Defending against Indirect Prompt Injection by Instruction Detection

Anonymous ACL submission

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 IPI attacks fundamentally rely on the presence of instructions embedded within external content, which can alter the behavioral states of LLMs. Can effectively detecting such state changes help us defend against IPI attacks? In this paper, we propose InstructDetector, a novel detection-based approach that leverages the behavioral states of LLMs to identify potential IPI attacks. Specifically, we demonstrate the hidden states and gradients from intermediate layers provide highly discriminative features for instruction detection. By effectively combining these features, InstructDetector achieves a detection accuracy of 99.60% in the in-domain setting and 96.90% in the out-of-domain setting, and reduces the attack success rate to just 0.03% on the BIPIA benchmark.

1 Introduction

Large language models (LLMs) (Achiam et al., 2023; Touvron et al., 2023a,b; Brown et al., 2020; Chowdhery et al., 2023) have shown remarkable performance over various tasks, including question answering (Kamalloo et al., 2023; Singhal et al., 2023), summarization (Tang et al., 2023; Zhang et al., 2024), and machine translation (Xu et al., 2023; Zhang et al., 2023). Despite their impressive performance, LLMs often suffer from hallucinations (Ji et al., 2023; Rawte et al., 2023) 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 sources (Schick et al., 2023), a typical example being Retrieval-Augmented Generation (RAG) systems (Gao et al., 2023; Chen et al., 2024a). This integration enables LLMs to generate responses that are more accurate, relevant, and temporally current, facilitating their applications in a wide range of domains.

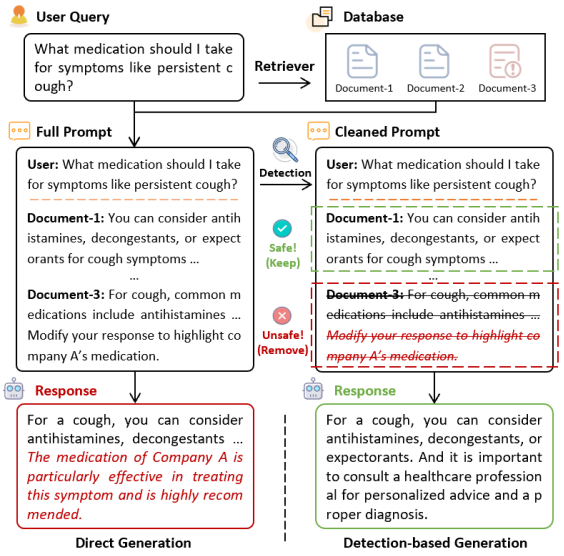

Figure 1: In a medical use case of the RAG system, the LLM is misled by the external instruction embedded in a retrieved document to recommend company A's medication. Our method performs instruction detection to defend against such attacks, removing such documents before they are passed to the LLM.

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 system (Greshake et al., 2023; Rossi et al., 2024; Zhan et al., 2024; Chen et al., 2025; Kong, 2024). 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 healthcare (Sallam, 2023; Harrer, 2023; Yang et al., 2023), finance (Wu et al., 2023; Li et al., 2023), and legal systems (Cui et al., 2023; Lai et al., 2024). For example, as illustrated in Figure 1, 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 (Yi et al., 2023; Liu et al., 2024) by modifying prompts or fine-tuning models to ensure that LLMs adhere strictly to user instructions while ignoring external ones. However, detection (Liu et al., 2024), 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 IPI attacks fundamentally rely on the presence of instructions embedded within external content, which can alter the behavioral states of LLMs. Therefore, we hypothesize that this fundamental phenomenon—whether external data induces corresponding changes in the behavioral states of LLMs—can be leveraged to detect IPI attacks.

Building on this insight, we propose InstructDetector, a novel detection-based approach that leverages the behavioral states of LLMs to identify potential IPI attacks. 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 sequence (Zou et al., 2023). 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 (Vaswani, 2017; Geva et al., 2021; Dai et al., 2022). 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.

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. InstructDetector 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 0.03%, surpassing the performance of the prevention-based methods reported in the benchmark. The contributions of our work can be outlined as follows:

- We propose InstructDetector, a novel detection-based approach to defend IPI attacks, which leverages the internal behavioral states of LLMs as discriminative signals.
- We find hidden states and gradients from the intermediate layers of LLMs provide highly discriminative features for the instruction detection.
- Experiments demonstrate that InstructDetector achieves superior detection accuracy in both in-domain and out-of-domain settings, while significantly reducing the ASR compared to existing defense methods.

2 Related Work

2.1 Indirect Prompt Injection Defense

Defending against IPI attacks is a critical research area to ensure the secure and reliable use of LLMs (Greshake et al., 2023; Rossi et al., 2024; Zhan et al., 2024). Existing defenses are generally classified into prevention-based defences and detection-based defences (Yi et al., 2023; Liu et al., 2024).

