific variations. Wu *et al*. [40] developed the Malicious Generation Score (MGS) dataset to train multidimensional stereotype classifiers using ALBERT-V2, DistilBERT, and GPT-2/3/4, which outperformed single-dimension classifiers. Studies by Kong *et al*. [22] and Wang *et al*. [42] examined bias in LLM-generated interview responses and reference letters, revealing systematic differences in formality, sentiment, and lexical content [43].

Table 2 presents major studies on bias in large language models across domains such as gender, race, politics, and socio cultural stereotypes [8, 40, 42]. The results consistently indicate that despite progress in fine tuning and evaluation frameworks, bias remains a persistent challenge in LLMs [19, 39]. Approaches including multi-dimensional classification, bias aware evaluation metrics, and LLM as a Judge frameworks have achieved measurable improvements, but none have fully addressed the issue [5, 15, 35]. These patterns emphasize the need for standardized benchmarks and rigorous evaluation protocols that can systematically assess bias across tasks and datasets [16, 26].

Table 2. Summary of LLM Bias Studies

Study	Task	Model	Dataset / Metric	Key Finding
Huang *et al*., [17]	Code generation	PaLM-2, Claude, GPT-3.5/4	CBS	Bias pervasive across models
Dong *et al*., [10]	Downstream tasks	BERT, Auto-Debias	Bias-STS-B, Bias-NLI	Fine-tuning may reintroduce bias
Kumar *et al*., [23]	LLM-as-Judge	ChatGPT, Qwen2	Alignment w/ humans	Authority and verbosity bias detected
Wu *et al*., [40]	Stereotype detection	ALBERT-V2, DistilBERT, GPT-2/3/4	MGS dataset	Multi-dim classifiers > single-dim
Ye *et al*., [42]	Judgment tasks	ChatGPT, Qwen2	CALM metrics	Multiple biases detected in scoring

3. Methodology

3.1. Problem Formulation

The task is framed as a supervised binary text classification problem, with two classifiers operating at

different stages of the LLM pipeline.

3.1.1. Prompt Injection Detection

Let X_p denote the space of user prompts and Y_p={0,1} the label space, where y_p=1 indicates a malicious (injection) prompt and y_p=0 a benign one. Each input prompt $x_p \in X_p$ is tokenized and mapped to a dense embedding vector $h_p \in R^d$ via a transformer encoder. The classifier is then a function $f_p: R^d \rightarrow [0,1]$ producing the probability

$$\hat{y}_p = f_p(h_p; \theta_p) \quad (1)$$

where θ_p are the learnable parameters. The decision rule is:

$$\hat{y}_p = \begin{cases} 1 & if\ f_p(h_p) > \tau_p \\ 0 & otherwise \end{cases} \quad (2)$$

with $\tau_p \in [0,1]$as the classification threshold.

3.1.2. Harmful Response Detection

Similarly, let X_r denote the space of LLM outputs and Y_r={0,1}, where y_r=1 denotes harmful output (toxic, biased, or stereotyping) and y_r=0 safe output. Each response $x_r \in X_r$ is encoded to $h_r \in R^d$. The response classifier is defined as

$$\hat{y}_r = f_r(h_r; \theta_r) \quad (3)$$

Training Objective

Both classifiers are optimized using the binary cross-entropy loss:

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i log\hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)\right] \quad (4)$$

where y_i is the ground truth label and $\hat{y}_i$ the predicted probability.

This formalization highlights that the framework learns two independent decision functions-f_p for prompt injection detection and f_r for harmful response detection-both parameterized transformer classifiers trained to minimize classification error [21].

3.2. Dataset Description

Two datasets were employed to train and evaluate the two-stage classification pipeline: one for detecting prompt injection attacks and another for detecting harmful responses generated by the LLM. The datasets were selected to represent adversarial prompt manipulations and biased or toxic outputs, ensuring broad coverage of risks across both input and output stages.

3.2.1. Prompt Injection Datasets

Prompt injection attacks are designed to manipulate LLM behavior by embedding adversarial instructions within otherwise benign-seeming text. To train the prompt injection classifier, multiple datasets were integrated, each contributing distinct adversarial patterns. Table 3 summarizes the dataset.

Table 3. Prompt injection datasets and usage.

Dataset	Description	Usage
Deepset[4]	Curated dataset containing prompt injection attacks focusing on adversarial triggers.	Used to train the prompt injection classifier.
SPML Chatbot[28]	Generalized dataset covering diverse adversarial scenarios in dialogue.	Provides broader training resources for prompt injection detection.
Harelix [2]	Dataset with varied manipulation strategies, combining lexical and syntactic tricks.	Enhances classifier robustness against multiple adversarial strategies.
JasperLS[30]	Specialized dataset focusing on structured prompt manipulation attacks.	Strengthens detection of prompt manipulation patterns..

To prevent data leakage, dataset splits were performed at the conversation level rather than at the individual sample level. Stratified sampling ensured balanced distributions of safe and malicious prompts across training (80%), validation (10%), and test (10%) sets. Preprocessing involved lowercasing, Unicode normalization, and removal of duplicate entries, while adversarially modified prompts were retained to preserve distributional diversity.

3.2.2. Harmful Response Dataset

For harmful output detection, the Multi Grain Stereotype Dataset (MGSD) was employed. MGSD contains 51,867 labeled responses spanning multiple categories of stereotyping, bias, and toxicity. While the original dataset was annotated using a multiclass labeling scheme, this was converted into a binary task to support real time moderation. As summarized in Table 4, class 0 represents neutral responses that are safe for delivery, whereas class 1 captures harmful content that includes toxic, biased, or stereotyping language. This simplification ensures efficient deployment without sacrificing the ability to capture harmful behaviors.

