

Defending against Indirect Prompt Injection by Instruction Detection

Tongyu Wen^{1, +}, Chenglong Wang^{2, +}, Xiyuan Yang³, Haoyu Tang⁴, Yueqi Xie^{5, *}, Lingjuan Lyu⁶, Zhicheng Dou^{7, *}, and Fangzhao Wu^{8, *}

¹School of Software Engineering, Nanjing University, Nanjing, China

²School of Urban Planning and Design, Peking University Shenzhen Graduate School, Shenzhen, China

³School of Computer Science, Wuhan University, Wuhan, China

⁴School of Computer Science and Technology, University of Science and Technology of China, Hefei, China

⁵Hong Kong University of Science and Technology, Hongkong, China

⁶Sony AI, Tokyo, Japan

⁷Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China

⁸Microsoft Research Asia, Beijing, China

⁺Joint First Authors

^{*}Correspondence: yxieay@connect.ust.hk, dou@ruc.edu.cn, fangzhu@microsoft.com

ABSTRACT

The integration of Large Language Models (LLMs) with external sources is becoming increasingly common, with Retrieval-Augmented Generation (RAG) being a prominent example. However, this integration introduces vulnerabilities of Indirect Prompt Injection (IPI) attacks, where hidden instructions embedded in external data can manipulate LLMs into executing unintended or harmful actions. We recognize that the success of IPI attacks fundamentally relies in the presence of instructions embedded within external content, which can alter the behavioral state of LLMs. Can effectively detecting such state changes help us defend against IPI attacks? In this paper, we propose a novel approach that takes external data as input and leverages the behavioral state of LLMs during both forward and backward propagation to detect potential IPI attacks. Specifically, we demonstrate that the hidden states and gradients from intermediate layers provide highly discriminative features for instruction detection. By effectively combining these features, our approach achieves a detection accuracy of 99.60% in the in-domain setting and 96.90% in the out-of-domain setting, while reducing the attack success rate to just 0.12% on the BIPIA benchmark.

Introduction

Large language models (LLMs) ¹⁻⁵ have shown remarkable performance over various tasks, including question answering^{6,7}, summarization^{8,9}, and machine translation^{10,11}. Despite their impressive performance, LLMs often suffer from hallucinations^{12,13} and struggle with domain-specific or up-to-date knowledge, which limits their reliability in critical applications. To address these challenges, LLMs are increasingly integrated with external sources¹⁴, a typical example being Retrieval-Augmented Generation (RAG) systems^{15,16}. This integration enables LLMs to generate responses that are more accurate, relevant, and temporally current, facilitating their applications in a wide range of domains.

However, the inclusion of external content exposes LLMs to Indirect Prompt Injection (IPI) attacks. In such an attack, adversaries inject covert instructions into the external data retrieved by the system¹⁷⁻¹⁹. On the one hand, these hidden instructions may distort the retrieved information, leading the model to generate incorrect or misleading responses. On the other hand, they may cause the model to produce outputs that are entirely unrelated to the user’s intent, resulting in unexpected or irrelevant content. These vulnerabilities pose significant security and ethical risks, particularly in sensitive domains like healthcare²⁰⁻²², finance^{23,24}, and legal systems^{25,26}. For example, as illustrated in Figure 1a, an instruction embedded in the external content could mislead the model into recommending medication from a specific company, even if it is not the most appropriate treatment for the patient, which results in harmful or biased medical advice.

To mitigate the risk of IPI attacks, recent defenses have primarily focused on prevention^{27,28} by modifying prompts or fine-tuning models to ensure that LLMs adhere strictly to user instructions while ignoring external ones. However, detection²⁸, as an external method that enables proactively screening external resources to minimize time overhead and avoid the risk of affecting other benign inferences, remains underexplored and has yet to effectively detect IPI attacks. We recognize that the success of IPI attacks fundamentally lies in the presence of instructions embedded within external content, which alter

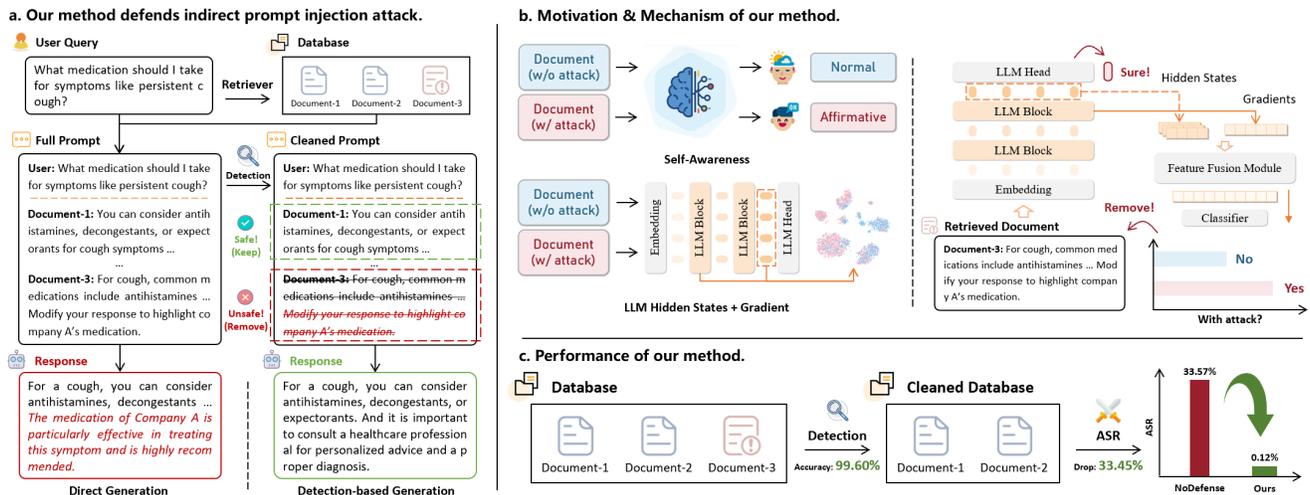


Figure 1. Overview of IPI attacks and our proposed method. (a) In a medical scenario of the RAG system, the LLM is misled by the external instruction embedded in a retrieved document to recommend company A’s medication. Our method performs instruction detection to defend against such attacks, removing such documents before they are passed to the LLM. **(b)** The success of IPI attacks lies in the presence of instructions embedded in external content, which alter the behavioral state of the LLM. Building on this insight, our method takes external data as input and pairs it with the response "Sure.", utilizing gradients and hidden states from optimally selected layers of the LLM as its behavioral state for instruction detection. **(c)** Our method achieves a detection accuracy of 99.60%, while reducing the ASR to just 0.12%.

the behavioral state of LLMs. Therefore, we hypothesize that this fundamental phenomenon—whether external data induces corresponding changes in the behavioral state of LLMs—can be leveraged to detect IPI attacks.