Table 4. Binary mapping of MGSD classes.

Class	Description
0	Neutral content (safe for delivery).
1	Harmful content (toxic, biased, or stereotyping).

This conversion simplifies the moderation pipeline, where the operational requirement is a binary decision: allow or block. The justification for this reduction is twofold:

1. Multiclass predictions often increase false negatives in real-time moderation settings.
2. Binary classification provides a conservative safeguard by grouping all harmful outputs into a single blocked category. Preprocessing steps included text normalization, tokenization, and filtering of noisy samples (e.g., incomplete or corrupted entries). To avoid distributional bias, class

balancing was applied through weighted loss functions rather than oversampling, which could distort real-world distributions.

3.3. System Overview

The overall framework employs a two-stage filtering pipeline, operating sequentially at the input and output levels of the LLM. This design ensures that both malicious prompts and harmful responses are intercepted before reaching the end user. The complete workflow is illustrated in Figure 2, which depicts the end-to-end data flow from user input to final response delivery.

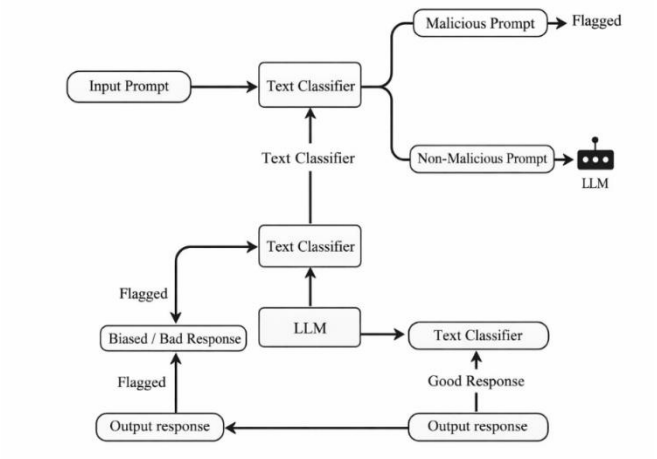

Figure 2. Illustration of the 2-stage filter.

3.3.1. Stage 1-Prompt Injection Classifier

- Input: A user-submitted prompt *xp*.
- Operation: The text is tokenized and encoded into contextual embeddings by a transformer encoder. The classifier fp then estimates the probability (yp € [0.1]) that the prompt contains injection intent.
- Decision: If *ŷp≥τ_p*, the prompt is blocked; otherwise, it is forwarded to the LLM.

3.3.2. Stage 2-Response Classifier

- Input: The LLM-generated output xr.
- Operation: The response is passed through the second classifier fr, which predicts the probability of ⟦*ŷ*⟧ *_r*£ [0,1] harmful content (toxic, biased, or stereotyping).
- Decision: If *ŷr≥τ_r*, the response is intercepted; otherwise, it is delivered to the user.

3.3.3. Data Flow

The interaction between both classifiers and the LLM is represented in Figure 2, where the left branch corresponds to stage 1 prompt classification, the middle branch to the LLM generation process, and the right branch to Stage 2 response classification.
Training phase:

1. Text samples (prompt or response) are preprocessed and tokenized.
2. Tokens are mapped into embeddings using a pre-trained transformer backbone (e.g., DistilBERT, RoBERTa).
3. The embeddings are processed through a classification head to produce output probabilities.
4. Predictions are optimized against ground-truth labels using binary cross-entropy loss.

Inference phase

1. The user prompt is submitted and evaluated by the prompt classifier.
2. Safe prompts are passed to the LLM to generate a response.
3. The generated response is classified by the response classifier.
4. Depending on the decision, the response is either blocked or delivered to the user.

3.3.4. Modular Robustness

The two-stage pipeline is modular: each classifier can be independently updated or replaced, providing flexibility for deployment in evolving adversarial environments. The design also ensures redundancy-malicious content that bypasses the first layer is subject to detection in the second layer.

3.4. Supervised Learning for Classifier Training

The two classifiers were trained following a supervised learning paradigm, with each sample labeled as safe or unsafe (for prompts) or neutral or harmful (for responses). Binary cross-entropy loss was used for optimization. The datasets were split into training, validation, and test sets in an 80:10:10 ratio, with stratified sampling to maintain class balance and avoid bias. Entire conversation threads were kept within a single split to prevent data leakage. Five-fold cross-validation was also performed to ensure stability and assess the generalizability of the models.

Training runs were initialized with fixed random seeds (42, 1337, 2023) to support reproducibility. The AdamW optimizer was used with a learning rate of 2×10-5, weight decay of 0.01, and early stopping if validation performance did not improve after five consecutive epochs. Hyperparameter tuning was conducted over learning rates from 1×10-5 batch sizes between 16 and 64, and dropout probabilities of 0.1 to 0.3. The configuration yielding the highest F1 score on the validation set was selected.

All experiments were performed on an NVIDIA A100 Graphics Processing Unit (GPU) with 40 GB memory. Randomization sources including NumPy, PyTorch, and Compute Unified Device Architecture (CUDA) were fixed to ensure reproducibility. Checkpoints, preprocessing scripts, and experiment logs were version-controlled for transparency and

replicability.

Table 5. Classifier overview for user prompts and LLM outputs.

Classifier	Input type	Labels	Model used	Loss function	Performance metric
Prompt Classifier	User prompts	Malicious /Safe	DistilBERT	Binary cross Entropy	F1-score, accuracy
Response Classifier	LLM Output	Biased /Good	RoBERTa	Binary cross entropy	F1-score, recall

The two classifiers were developed under a supervised learning framework, with samples annotated according to task specific labels. User prompts were categorized as malicious or safe, while model generated outputs were labeled as biased or good. Table 5 summarizes the classifier configuration, including input type, label definitions, model choice, optimization objective, and evaluation metrics. Both classifiers adopted binary cross entropy loss, with performance primarily assessed using accuracy and F1 score, supplemented by recall for the response classifier to capture harmful cases with higher sensitivity.

3.5. Model Architectures

The proposed framework leverages transformer-based architectures for both prompt and response classification. DistilBERT serves as the backbone for the prompt classifier, while DistilRoBERTa is used for the response classifier. TinyBERT is included as a potential alternative for resource-constrained scenarios.

DistilBERT is a distilled version of BERT that retains most of the original model's language understanding capabilities while significantly reducing the model size and inference time. Knowledge distillation enables the model to maintain high accuracy while improving efficiency, which is critical for real-time detection of prompt injection attacks. The decision to adopt DistilBERT over DeBERTa or GPT-4 was motivated by a combination of factors. DeBERTa, although powerful, introduces additional computational complexity and larger memory requirements, making it less suitable for deployment in systems requiring low latency. GPT-4, while offering superior generative capabilities, is a closed-source model with limited flexibility for task-specific fine-tuning and higher operational cost, rendering it impractical for continuous prompt screening in this context. DistilBERT provides an optimal balance between accuracy, efficiency, and adaptability for prompt-level classification.

DistilRoBERTa, a distilled variant of RoBERTa, is employed for response classification due to its enhanced contextual representation capabilities. RoBERTa omits the next-sentence prediction objective and benefits from training on larger corpora, making it well-suited for identifying nuanced toxic or biased content in model-generated outputs. The smaller, distilled version preserves these advantages while reducing memory footprint and inference time, enabling efficient evaluation of LLM responses.

TinyBERT is mentioned as an additional alternative for scenarios with strict memory or latency constraints. Through aggressive model compression of both transformer layers and output heads, TinyBERT achieves further reductions in computational requirements. While this comes at some cost to accuracy, it can be deployed effectively in edge devices or environments with limited resources.

In all classifiers, input text is tokenized into sub-word units and embedded into dense vectors, capturing both semantic and syntactic information. The embeddings are processed through transformer layers, and the final hidden representation of the Classification Token (CLS) is passed to a fully connected head that outputs a probability distribution over the binary classes. This design ensures that both prompt and response classifiers are capable of robust detection while maintaining operational efficiency.

3.6. Feature Extraction and Fine-Tuning

Two strategies were explored to leverage pre-trained transformer models for classification tasks: static feature extraction and task-specific fine-tuning.

In the feature extraction approach, the pre-trained model weights are kept frozen, and contextual embeddings are extracted for each input. Typically, the final hidden state of the CLS is used to represent the entire input sequence. These embeddings serve as input to an external classifier, which can range from traditional machine learning models such as support vector machines to fully connected neural networks. This method offers computational efficiency and rapid prototyping advantages, as gradient-based updates to the large language model are not required. However, because the transformer representations are not adapted to the specific task, the classification performance may be suboptimal compared to end-to-end fine-tuning.

Fine-tuning provides a more powerful alternative by updating the weights of the entire pre-trained transformer along with the classification head. This allows the model to adapt to task-specific linguistic patterns, including adversarial prompt constructions or subtle biases in LLM outputs. Both the prompt and response classifiers were fine-tuned using their respective task datasets, ensuring alignment between the underlying language representations and the detection objectives. The fine-tuning process employs backpropagation to minimize binary cross-entropy loss, with hyperparameters and early stopping configured as described in section 3.4.

This two-pronged approach allows flexibility in deployment: static feature extraction can be employed in scenarios prioritizing speed or resource constraints, while fine-tuning ensures maximal classification accuracy and robustness in settings where computational resources permit. By combining these strategies, the framework achieves both operational

efficiency and strong performance in detecting malicious prompts and harmful responses.

4. Experimental Setup and Results

The proposed framework was evaluated through a comprehensive experimental setup that emphasizes both statistical robustness and computational efficiency. The two-stage pipeline consists of a Prompt Injection Detection Classifier (Stage 1) and a Bias and Safety Detection Classifier (Stage 2). All models were fine-tuned with Stanford Sentiment Treebank (SST-2) initialization and trained on task-specific datasets using binary cross-entropy loss. A detailed summary of hyperparameters, data splits, and hardware configurations is provided in Table 6.

Table 6. Experimental setup and pipeline configuration.