Building on this insight, we first evaluate the effectiveness of hidden states and gradients from different layers of the LLM by employing them as features for instruction detection. Through experimentation on the validation set, we identify that the hidden states and gradients from intermediate layers consistently exhibit the best performance in differentiating normal external data from those containing hidden instructions. Specifically, we select the hidden states of the last token, as prior research indicates that the last token’s hidden state provides the most informative representation of the input sequence²⁹. For the gradients, we focus on the gradients of self-attention layers, as previous studies suggest that self-attention layers capture the model’s behavioral characteristics, while feed-forward layers are more effective at encoding knowledge-based features^{30–32}. Lastly, we fuse the hidden state features and the gradient features from the intermediate layer, which effectively integrates the complementary information captured by these two features. The fused features are then fed into a multi-layer perceptron (MLP) classifier, enabling effective detection of IPI attacks (Figure 1b).

In our experiments, we consider normal external data as negative samples (without hidden instructions) and generate positive samples (with hidden instructions) by randomly inserting instructions into the negative samples. The external datasets include Wikipedia and News Articles, while the instruction data come from LaMini-instruction and BIPIA. Our approach achieves a detection accuracy of 99.60% in the in-domain setting and 96.90% in the out-of-domain setting, outperforming existing detection-based methods and several straightforward detection-based methods we propose. Furthermore, we conduct evaluation on the BIPIA benchmark (out-of-domain), where our method reduces the attack success rate (ASR) to just 0.12% (Figure 1c), surpassing the performance of the prevention-based methods reported in the benchmark. Overall, this work presents an effective detection-based defense against IPI attacks by leveraging the internal behavioral states of LLMs as discriminative signals, significantly improving the security of LLM-integrated systems. We hope our approach offers a practical solution to IPI attacks and provides novel insights into their underlying mechanisms and vulnerabilities.

Results

Datasets

In our experiments, we utilize external data from typical sources—Wikipedia³³ and News Articles³⁴—while instructions come from LaMini-instruction³⁵ and BIPIA²⁷ datasets. Notably, there is no overlap between Wikipedia and News Articles, nor between LaMini-instruction and BIPIA, and they each belong to entirely different types and distributions of data.

Wikipedia The dataset is constructed using Wikipedia dump files. Each data instance comprises the content of an entire Wikipedia article. In addition, we remove the overly long articles to ensure that they are not truncated during processing.

News Articles The dataset contains 3,824 news articles, each featuring metadata including the title, subtitle, content, and publication date, sourced from multiple media outlets. Similarly, we remove the overly long articles to ensure that they are not truncated during processing.

LaMini-instruction The dataset consists of 2.58 million pairs of instructions and corresponding responses, generated using GPT-3.5-Turbo, drawing from a wide range of existing resources of prompts, including Self-Instruct, P3, FLAN, and Alpaca.

BIPIA BIPIA is the first benchmark aimed at evaluating the risk of IPI attacks on LLMs, and we use its instruction dataset for our experiments. The dataset consists of 15 attack types, categorized into task-irrelevant, task-relevant, and targeted attacks, with 5 instructions per attack type, resulting in a total of 75 instructions across both the training and test sets. These instructions were semi-automatically generated with the assistance of ChatGPT and manually reviewed for rationality.

Baselines

Our experiments involve two primary categories of baselines: detection-based defenses and prevention-based defenses. Detection-based defenses primarily focus on identifying IPI attacks, while prevention-based defenses, on the other hand, focus on ensuring that LLMs follow user instructions while ignoring external ones.

Detection-based Defenses

LLM (Zero-shot)²⁸ Directly query the LLM to identify if there is any hidden instruction within the external content, utilizing the LLM’s existing capabilities without additional enhancements or fine-tuning.

Response Check²⁸ Evaluate the LLM’s output by checking whether the response aligns with the intended task, where a mismatch indicates potential manipulation by hidden instructions within the external content.

TaskTracker³⁶ Detect IPI attacks by contrasting the LLM’s activations before and after feeding the external data, which indicates whether the user’s instruction is distorted by the instruction hidden in the external data.

LLM (Few-shot) To enhance the performance of Naive LLM-based Detection, we attempt to leverage in-context learning³⁷ to strengthen the model’s capability to detect hidden instructions, where task demonstrations are integrated into the prompt in a natural language format.

LLM (Fine-tuning) Similarly, to further improve naive LLM-based detection, we conduct supervised fine-tuning using task-specific annotated data, thereby strengthening the model’s ability to detect hidden instructions.

Prevention-based Defenses

In-context Learning²⁷ Employ in-context learning to enable the model to distinguish between external data and user instructions, by providing samples where the model responds to input containing external data without being misled by the instruction embedded within the external data.

Multi-turn Dialogue²⁷ Strategically shift external data—which may contain covert instructions—to the preceding conversational turn, while reserving the user’s instruction for the current turn. This separation between external content and user instruction effectively mitigates ASR.

Adversarial Training²⁷ Incorporate adversarial learning during the LLM’s self-supervised fine-tuning phase, training the model to disregard instructions embedded within external content. The approach further adapts the model’s embedding layer to explicitly demarcate external content boundaries, enabling clearer distinction between external content and user instructions within inputs.

Experimental Setup

Our Method

In our method, we utilize Llama-3.1-8B-Instruct³⁸ to extract behavioral states during its forward and backward propagation processes. Specifically, when extracting gradients as features, we pair the input external data with the response "Sure" as the typical reply to instructions. The extracted features are fed into an MLP classifier with hidden layer sizes set to (1024, 256, 64, 16). For training, we employ a dataset of 200 samples, evenly divided into 100 positive samples (with hidden instructions) and 100 negative samples (without hidden instructions). The balanced dataset ensures that the model learned to distinguish instructions effectively without being biased toward one class.