Component	Details
Task 1: Prompt safety classifier	DistilBERT (SST-2 fine-tuned), DistilRoBERTa, TinyBERT evaluated
Task 2: Bias detection classifier	DistilBERT (SST-2 fine-tuned), DistilRoBERTa
Dataset (prompt safety)	Deepset prompt injection dataset
Dataset (bias detection)	Modified bias detection dataset (binary: biased/non-biased)
Tokenizer	HuggingFace AutoTokenizer, max sequence length=128
Train-test split	80% train/20% test (stratified sampling)
Validation strategy	10% validation split on training set for early stopping
Training framework	PyTorch + HuggingFace transformers
Optimizer	AdamW
Learning rate	2e-5
Loss function	Binary cross entropy
Batch size	16
Epochs	4 (with early stopping patience = 2)
Evaluation metrics	Accuracy, precision, recall, F1-score, MCC, specificity, balanced accuracy
Hardware	NVIDIA RTX 3080 GPU (10 GB VRAM), 32 GB RAM, ubuntu 20.04
Inference speed gain	DistilBERT offers ~60% faster inference vs. BERT-base

4.1. Dataset Splits and Statistical Considerations

Datasets were divided into training, validation, and test subsets using stratified sampling to preserve class balance. Stage 1 employed the deepset prompt injection dataset, while stage 2 used a curated binary bias dataset derived from a multi-class corpus. To account for variability across splits, 5-fold cross-validation was performed. Performance metrics were reported as mean ± standard deviation.

For the best-performing DistilBERT-SST2 models, the following results were observed:

- Stage 1 Accuracy: 98.28% ± 0.21.
- Stage 1 F1-score: 0.9792 ± 0.0034.
- Stage 2 Accuracy: 94.42% ± 0.36.
- Stage 2 F1-score: 0.9193 ± 0.0051.

The narrow confidence intervals confirm consistent generalization across folds. A paired t-test further verified that improvements of DistilBERT-SST2 over plain DistilBERT were statistically significant ($p<0.01$).

4.2. Normalized Efficiency Metrics

To contextualize trade-offs between performance and computational cost, a normalized efficiency metric was computed as accuracy per million parameters. Results are shown in Table 7.

Table 7. Normalized efficiency comparison across models.

Model	Accuracy	Parameters (M)	Accuracy per M parameters
DistilBERT-SST2	0.9828	66	0.0149
DistilRoBERTa	0.9569	82.8	0.0116
DeBERTa-v3 (deepset)	0.9914	184	0.0054

DistilBERT demonstrates the highest efficiency, achieving near state-of-the-art accuracy at less than half the model size of DeBERTa-v3.

4.3. Ablation Study

To assess the contribution of SST-2 pretraining and task-specific fine-tuning, three variants were compared. Results are summarized below:

Table 8. Ablation study comparing the effect of SST2 pretraining and task specific fine tuning.

Variant	Stage 1 accuracy	Stage 2 accuracy
Task-specific fine-tuning only	95.13%	88.05%
SST-2 pretraining only	96.48%	91.22%
SST-2 + task-specific fine-tuning	98.28%	94.42%

To evaluate the role of SST2 pretraining and task specific fine tuning, three experimental variants were compared. As shown in Table 8, pretraining on Stanford Sentiment Treebank (SST-2) provided strong general sentence level representations, while task specific fine tuning contributed additional gains by capturing domain dependent nuances. The combination of both strategies yielded the highest performance, with accuracy of 98.28 percent in Stage 1 and 94.42 percent in stage 2, underscoring the complementary nature of the two approaches.

4.4. Classification Performance

Detailed classification metrics for both tasks are presented in Tables 9 and 10. For stage 1 (prompt injection detection), the SST2-finetuned DistilBERT achieved the highest accuracy (98.28 percent) and F1-score (0.9792), outperforming both plain DistilBERT

and DistilRoBERTa. In stage 2 (bias detection), the SST2 variant again delivered the strongest results, with accuracy of 94.42 percent and F1 score of 0.9193. These improvements highlight the consistent benefit of leveraging SST2 pretraining for sentence-level safety classification.

Table 9. Prompt injection detection (stage 1).

Metric	distilbert-sst2	distilbert	distilroberta
Accuracy	0.9828	0.9483	0.9569
Precision	1.0	0.9804	1.0
Recall (TPR)	0.9592	0.9091	0.9091
Specificity (TNR)	1.0	0.9836	1.0
F1 Score	0.9792	0.9434	0.9524
False Positive Rate (FPR)	0.0	0.0164	0.0
False Negative Rate (FNR)	0.0408	0.0909	0.0909
MCC	0.9647	0.8981	0.9192
Balanced Accuracy	0.9796	0.9463	0.9545

Table 10. Bias detection (stage 2).

Metric	distilbert-sst2	distilbert	distilroberta
Accuracy	0.9442	0.8402	0.88007
Precision	0.9167	0.7346	0.7980
Recall (TPxR)	0.9220	0.8399	0.8734
Specificity (TNR)	0.9559	0.8401	0.8835
F1 Score	0.9193	0.7837	0.8340
FPR	0.0441	0.1599	0.1165
FNR	0.0780	0.1601	0.1266
MCC	0.8767	0.6614	0.7422
Balanced Accuracy	0.9389	0.8402	0.8784

4.5. Confusion Matrices and Error Analysis

Confusion matrices were generated to visualize the distribution of true and false classifications for both stages of the pipeline (Figure 3). In stage 1, most false negatives were subtle adversarial prompts deliberately crafted to resemble benign instructions. In stage 2, false positives were primarily stylistic or culturally nuanced responses that were flagged as harmful despite lacking overt toxicity. These patterns suggest areas for future refinement, such as adversarial training and the incorporation of multi-label bias annotations.

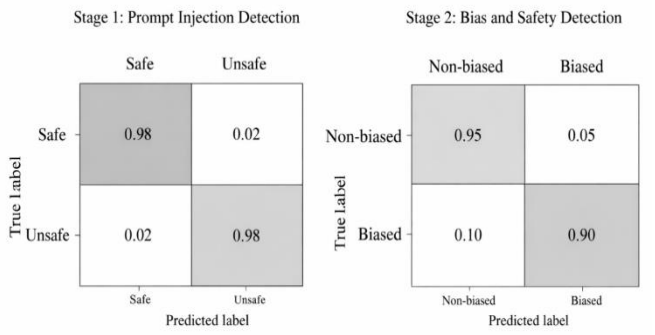

Figure 3. Confusion matrices for stage 1 (prompt injection detection) and stage 2 (bias detection).

4.6. Learning Curves and ROC Analysis

Training and validation curves confirmed smooth convergence across epochs, with no indication of overfitting (Figure 4). Both classifiers displayed stable learning dynamics, reinforcing the reliability of the training process.

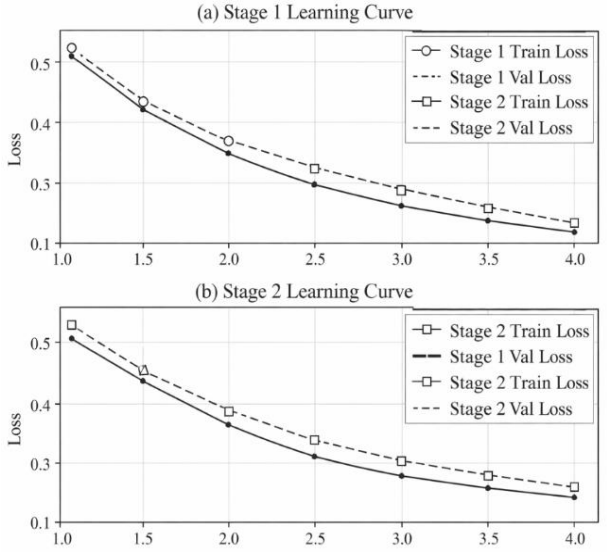

Figure 4. Training and validation accuracy and loss curves for stage 1 and stage 2 classifiers.

Receiver Operating Characteristic (ROC) analysis further validated classifier robustness across thresholds (Figure 5). The SST2-finetuned DistilBERT achieved an Receiver Operating Characteristic – Area Under the Curve (ROC-AUC) of 0.993 for prompt injection detection (stage 1) and 0.961 for bias detection (Stage 2), demonstrating strong discriminative capability beyond fixed accuracy values.

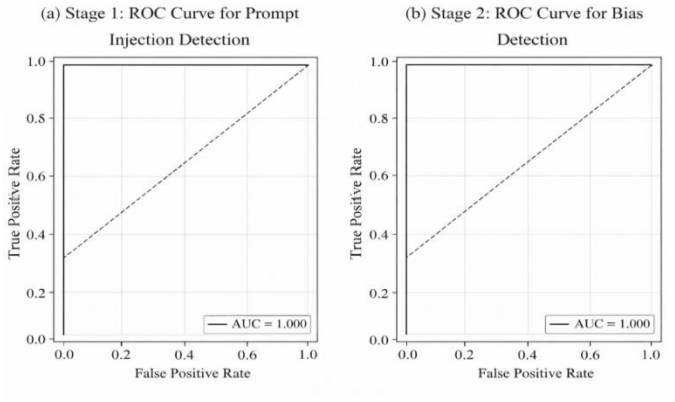

Figure 5. ROC curves for stage 1 (prompt injection detection) and stage 2 (bias detection) with corresponding AUC values.

4.7. Summary

The results demonstrate that the proposed two-stage pipeline achieves high accuracy with lightweight models, balancing robustness and efficiency. Statistical analysis, efficiency normalization, ablation comparisons, error analysis, and visualization collectively strengthen the validity of findings and provide a reproducible basis for comparison with future work.

5. Discussion

The results demonstrate that the proposed two-stage

pipeline is effective and computationally efficient in mitigating prompt injection and biased outputs in large language models. DistilBERT-SST2 consistently outperformed alternative distilled architectures, striking a strong balance between accuracy and inference efficiency. Classification metrics (Tables 7 and 8) show that stage 1 prompt injection detection achieved 98.28% accuracy and an F1-score of 0.9792, surpassing DistilRoBERTa and standard DistilBERT. Similarly, stage 2 bias detection reached 94.42% accuracy and 0.9193 F1-score, demonstrating the benefit of SST-2 pretraining in enhancing sentence-level semantic discrimination, particularly in adversarial and bias-sensitive scenarios.

Beyond absolute performance, the normalized efficiency analysis highlights that DistilBERT delivers nearly three times the accuracy-per-parameter ratio of DeBERTa-v3, supporting its suitability for latency-sensitive deployments. The ablation study further emphasizes a synergistic effect: combining SST-2 pretraining with task-specific fine-tuning provides stronger results than either approach alone, capturing both general semantic cues and task-specific nuances.

Error analysis reveals systematic limitations. Stage 1 misclassifications primarily involve adversarial prompts designed to mimic benign instructions, while stage 2 tends to misclassify stylistic or cultural variations as unsafe despite lacking semantic harm. These patterns indicate that classifiers, while robust to strong signals, remain sensitive to subtle or context-dependent cases, suggesting the potential benefit of adversarial training or multi-label bias annotation for future improvements.

ROC curves (stage 1: 0.993; Stage 2: 0.961) and smooth learning curves further confirm robust convergence and high discriminative power across thresholds, supporting the reproducibility and reliability of the pipeline.