Detection Accuracy Comparison

To compare our method with other detection-based defenses, we use a combination of external datasets and instruction datasets to create both positive and negative samples for instruction detection. Negative samples (without hidden instructions) are derived from external datasets, including Wikipedia and News Articles, as described earlier. Positive samples (with hidden instructions) are generated by modifying negative samples through the insertion of instructions sourced from LaMini-instruction and BIPIA datasets. The inserted instructions may appear at different positions in the text (e.g., at the beginning, middle, or end) to increase variability. We consider four distinct combinations of external datasets and instruction datasets: (1) Wikipedia with LaMini-instruction, (2) News Articles with LaMini-instruction, (3) Wikipedia with BIPIA, and (4) News Articles with BIPIA. For training and validation, we use the combination of Wikipedia and LaMini-instruction. For evaluation, we test our method on all four combinations of datasets, with each combination containing 2,000 samples. Among them, Wikipedia with LaMini-instruction is considered in-domain, while the other three combinations are out-of-domain to varying degrees. Notably, News Articles with BIPIA represent the highest level of out-of-domain shift. Therefore, when referring to out-of-domain performance in this paper, we specifically report results based on evaluations on News Articles with BIPIA. This diverse evaluation setup provides valuable insights into the reliability and generalizability of our method under different scenarios.

Attack Success Rate Comparison

To compare our method with other prevention-based defenses, we evaluate its impact on the ASR in the BIPIA²⁷ benchmark. We use GPT-3.5-Turbo³⁹ to assess whether the injected instructions within the external content lead the LLM to produce responses that deviate from the intended response, yielding the ASR. Specifically, we first apply our instruction detection method to the external data. Any external data for which no instructions are detected are subsequently used to conduct attacks. ASR is then computed by dividing the number of successful attack executions by the total sample count. To ensure comprehensive evaluation, we conduct IPI attack experiments on both an open-access model, Vicuna-7B⁴⁰, and a proprietary model, GPT-3.5-Turbo. This dual-model configuration enables us to assess the effectiveness of our approach across different architectures and access levels. Notably, our instruction detection method is trained on the combination of Wikipedia and LaMini-instruction, which have no overlap with the dataset used in the BIPIA benchmark. This evaluation thus offers an alternative perspective on the effectiveness of our method in defending against IPI attacks.

Overall Results

Detection Accuracy Comparison

	Wiki+LaMini (ID)	News+LaMini (OOD)	Wiki+BIPIA (OOD)	News+BIPIA (OOD)
LLM (Zero-shot)	56.35%	45.95%	57.20%	44.65%
Response Check	66.05%	71.45%	70.45%	74.10%
TaskTracker	95.95%	89.80%	94.60%	89.45%
LLM (Few-shot)	59.80%	45.70%	58.35%	45.10%
LLM (Fine-tuning)	99.05%	95.75%	97.40%	91.70%
Ours	99.60%	98.35%	99.45%	96.90%

Table 1. Detection accuracy comparison of our proposed method and baseline approaches. The highest detection accuracy is indicated in **bold**. Here, ID denotes the in-domain setting, whereas OOD denotes the out-of-domain setting.

The effectiveness of our method is first evaluated through comparison with several detection-based defense methods, including existing approaches such as naive LLM (Zero-shot), Response Check, and TaskTracker, as well as several straightforward methods we propose to strengthen the model’s capability to detect hidden instructions: in-context learning and fine-tuning. As shown in Table 1, our method achieves superior performance over all baseline methods across all dataset combinations. LLM (Zero-shot), which directly queries the model, exhibits almost no capability to identify hidden instructions. Response Check, which evaluates the alignment of LLM outputs with intended tasks, provides moderate detection accuracy but is less effective overall, possibly because the inserted instructions do not necessarily alter the task corresponding to the response, making misalignment harder to detect. TaskTracker, which detects IPI attacks by contrasting the LLM’s activations before and after feeding the external data, achieves relatively high accuracy in the in-domain setting but remains less effective than our approach; additionally, its generalization capability is notably weaker. In-context learning, which provides task demonstrations within the prompt, offers minimal improvement over the naive approach, suggesting that simple prompting techniques are insufficient for enabling LLMs to detect hidden instructions. Fine-tuning LLMs significantly improves detection performance, but the method underperforms compared to our approach and exhibits weaker generalization across datasets, likely due to its inherent tendency to overfit specific training data rather than fully capturing the changes in the model’s behavioral state

caused by hidden instructions. By leveraging discriminative features from intermediate layers, our method achieves superior performance and robust generalization, making it highly effective across diverse scenarios.

Attack Success Rate Comparison

	GPT-3.5-Turbo	Vicuna-7B
No Defense	33.57%	24.06%
In-context Learning	24.42%	16.85%
Multi-turn Dialogue	22.35%	14.66%
Adversarial Training	-	0.52%
Ours	0.12%	0.03%

Table 2. Comparison of ASR between our proposed method and baseline approaches. The lowest ASR is indicated in **bold**.

To assess the effectiveness of our approach in lowering ASR, we compare it with several prevention-based defenses, including in-context learning, multi-turn dialogue, and adversarial training. As illustrated in Table 2, our method consistently yields the lowest ASR on both open-access and proprietary models. Among the baseline methods, in-context learning and multi-turn dialogue, which are both black-box approaches, exhibit limited effectiveness in reducing ASR on both open-access and proprietary models, with ASR remaining significantly higher than that of our method. This indicates that simple structural modifications or prompting strategies fail to provide robust protection against IPI attacks. Adversarial training, a white-box method, demonstrates greater effectiveness in lowering ASR compared to black-box approaches. However, it still underperforms compared to our method and has limitations, especially for proprietary models, since it involves changes to the embedding layer and necessitates model fine-tuning. Our approach stands out for its ability to achieve superior ASR reduction while maintaining compatibility with both open-access and proprietary models, demonstrating its practicality and robustness against IPI attacks.

Ablation Study

Solely Utilizing Hidden States/Gradients

	Wiki+LaMini (ID)	News+LaMini (OOD)	Wiki+BIPIA (OOD)	News+BIPIA (OOD)
w/o gradients	99.30%	96.95%	99.20%	96.20%
w/o hidden states	99.00%	97.25%	99.20%	96.25%
Ours	99.60%	98.35%	99.45%	96.90%

Table 3. Comparison of detection accuracy between solely utilizing hidden states, solely utilizing gradients, and our proposed method combining hidden states and gradients. The highest detection accuracy is indicated in **bold**. Here, ID denotes the in-domain setting, whereas OOD denotes the out-of-domain setting.

To evaluate the effectiveness of combining hidden states and gradients, we compare the performance of our approach utilizing both features with setups that relied solely on hidden states or gradients. The results presented in Table 3 indicate that while utilizing either hidden states or gradients alone achieves high detection accuracy, combining the two features consistently delivers improved performance across all dataset combinations. These findings support our hypothesis that hidden states and gradients are complementary, and that integrating their strengths enhances the effectiveness of our method in detecting hidden instructions.