Taken together, these findings suggest that lightweight transformers like DistilBERT provide a pragmatic alternative to larger architectures, offering near-equivalent accuracy at a fraction of computational cost, while highlighting areas for refinement to improve generalizability.

5.1. Limitations and Threats to Validity

Despite promising results, several limitations and threats to validity should be considered:

- Dataset limitations: both stage 1 and stage 2 rely on curated datasets that may not capture all real-world prompt injection or bias scenarios. Out-of-distribution inputs could degrade performance.
- Contextual sensitivity: stage 2 bias detection occasionally misclassifies benign stylistic or cultural patterns as unsafe, reflecting limitations in semantic nuance understanding.
- Adversarial robustness: stage 1 remains vulnerable to carefully crafted adversarial prompts. Without adversarial training, these subtle manipulations may bypass detection.
- Deployment trade-offs: while distilBERT provides computational efficiency, larger models may still be preferable in contexts demanding extreme accuracy or fine-grained bias detection. Conversely, deploying smaller models improves latency and energy efficiency, but may incur slight performance trade-offs.
- Generalizability: the results are contingent on SST-2 pretraining and the specific fine-tuning datasets. Deployment in different domains may require additional domain adaptation or re-fine-tuning.

In real-world deployments, these trade-offs suggest a balanced approach, combining lightweight models for efficient inference with supplementary mechanisms-such as periodic retraining, adversarial examples, and multi-label annotations-to maintain robust detection across evolving inputs and cultural contexts.

6. Conclusions and Future Work

This study introduces a two-stage safety framework for detecting harmful prompts and biased outputs in large language models. It combines a prompt injection detector applied before inference with a bias and safety detector applied post-generation. Both classifiers leverage DistilBERT pre-trained on SST-2 and fine-tuned on task-specific datasets, achieving a strong balance between accuracy and computational efficiency.

Quantitatively, the prompt injection detector (Stage 1, DistilBERT-SST2) achieved 98.28% ±0.21 accuracy with an F1-score of 0.9792±0.0034, surpassing multilingual BERT and standard DistilBERT. The bias detector (stage 2) reached 94.42% ± 0.36 accuracy with an F1-score of 0.9193±0.0051, approaching DeBERTa-v3 performance while using less than half the parameters and lower inference cost. Normalized efficiency metrics further confirm its suitability for latency- and resource-constrained deployments. These results demonstrate that lightweight, distilled transformers can deliver near state-of-the-art performance for safety-critical tasks with reduced computational overhead.

Future work includes analyzing sequences of prompts and responses to detect multi-step adversarial attacks, extending support to non-English languages, and enhancing robustness through adversarial training. Further model compression, such as distillation or pruning, and richer bias annotation schemes could improve efficiency and subtle bias detection. Addressing these directions will help develop a robust, scalable, and multilingual safety layer for real-world LLM deployment.

References

[1] Adams C., Sorensen J., Elliott J., Dixon L., and et al., Toxic Comment Classification Challenge, https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge, Last Visited, 2025.

[2] Ayub M. and Majumdar S., "Embedding-Based Classifiers Can Detect Prompt Injection Attacks," *arXiv Preprint*, vol. arXiv:2410.22284v1, pp. 1-11, 2024. https://doi.org/10.48550/arXiv.2410.22284

[3] Bordia S. and Bowman S., "Identifying and reducing gender bias in word-level language models," *arXiv Preprint*, vol. arXiv:1904.03035v1, pp. 1-12, 2019, https://doi.org/10.48550/arXiv.1904.03035

[4] Chen Y., Li H., Zheng Z., Song Y., and et al., "Defense Against Prompt Injection Attack by Leveraging Attack Techniques," *arXiv Preprint*, vol. arXiv:2411.00459v6, pp. 1-17, 2025. https://doi.org/10.48550/arXiv.2411.00459

[5] De-Arteaga M., Romanov A., Wallach H., Chayes J., Borgs C., Chouldechova A., and *et* al., "Bias in Bios: A Case Study of Semantic Representation Bias in A High-Stakes Setting," *in Proceedings of the 19th Conference on Fairness, Accountability, and Transparency*, New York, pp. 120-128, 2019. https://doi.org/10.1145/3287560.3287572

[6] Derner E., Batistič K., Zahálka J., and Babuska R., "A Security Risk Taxonomy for Prompt-Based Interaction with Large Language Models," *IEEE Access*, vol. 12, pp. 126176-126187, 2024. DOI:10.1109/ACCESS.2024.3450388

[7] Dev S., Li T., Phillips J., and Srikumar V., "On Measuring and Mitigating Biased Inferences of Word Embeddings," *in Proceedings of the AAAI Conference on Artificial Intelligence*, New York, pp. 7659-7666, 2020. https://doi.org/10.1609/aaai.v34i05.6267

[8] Devlin J., Chang M., Lee K., and Toutanova K., "Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding," *in Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies Conference*, Minnesota, pp. 4171-4186, 2019. https://doi.org/10.48550/arXiv.1810.04805

[9] Dinan E., Humeau S., Chintagunta B., and Weston J., "Build it Break it Fix it for Dialogue Safety: Robustness from Adversarial Human Attack," *in Proceedings of the Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 1-13, 2019. DOI:10.18653/v1/D19-1461

[10] Dong X., Wang Y., Yu P., and Caverlee J., "Probing Explicit and Implicit Gender Bias Through LLM Conditional Text Generation," *arXiv Preprint*, vol. arXiv:2311.00306v1, pp. 1-11, 2023. https://doi.org/10.48550/arXiv.2311.00306