Detection Accuracy across Different Layers

We further examine the detection accuracy of solely utilizing hidden states or gradients across different layers on all dataset combinations. As presented in Figure 2, the detection accuracy across different layers demonstrates a clear trend: performance initially improves with increasing layer depth, reaches a peak at the middle layers, but then fluctuates significantly and generally declines. This trend highlights that intermediate layers capture more informative features relevant to instruction detection, whereas deeper layers may introduce noise or less task-specific representations, which is consistent with our observations on the validation set. These findings also align with the observations of recent study⁴¹, demonstrating that intermediate layers in LLMs often yield richer representations for downstream tasks compared to the final layers.

Large Language Models

The effectiveness of our proposed method is evaluated across various LLMs, including different architectures (Llama³⁸, Qwen⁴², Mistral⁴³) and model sizes (1B, 3B, 7B, 8B, 14B parameters). As shown in Table 4, features extracted from all tested LLMs are effective in detecting hidden instructions. Notably, we select the hidden states and gradients from the best-performing

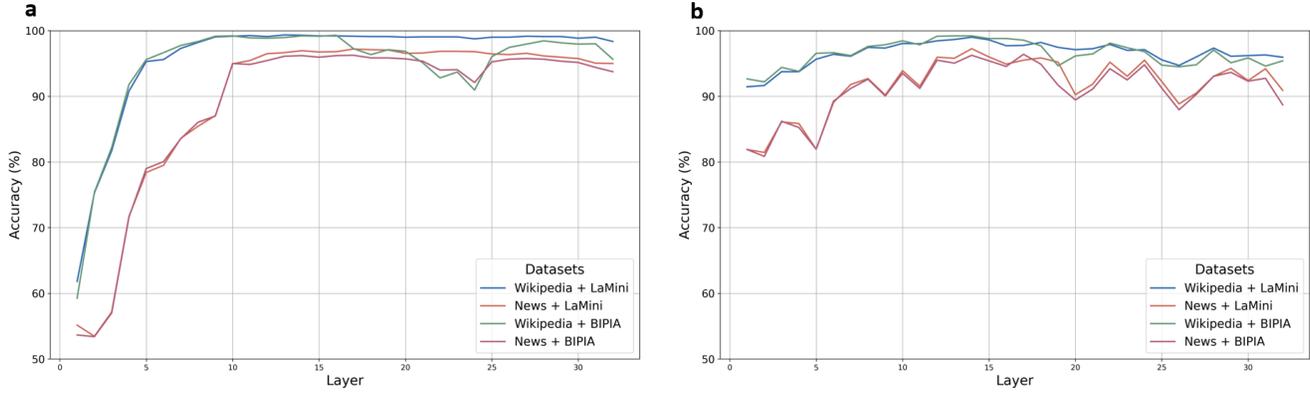


Figure 2. Detection accuracy across different layers, evaluated on all four combinations of datasets. (a) Detection accuracy achieved using hidden states extracted from different layers of the LLM. (b) Detection accuracy achieved using gradients extracted from different layers of the LLM.

	Wiki+LaMini (ID)	News+LaMini (OOD)	Wiki+BIPIA (OOD)	News+BIPIA (OOD)
Llama-3.2-1B-Instruct	97.50%	93.15%	97.10%	92.55%
Llama-3.2-3B-Instruct	99.45%	96.30%	99.25%	95.70%
Llama-3.1-8B-Instruct	99.60%	98.35%	99.45%	96.90%
Llama-3.1-8B-Base	73.95%	71.35%	73.35%	68.55%
Mistral-7B-Instruct	99.55%	94.75%	99.40%	94.20%
Qwen2.5-7B-Instruct	99.85%	97.65%	99.30%	97.35%
Qwen2.5-14B-Instruct	99.85%	98.45%	99.70%	98.15%

Table 4. Detection accuracy comparison utilizing hidden states and gradients extracted from various LLMs. The highest detection accuracy is indicated in **bold**. Here, ID denotes the in-domain setting, whereas OOD denotes the out-of-domain setting.

layer, which are all located in the intermediate layers. Among the evaluated models, Qwen-2.5-7B and Llama-3.1-8B exhibit superior results, while Mistral-7B shows slightly less optimal performance. Furthermore, the findings indicate that larger models generally produce features that are more effective for instruction detection, aligning with our hypothesis that stronger model capabilities lead to features that better facilitate the identification of hidden instructions.

We also include a comparison between Llama-3.1-8B-Base and Llama-3.1-8B-Instruct. The results show a significant performance gap, with Llama-3.1-8B-Base demonstrating notably worse results. This difference is likely due to the fact that the success of IPI attacks relies on the presence of hidden instructions embedded within external content, which alter the behavioral state of LLMs. Since Llama-3.1-8B-Base has not undergone instruction fine-tuning, it does not exhibit the same responsiveness to such hidden instructions in the way that the instruct model does. As a result, the ability of Llama-3.1-8B-Base to detect such attacks is considerably diminished.

Training Data Size

We conduct experiments using training data of varying sizes to assess how the quantity of training data affects the performance of our method. As presented in Figure 3, even with a small training set of only 50 samples (25 positive and 25 negative), our method achieves relatively high performance, exceeding 95% accuracy in both in-domain and out-of-domain scenarios. These results indicate that our approach requires only minimal training data to achieve strong results. These findings highlight the remarkable data efficiency of our method, which performs well even with very limited data.