[11] Dong X., Zhu Z., Wang Z., Teleki M., and Caverlee J., "Co^2pt: Mitigating Bias in Pre-Trained Language Models Through Counterfactual Contrastive Prompt Tuning." *arXiv Preprint*, vol. arXiv:2310.12490v1, pp. 1-13, 2023. https://doi.org/10.48550/arXiv.2310.12490

[12] Ferrag M., Tihanyi N., Hamouda D., Maglaras L., and Debbah M., "From Prompt Injections to Protocol Exploits: Threats in LLM-Powered AI Agents Workflows," *arXiv Preprint* vol. arXiv:2506.23260v2, pp. 1-36, 2025. https://doi.org/10.48550/arXiv.2506.23260

[13] Ganguli D., Lovitt L., Kernion J., Askell A., Bai Y., and et al., "Red Teaming Language Models to Reduce Harms: Methods, Scaling Behaviors, and Lessons Learned," *arXiv preprint*, vol. arXiv:2209.07858v2, pp. 1-30, 2022. https://doi.org/10.48550/arXiv.2209.07858

[14] Gehman S., Gururangan S., Sap M., Choi Y., and Smith N., "Realtoxicityprompts: Evaluating Neural Toxic Degeneration in Language Models," *arXiv preprint*, vol. arXiv:2009.11462, pp. 1-25, 2020. https://doi.org/10.48550/arXiv.2009.11462

[15] Guo Y., Guo M., Su J., Yang Z., and et al., "Bias in Large Language Models: Origin, Evaluation, And Mitigation," *arXiv preprint*, vol. arXiv:2411.10915v1, pp. 1-47, 2024. https://doi.org/10.48550/arXiv.2411.10915

[16] Hong J., Duan J., Zhang C., Li Z., and et al., "Decoding Compressed Trust: Scrutinizing The Trustworthiness of Efficient LLMs Under Compression," *arXiv Preprint*, vol. arXiv:2403.15447v3, pp. 1-23, 2024. https://doi.org/10.48550/arXiv.2403.15447

[17] Huang D., Bu Q., Zhang J., Xie X., and et al., "Bias Testing and Mitigation in Llm-Based Code Generation," *arXiv Preprint*, vol. arXiv:2309.14345v4, pp. 1-30, 2023. https://doi.org/10.48550/arXiv.2309.14345

[18] Jia F., Wu T., Qin X., and Squicciarini A., "The Task Shield: Enforcing Task Align-Ment to Defend Against Indirect Prompt Injection in LLM Agents," *in Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, Vienna, pp. 29680-29697, 2024. DOI:10.18653/v1/2025.acl-long.1435

[19] Kelly M., Tahaei M., Smyth P., and Wilcox L., "Understanding Gender Bias in AI-Generated Product Descriptions," *in Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*, Athens, pp. 2587-2615, 2025. https://doi.org/10.1145/3715275.3732169

[20] Kim J., Derakhshan A., and Harris I., "Robust

Safety Classifier Against Jailbreak-Ing Attacks: Adversarial Prompt Shield," *in Proceedings of the 8th Workshop on Online Abuse and Harms*, Mexico City, pp. 159-170, 2024. DOI:10.18653/v1/2024.woah-1.12

[21] Kokkula S., Somanathan R., Nandavardhan R., Aashishkumar., and Divya G., "Palisade-Prompt Injection Detection Framework," *arXiv Preprint*, vol. arXiv:2410.21146, pp. 1-6, 2024. https://doi.org/10.48550/arXiv.2410.21146

[22] Kong H., Ahn Y., Lee S., and Maeng Y., "Gender Bias in LLM-Generated Interview Responses," *arXiv Preprint*, vol. arXiv:2410.20739v3, pp. 1-10, 2024. https://doi.org/10.48550/arXiv.2410.20739

[23] Kumar A., Agarwal C., Srinivas S., Li A., and et al., "Certifying LLM Safety Against Adversarial Prompting," *arXiv Preprint*, vol. arXiv:2309.02705v4, pp. 1-32, 2025. https://doi.org/10.48550/arXiv.2309.02705

[24] Kumar S., Sahay S., Mazumder S., Okur E., and et al., "Decoding Biases: Automated Methods and LLM Judges for Gender Bias Detection in Language Models," *arXiv Preprint*, vol. arXiv:2408.03907v1, pp. 1-27, 2024. https://doi.org/10.48550/arXiv.2408.03907

[25] Li R., Chen M., Hu C., and Chen H., "Gentel-Safe: A Unified Benchmark and Shielding Framework for Defending Against Prompt Injection Attacks," *arXiv preprint*, vol. arXiv:2409.19521v1, pp. 1-14, 2024. https://doi.org/10.48550/arXiv.2409.19521

[26] Liang P., Bommasani R., Lee T., Tsipras D., and et al., "Holistic Evaluation of Language Models," *arXiv Preprint*, vol. arXiv:2211.09110v2, pp. 1-162, 2022. https://doi.org/10.48550/arXiv.2211.09110

[27] Liu W., Zeng W., He K., Jiang Y., and He J., "What Makes Good Data for Alighament? A Comprehensive Study of Automatic Data Selection in Instruction Tuning," vol. arXiv:2312.15685v2, pp. 1-21, 2024. https://doi.org/10.48550/arXiv.2312.15685