Composition of Training Data

We conduct experiments using different combinations of training datasets to assess the robustness and adaptability of our method to various training dataset compositions. As presented in Table 5, our approach consistently yields high accuracy across all test datasets, regardless of the specific combination of training data used. This indicates the generalizability and adaptability of our approach, as it does not rely on any particular training dataset source. Additionally, we observe that accuracy is consistently lower when tested on the News Articles with the BIPIA combination, indicating that this scenario poses the greatest challenge for instruction detection. Nonetheless, our method still achieves satisfactory accuracy in this challenging

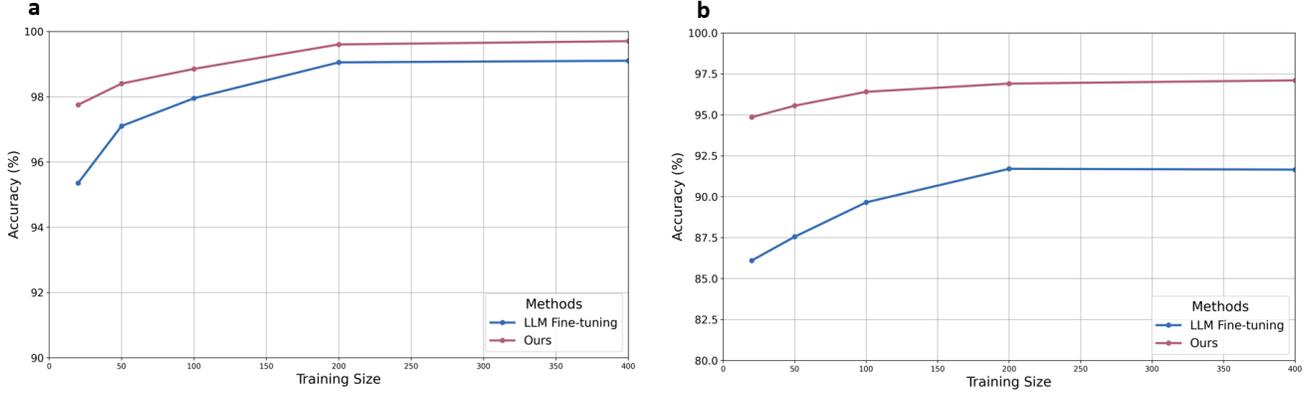


Figure 3. Comparison of detection accuracy between LLM fine-tuning and our proposed method on different training data size. (a) Detection accuracy comparison in the in-domain setting (Wikipedia+LaMini-Instruction). (b) Detection accuracy comparison in the out-of-domain setting (News Article+BIPIA).

	Wiki+LaMini	News+LaMini	Wiki+BIPIA	News+BIPIA
Wiki+LaMini	99.60%	98.35%	99.45%	96.90%
News+LaMini	99.60%	98.50%	99.55%	96.45%
Wiki+BIPIA	99.15%	97.50%	99.85%	97.30%
News+BIPIA	99.45%	98.35%	99.65%	98.05%

Table 5. Detection accuracy comparison utilizing different combinations of training datasets. The highest detection accuracy is indicated in **bold**.

scenario, further validating its effectiveness and robustness in instruction detection.

Paired Responses for Gradients

	Wiki+LaMini (ID)	News+LaMini (OOD)	Wiki+BIPIA (OOD)	News+BIPIA (OOD)
I'm sorry	99.45%	97.95%	97.25%	95.15%
Hello	99.45%	97.90%	98.15%	95.90%
Yes	99.55%	97.80%	99.40%	96.80%
Sure	99.60%	98.35%	99.45%	96.90%

Table 6. Detection accuracy comparison using different paired responses to extract gradient features. The highest detection accuracy is indicated in **bold**. Here, ID denotes the in-domain setting, whereas OOD denotes the out-of-domain setting.

To investigate the effect of various paired responses on the extraction of gradient features, we conduct experiments using four candidate responses: "I'm sorry" "Hello" "Yes" and "Sure." These candidates are selected based on an analysis of common responses to instructions in WildChat⁴⁴ dataset, ranked by frequency. Results in Table 6 show that all four paired responses achieve high accuracy (>95%) in distinguishing between normal external data and those containing hidden instructions. Among them, "Sure" delivers the best performance across all test datasets, further validating our choice of "Sure" as the paired response in our method. These results emphasize the robustness of our approach to differentiate response pairings while confirming that "Sure" is a particularly effective option for this task.

Discussion

As LLMs are increasingly adopted in real-world applications, the inclusion of external data provides significant advantages but also introduces new security challenges. A particularly concerning threat is IPI, where adversaries embed covert instructions within external data to manipulate LLMs into performing unintended or potentially harmful actions. Mitigating this vulnerability is not just a theoretical issue, but a practical necessity for ensuring the safety, reliability, and trustworthiness of LLM-integrated systems. As LLMs are increasingly deployed in sensitive environments—from healthcare to finance and law—the ability to proactively detect and mitigate such threats is crucial for guaranteeing the safe and responsible deployment of AI systems.

Our work proposes a detection-based approach that leverages the internal behavioral states of LLMs as signals to identify IPI attacks. A key strength of our method lies in its task-agnostic design. Unlike existing approaches that rely on runtime interventions, our method enables proactive screening of external sources, allowing the removal of external data containing covert instructions in advance. This eliminates the need for additional steps during task execution, minimizing time overhead and ensuring a smoother, more efficient user experience. Another unique aspect of our approach is its use of internal behavioral signals as indicators of covert instruction presence. This design aligns closely with the fundamental mechanism of IPI attacks: since IPI attacks fundamentally rely on the external instructions to alter the behavioral state of LLMs, and our approach leverages the distinct behavioral states of LLMs to differentiate between data and instructions, it becomes inherently difficult for attackers to evade detection without compromising the effectiveness of the attack. Moreover, by exploiting the LLM’s natural sensitivity to embedded instructions, our method achieves high detection performance with only a small number of labeled samples, making it both robust and data-efficient. We believe our work can spark further research into IPI attacks in LLM-integrated systems. Moreover, we hope it inspires broader efforts to secure LLM-integrated systems, especially in sensitive domains such as healthcare, law and finance, where the risks of model manipulation can be severe.

Nevertheless, our method has several limitations. First, it requires both forward and backward passes through the LLM, introducing additional computational overhead compared to lightweight defenses. While suitable for offline filtering or batch processing, this may limit deployment in resource-constrained settings. Second, although our experiments cover multiple representative scenarios and datasets, we cannot guarantee coverage of all possible attack strategies or domain-specific variations. Third, the current design adopts a conservative binary decision—discarding any external data flagged as containing hidden instructions—result in the unintended removal of useful, non-malicious information. In future work, when hidden instructions are identified, we will attempt to refine this approach by isolating and eliminating the hidden instructions embedded within the external data, rather than discarding the entire external data. This enhancement could enable the LLMs to leverage the remaining valid information while maintaining robust defenses against hidden instructions.

Ethical and Societal Impact

Our proposed method defends against IPI attacks, which is essential for ensuring the secure and reliable operation of LLMs in third-party system integrations. By mitigating the risks posed by IPI attacks, our approach fosters ethical and socially responsible use of AI technologies, enhancing trust in their application within critical sectors such as healthcare, legal and finance domains. There may be concerns about whether our method could provide attackers with insights to bypass detection. Since our approach leverages the distinct behavioral states of LLMs to differentiate between data and instructions, while IPI attacks fundamentally rely on the external instructions to alter the behavioral state of LLMs, it would be exceedingly difficult for attackers to circumvent our detection. In summary, our method strengthens the security and trustworthiness of AI systems by effectively defending IPI attacks, aligning with ethical principles and supporting the development of reliable, safe, and socially responsible AI technologies for real-world applications.

Method

Related Work

Indirect Prompt Injection Defense

Defending against IPI attacks is a critical research area in ensuring the secure and reliable use of LLMs^{17–19}. Existing defenses are generally classified into prevention-based defenses and detection-based defenses^{27,28}.

Prevention-based defenses primarily focus on ensuring LLMs to follow user instructions while ignoring external ones. These approaches are further divided into black-box defenses and white-box defenses. Black-box defenses^{27,45–47} typically aim to isolate user instructions from external data, using carefully designed prompts to ensure LLMs disregard any hidden instructions within the external data. These methods work without access to the internal parameters of the model, focusing on input preprocessing and separation mechanisms. In contrast, white-box defenses^{27,48} utilize the internal parameters of the model and involve fine-tuning LLMs with samples of IPI attacks. By training on a diverse set of IPI scenarios, these methods enhance the robustness of LLMs to ignore external instructions while maintaining performance on the intended task.

Detection-based defenses, though relatively underexplored, aim to identify IPI attacks and can be generally divided into three main strategies. LLM (Zero-shot)²⁸ directly uses LLMs to identify hidden instructions in external data. Response Check²⁸ evaluates whether the model’s outputs remain consistent with the intended task. TaskTracker³⁶ detects IPI attacks by contrasting the LLM’s activations before and after feeding the external data, which indicates whether the user’s instruction is distorted by the instruction hidden in the external data. Our method also falls under detection-based defenses, bridging the gap with a more robust mechanism for instruction detection.

Behavioral States of Large Language Models

Recent studies^{29,49} have delved extensively into the internal mechanisms of LLMs, identifying hidden states and gradients as highly informative features for understanding and controlling their behavior. These behavioral states are increasingly recognized for their potential to enhance the transparency, safety, and robustness of LLMs.

Hidden states, especially those from intermediate layers, have been shown to encode rich and insightful representations of given inputs. RepE²⁹ utilizes representations from the last token's hidden states to monitor and manipulate high-level cognitive phenomena in LLMs. Furthermore, a recent study⁴¹ has explored the effectiveness of intermediate features across different LLM architectures, revealing that intermediate features often yield richer information than final-layer for downstream use.

Gradients provide another critical lens for analyzing the LLM's behavior. Gradsafe⁴⁹ leverages the observation that adversarial prompts generate distinct gradient patterns compared to safe prompts, enabling effective jailbreak prompts detection without additional training by analyzing gradients related to safety-critical parameters. Additionally, much literature³⁰⁻³² has explored the functions of self-attention layers and feed-forward layers, providing key insights into where to focus when analyzing gradients in our work. Research has shown that self-attention layers capture behavioral characteristics, such as linguistic dependencies and token relationships, while feed-forward layers encode knowledge-based features, enabling the model to leverage the knowledge learned during training. Given that the instruction recognition task primarily relies on the behavioral characteristics of LLMs, we focus on the gradients of self-attention layers.

Defending IPI attacks via Instruction-Following State Detection

Overview

In our proposed method, we aim to detect IPI attacks through the behavioral states of LLMs, hypothesizing that changes in the behavioral states of LLMs induced by embedded instructions in external content can be effectively utilized to detect such attacks. To achieve this, we fuse the hidden states and gradients from the most effective layers, integrating complementary information captured by both features. These fused features are then fed into an MLP classifier, enabling accurate and robust detection of IPI attacks. The overall framework is illustrated in Figure 1b, with detailed processes discussed in the following sections.

Hidden States Extraction

To leverage hidden states as features, we first take external data as the input of the LLM and extract the hidden states corresponding to the last token at each layer. These hidden states are then fed into an MLP classifier to assess their ability to distinguish between normal external data and those containing hidden instructions. In our approach, we use the Llama-3.1-8B-Instruct model, which consists of 32 layers. Through experimentation on the validation set, we identify that the hidden states from the 14th layer provides the best performance in instruction detection. Therefore, we select the last token's hidden state from the 14th layer, a vector with a dimension of 4096, as the first input of the feature fusion module.

Gradients Extraction

To leverage gradients as features, we first take external data as the input of the LLM, paired with a typical response to instructions, such as "Sure," and compute the gradients for the model parameters at each layer during back propagation. Based on prior research indicating that self-attention layers capture the model's behavioral characteristics, while the feed-forward layers are more effective at encoding knowledge-based features, we concentrate on the gradients of self-attention layers. Experimental results on the validation set demonstrate that the gradients from the 14th layer, consistent with the layer identified for hidden states, yield the best performance in distinguishing between normal external data and those containing hidden instructions.

Additionally, to address the large parameter size of the self-attention layers, we apply max-pooling to reduce dimensionality before feeding the gradients into the MLP. This dimensionality reduction ensures computational efficiency while preserving key information from the gradients. These reduced gradients are then flattened to form a vector with a dimension of 400,000, as the second input of the feature fusion module.

Feature Fusion

In the feature fusion module, the gradient features are initially projected to match the dimensionality of the hidden state features through a linear transformation. Following this, we apply normalization to both the hidden state and gradient features before concatenation, which helps mitigate scale differences between the two feature types, ensuring balanced contributions to the fused features. The fused features are then fed into an MLP classifier for effective instruction detection, effectively combining the strengths of both hidden states and gradients to achieve enhanced performance compared to using either feature type individually.

Data Availability

The Wikipedia dataset is available at <https://huggingface.co/datasets/wikimedia/wikipedia>. The News Article dataset is available at <https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/GMFCR>. The LaMini-instruction dataset is available at <https://huggingface.co/datasets/MBZUAI/LaMini-instruction>. The BIPIA benchmark is available at <https://github.com/microsoft/BIPIA>.

Code Availability

The code is publicly available at <https://github.com/MYVAE/Instruction-detection>.