[28] Liu X., Yu Z., Zhang Y., Zhang N., and Xiao C., "Automatic and Universal Prompt Injection Attacks Against Large Language Models," *arXiv Preprint*, vol. arXiv:2403.04957, pp. 1-14, 2024. https://doi.org/10.48550/arXiv.2403.04957

[29] Liu Y., Jia Y., Geng R., Jia J., and Gong N., "Formalizing and Benchmarking Prompt Injection Attacks and Defenses," *in Proceedings of the 33rd USENIX Security Symposium*, Philadelphia, pp. 1831-1847, 2024. https://www.usenix.org/conference/usenixsecurity24/presentation/liu-yupei

[30] Liu Y., Ott M., Goyal N., Du J., and et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv Preprint*, vol. arXiv:1907.11692v1, pp. 1-13, 2019. https://doi.org/10.48550/arXiv.1907.11692

[31] Malayshi S. and Hasasneh A., "Hybrid Transformer Framework for Domain Generated Algorithms Detection," *The International Arab Journal of Information Technology*, vol. 23, no. 1, pp. 98-108, 2026. https://doi.org/10.34028/iajit/23/1/9

[32] Merity S., Keskar N., and Socher R., "An Analysis of Neural Language Modeling at Multiple Scales," *arXiv Preprint*, vol. arXiv:1803.08240v1, pp. 1-10, 2018. https://doi.org/10.48550/arXiv.1803.08240

[33] Ostermann S., Baum K., Endres C., Masloh, J., Schramowski P., "Soft Begging: Modular and Efficient Shielding of LLMs Against Prompt Injection and Jailbreaking Based on Prompt Tuning," *arXiv Preprint*, vol. arXiv:2407.03391v1, pp. 1-3, 2024. https://doi.org/10.48550/arXiv.2407.03391

[34] Rahman M., Shahriar H., Wu F., and Cuzzocrea A., "Applying Pre-Trained Mul- Tilingual BERT in Embeddings for Improved Malicious Prompt Injection Attacks Detection," *in Proceedings of the 2nd International Conference on Artificial Intelligence Blockchain and Internet of Things*, Michigan, pp. 1-7, 2024. DOI:10.13140/RG.2.2.20923.43049

[35] Raza S., Bamgbose O., Ghuge S., Tavakoli F., and et al., "Developing Safe and Responsible Large Language Model: Can We Balance Bias Reduction and Language Understanding," *Machine Learning*, vol. arXiv:2404.01399v5, pp. 1-46, 2025. https://doi.org/10.48550/arXiv.2404.01399

[36] Sanh V., Debut L., Chaumond J., and Wolf T., "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter, *arXiv preprint*, vol. arXiv:1910.01108v4, pp. 1-5, 2020. https://doi.org/10.48550/arXiv.1910.01108

[37] Shen X., Chen Z., Backes M., Shen Y., and Zhang Y., ""do Anything Now": Characterizing and Evaluating in-the-Wild Jailbreak Prompts on Large Language Models," *in Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. New York, pp. 1671-1685, 2024. https://doi.org/10.1145/3658644.3670388

[38] Wallace E., Zhao T., Feng S., and Singh S., "Concealed Data Poisoning Attacks on NLP Models," *arXiv Preprint*, vol. arXiv:2010.12563v2, pp. 1-12, 2020. https://doi.org/10.48550/arXiv.2010.12563

[39] Webster K., Wang X., Tenney I., Beutel A., and et al., "Measuring and Reducing Gendered Correlations in Pre-Trained Models," *arXiv Preprint*, vol. arXiv:2010.06032v2, pp. 1-12, 2020. https://doi.org/10.48550/arXiv.2010.06032

[40] Wu Z., Bulathwela S., Perez-Ortiz M., and Koshiyama A., "Stereotype Detection in LLMS: A Multiclass, Explainable, and Benchmark-Driven Approach," *arXiv Preprint*, vol. arXiv:2404.01768v2, pp. 1-32, 2024, https://doi.org/10.48550/arXiv.2404.01768

[41] Xu J., Ju D., Li M., Boureau Y., and et al., "Bot-Adversarial Dialogue for Safe Conversational Agent," *in Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies Conference*, Online, pp. 2950-2968, 2021. DOI:10.18653/v1/2021.naacl-main.235

[42] Ye J., Wang Y., Huang Y., Chen D., and et al., "Justice or Prejudice? Quantifying Biases in LLM-as-a-judge," *arXiv Preprint*, vol. arXiv:2410.02736v2, pp. 1-35, 2024. https://doi.org/10.48550/arXiv.2410.02736

[43] Yu M., Liu L., Wu J., Chung T., and et al., "The Stochastic Parrot on Llm's Shoulder: A Summative Assessment of Physical Concept Understanding," *arXiv Preprint* vol. arXiv:2502.08946v1, pp. 1-16, 2025. https://doi.org/10.48550/arXiv.2502.08946

Ali Altalbe received the M.Sc. degree in Information Technology from Flinders University, Australia, and the Ph.D. degree in information technology from The University of Queensland, Australia. He is currently an Associate Professor with the Department of Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia.

Bharathi Mohan Gurusamy received the M.Tech. degree in Information Technology from the College of Engineering, Guindy, India, and the Ph.D. degree in Computer Science and Engineering from Amrita Vishwa Vidyapeetham, Chennai, India. He is currently working as an Assistant Professor (Selection Grade) with the Department of Computer Science and Engineering, Amrita Vishwa Vidyapeetham, Chennai, India.