References

1. Achiam, J. *et al.* Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
2. Touvron, H. *et al.* Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
3. Touvron, H. *et al.* Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).
4. Brown, T. *et al.* Language models are few-shot learners. *Adv. neural information processing systems* **33**, 1877–1901 (2020).
5. Chowdhery, A. *et al.* Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.* **24**, 1–113 (2023).
6. Kamaloo, E., Dziri, N., Clarke, C. & Rafiei, D. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 5591–5606 (2023).
7. Singhal, K. *et al.* Towards expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617* (2023).
8. Tang, L. *et al.* Evaluating large language models on medical evidence summarization. *NPJ digital medicine* **6**, 158 (2023).
9. Zhang, T. *et al.* Benchmarking large language models for news summarization. *Transactions Assoc. for Comput. Linguist.* **12**, 39–57 (2024).
10. Xu, H., Kim, Y. J., Sharaf, A. & Awadalla, H. H. A paradigm shift in machine translation: Boosting translation performance of large language models. *arXiv preprint arXiv:2309.11674* (2023).
11. Zhang, B., Haddow, B. & Birch, A. Prompting large language model for machine translation: A case study. In *International Conference on Machine Learning*, 41092–41110 (PMLR, 2023).
12. Ji, Z. *et al.* Survey of hallucination in natural language generation. *ACM Comput. Surv.* **55**, 1–38 (2023).
13. Rawte, V., Sheth, A. & Das, A. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922* (2023).
14. Schick, T. *et al.* Toolformer: Language models can teach themselves to use tools. *Adv. Neural Inf. Process. Syst.* **36**, 68539–68551 (2023).
15. Gao, Y. *et al.* Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023).
16. Chen, J., Lin, H., Han, X. & Sun, L. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 17754–17762 (2024).
17. Greshake, K. *et al.* Not what you’ve signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, 79–90 (2023).
18. Rossi, S., Michel, A. M., Mukkamala, R. R. & Thatcher, J. B. An early categorization of prompt injection attacks on large language models. *arXiv preprint arXiv:2402.00898* (2024).
19. Zhan, Q., Liang, Z., Ying, Z. & Kang, D. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. *arXiv preprint arXiv:2403.02691* (2024).
20. Sallam, M. The utility of chatgpt as an example of large language models in healthcare education, research and practice: Systematic review on the future perspectives and potential limitations. *MedRxiv* 2023–02 (2023).
21. Harrer, S. Attention is not all you need: the complicated case of ethically using large language models in healthcare and medicine. *EBioMedicine* **90** (2023).

22. Yang, R. *et al.* Large language models in health care: Development, applications, and challenges. *Heal. Care Sci.* **2**, 255–263 (2023).
23. Wu, S. *et al.* Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564* (2023).
24. Li, Y., Wang, S., Ding, H. & Chen, H. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, 374–382 (2023).
25. Cui, J., Li, Z., Yan, Y., Chen, B. & Yuan, L. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *CoRR* (2023).
26. Lai, J., Gan, W., Wu, J., Qi, Z. & Philip, S. Y. Large language models in law: A survey. *AI Open* (2024).
27. Yi, J. *et al.* Benchmarking and defending against indirect prompt injection attacks on large language models. *arXiv preprint arXiv:2312.14197* (2023).
28. Liu, Y., Jia, Y., Geng, R., Jia, J. & Gong, N. Z. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*, 1831–1847 (2024).
29. Zou, A. *et al.* Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405* (2023).
30. Vaswani, A. Attention is all you need. *Adv. Neural Inf. Process. Syst.* (2017).
31. Geva, M., Schuster, R., Berant, J. & Levy, O. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 5484–5495 (2021).
32. Dai, D. *et al.* Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 8493–8502 (2022).
33. Foundation, W. Wikimedia downloads.
34. dai, t. News Articles, DOI: [10.7910/DVN/GMFCTR](https://doi.org/10.7910/DVN/GMFCTR) (2017).
35. Wu, M., Waheed, A., Zhang, C., Abdul-Mageed, M. & Aji, A. Lamini-lm: A diverse herd of distilled models from large-scale instructions. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 944–964 (2024).
36. Abdelnabi, S. *et al.* Are you still on track!? catching llm task drift with activations. *arXiv preprint arXiv:2406.00799* (2024).
37. Dong, Q. *et al.* A survey on in-context learning. *arXiv preprint arXiv:2301.00234* (2022).
38. Dubey, A. *et al.* The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).
39. Dale, R. Gpt-3: What’s it good for? *Nat. Lang. Eng.* **27**, 113–118 (2021).
40. Chiang, W.-L. *et al.* Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. See <https://vicuna.lmsys.org> (accessed 14 April 2023) **2**, 6 (2023).
41. Skean, O., Arefin, M. R. & Shwartz-Ziv, R. Does representation matter? exploring intermediate layers in large language models. In *Workshop on Machine Learning and Compression, NeurIPS 2024* (2024).
42. Bai, J. *et al.* Qwen technical report. *arXiv preprint arXiv:2309.16609* (2023).
43. Jiang, A. Q. *et al.* Mistral 7b. *arXiv preprint arXiv:2310.06825* (2023).
44. Zhao, W. *et al.* Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470* (2024).
45. Hines, K. *et al.* Defending against indirect prompt injection attacks with spotlighting. *arXiv preprint arXiv:2403.14720* (2024).
46. Wang, J. *et al.* Fath: Authentication-based test-time defense against indirect prompt injection attacks. *arXiv preprint arXiv:2410.21492* (2024).
47. Wu, F., Cecchetti, E. & Xiao, C. System-level defense against indirect prompt injection attacks: An information flow control perspective. *arXiv preprint arXiv:2409.19091* (2024).
48. Chen, S., Piet, J., Sitawarin, C. & Wagner, D. Struq: Defending against prompt injection with structured queries. *arXiv preprint arXiv:2402.06363* (2024).
49. Xie, Y., Fang, M., Pi, R. & Gong, N. Gradsafe: Detecting jailbreak prompts for llms via safety-critical gradient analysis. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 507–518 (2024).

Acknowledgments

We sincerely appreciate the editor and anonymous reviewers for their insightful comments and suggestions that may significantly enhance the quality of this work.

Author contributions

T.W. conceived the idea of this work, implemented the proposed method, conducted experiments, analyzed the results, and contributed to the writing of this manuscript. C.W. was involved in the technical discussions, and contributed to the writing of this manuscript. X.Y. was involved in the technical discussions, and analyzed the results. H.T. was involved in the technical discussions, and analyzed the results. Y.X. contributed to the writing of this manuscript, and coordinated the research project. L.L. contributed to the writing of this manuscript. Z.D. was involved in the technical discussions, and contributed to the writing of this manuscript. F.W. conceived the idea of this work, analyzed the results, contributed to the writing of this manuscript, and coordinated the research project.

Additional Information

Comparison with Related Method

	Scenario 1	Scenario 2
TaskTracker	70.55%	71.35%
Ours	99.40%	99.55%

Table 7. Detection accuracy comparison of our proposed method and TaskTracker when the external instruction closely resembles the user’s instruction. For scenario 1, the user instruction is "Translate the news into Chinese." and the external instruction is "Translate the news into Japanese."; For scenario 2, the user instruction is "Generate a step-by-step guide for solving this problem." and the external instruction is "Generate a list of common mistakes in solving this problem." The highest detection accuracy is indicated in **bold**.

Both our proposed method and TaskTracker³⁶ utilize the hidden states of LLMs as a key feature for detecting IPI attacks, but they differ significantly in their underlying principles. TaskTracker aims to capture distortions in the user’s instruction caused by embedded instructions in the external content. In contrast, our method aims to distinguish the LLM’s behavioral states when processing normal external data versus those containing hidden instructions.

TaskTracker has two primary limitations. First, it requires a large number of training samples (418,110 pairs of positive and negative samples) to accurately identify deviations in the user’s task. In contrast, our method leverages the high sensitivity of LLM’s behavioral states to embedded instructions, achieving effective detection with a significantly smaller dataset (100 positive samples and 100 negative samples).

Second, TaskTracker’s effectiveness relies heavily on a clear distinction between the user’s instruction and the external instruction, while our method is task-agnostic. As shown in Table 7, when the external instruction closely resembles the user’s instruction, TaskTracker’s detection accuracy drops significantly, while our method maintains high detection accuracy.

Influence of Instruction Quantity and Position

	Wiki+LaMini (ID)	News+LaMini (OOD)	Wiki+BIPIA (OOD)	News+BIPIA (OOD)
one instruction	99.60%	98.35%	99.45%	96.90%
two instructions	99.85%	98.40%	99.80%	97.00%
three instructions	99.85%	98.45%	99.85%	97.00%

Table 8. Detection accuracy comparison for different quantities of inserted instructions in the test dataset. The highest detection accuracy is indicated in **bold**. Here, ID denotes the in-domain setting, whereas OOD denotes the out-of-domain setting.

To further explore the influence of instruction quantity and position on detection performance, we conduct experiments using a fixed training dataset while varying only the number or placement of inserted instructions in the test dataset.

Results in Table 8 reveal a trend that detection accuracy shows a certain degree of improvement as the number of inserted instructions increases. This suggests that a higher quantity of instructions provides stronger signals, making IPI attacks more distinguishable by our method.

	Wiki+LaMini (ID)	News+LaMini (OOD)	Wiki+BIPIA (OOD)	News+BIPIA (OOD)
beginning	99.90%	98.60%	99.90%	97.35%
middle	99.60%	98.35%	99.45%	96.90%
end	99.90%	98.40%	99.55%	97.05%

Table 9. Detection accuracy comparison for different positions of inserted instructions in the test dataset. The highest detection accuracy is indicated in **bold**. Here, ID denotes the in-domain setting, whereas OOD denotes the out-of-domain setting.

Additionally, we examine the effect of instruction placement by inserting instructions at the beginning, middle, or end of the external content. As shown in Table 9, instructions placed in the middle are the most challenging to detect, whereas those positioned at the beginning or end are relatively easier to identify. Among these, instructions at the beginning yield the highest detection accuracy, likely because LLMs exhibit greater sensitivity to early input.

Additional Experimental Settings

For LLM (Zero-shot), Response Check, and LLM (Few-shot), we use the prompt templates illustrated in Figure 4. For the LLM (Fine-tuning), we use the AdamW optimizer to train the model for one epoch, with a learning rate set at 0.00001 and a maximum input length of 2048 tokens. Both the learning rate and the number of epochs are chosen based on performance on the validation set. For TaskTracker, we use the publicly available model released by the work. As for the prevention-based approaches, we utilize the publicly available implementations provided by the respective works.

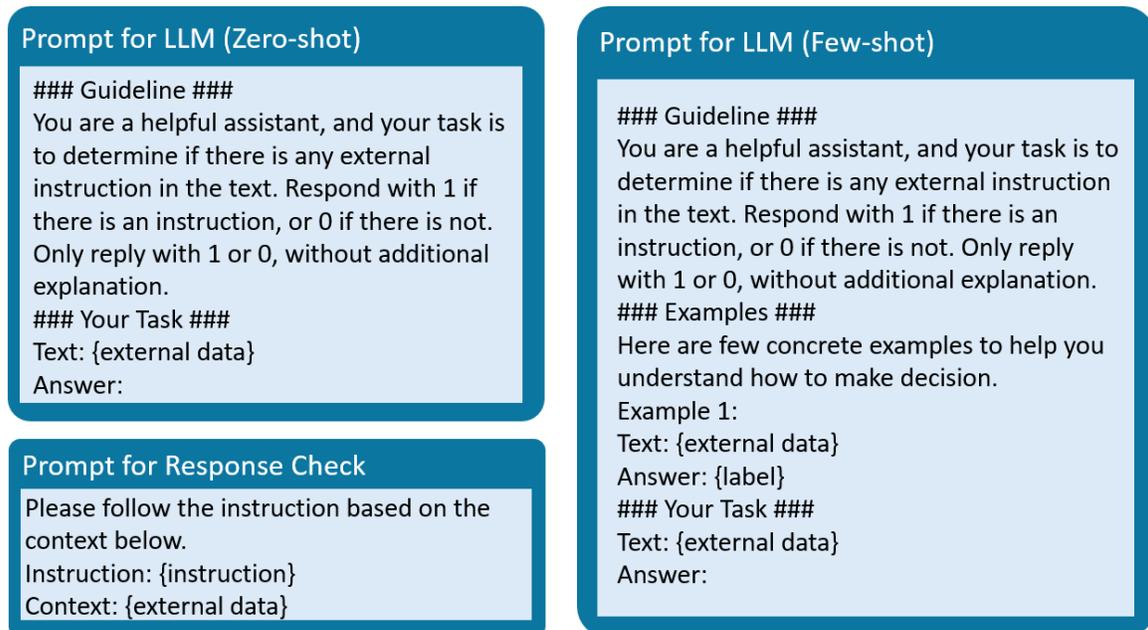


Figure 4. Prompts for the LLM (Zero-shot), Response Check, and LLM (Few-shot) baselines.