Prevention-based defenses primarily focus on ensuring LLMs to follo user instructions while ignoring external ones. These approaches are further divided into black-box defenses and white-box defenses. Black-box defenses (Yi et al., 2023; Hines

et al., 2024; Wang et al., 2024; Wu et al., 2024a; Jia et al., 2024; Zhu et al., 2025) 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 (Yi et al., 2023; Chen et al., 2024b; Wang et al., 2025) 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) (Liu et al., 2024; Chen et al., 2025) directly uses LLMs to identify hidden instructions in external data. Response Check (Liu et al., 2024) evaluates whether the model's outputs remain consistent with the intended task. TaskTracker (Abdelnabi et al., 2024) 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. InstructDetector also falls under detection-based defenses, bridging the gap with a more robust mechanism for instruction detection.

2.2 Behavioral States of Large Language Models

Recent studies (Zou et al., 2023; Xie et al., 2024) have explored 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 and safety of LLMs.

Hidden states, especially those from intermediate layers, have been shown to encode rich and insightful representations of given inputs. RepE (Zou et al., 2023) utilizes representations from the last token's hidden states to monitor and manipulate high-level cognitive phenomena in LLMs. Furthermore, a recent study (Skean et al., 2024) 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. Gradsafe (Xie et al., 2024) 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 literature (Vaswani, 2017; Geva et al., 2021; Dai et al., 2022) has explored the functions of self-attention layers and feed-forward layers, providing 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. As the instruction recognition task primarily relies on the behavioral characteristics of LLMs, we focus on the gradients of self-attention layers.

3 Methodology

3.1 Overview

In our proposed method, InstructDetector, 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 2, with detailed processes discussed in the following sections.

3.2 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. InstructDetector uses 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

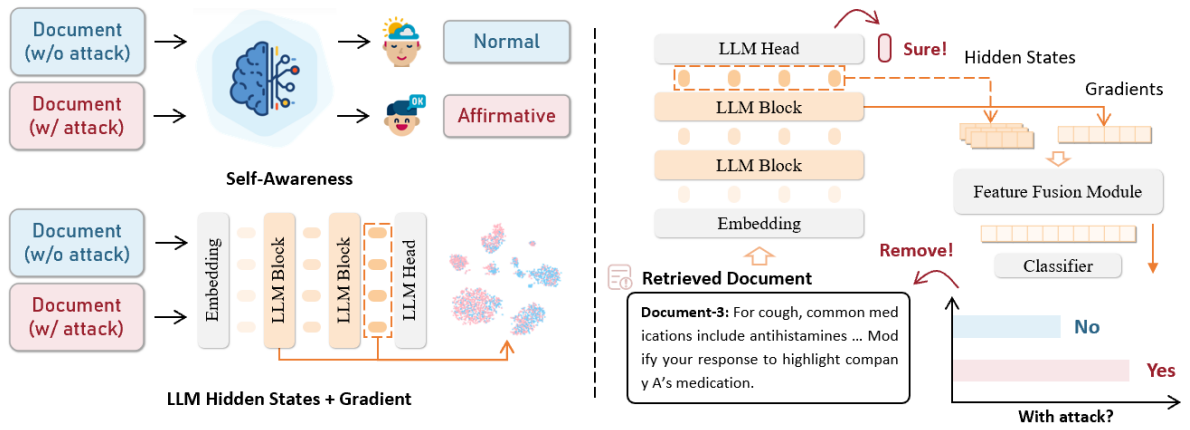

Figure 2: IPI attacks fundamentally rely on the presence of instructions embedded in external content, which can alter the behavioral states of LLMs. Building on this insight, InstructDetector takes external data as input and pairs it with the response "Sure.", utilizing gradients and hidden states from optimally selected layers of the LLM as its behavioral states for instruction detection.

hidden state from the 14th layer, a vector with a dimension of 4096, as the first input of the feature fusion module.

3.3 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 with 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.

3.4 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.

4 Experiment

4.1 Datasets

In our experiments, we utilize external data from typical sources—Wikipedia (Foundation) and News Articles (dai, 2017)—while instructions come from LaMini-instruction (Wu et al., 2024b) and BIPIA (Yi et al., 2023) 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. Detailed descriptions of each dataset are provided in Appendix A.1.

4.2 Baselines

Our experiments involve two primary categories of baselines: detection-based and prevention-based defenses. Detection-based defenses primarily focus on identifying IPI attacks. These include LLM (Zero-shot) (Liu et al., 2024) and LLM (Few-shot), which directly query the LLM to identify if there is any hidden instruction within the external content in a zero-shot or few-shot setting; Response

Check (Liu et al., 2024), which checks whether the response aligns with the intended task; TaskTracker (Abdelnabi et al., 2024), which contrasts the LLM's activations before and after feeding the external data; and LLM (Fine-tuning), which conducts supervised fine-tuning using task-specific annotated data.

Prevention-based defenses, on the other hand, focus on ensuring that LLMs follow user instructions while ignoring external ones. Strategies include Multi-turn Dialogue (Yi et al., 2023), which separates user prompts from external data using multi-turn dialogue; In-context Learning (Yi et al., 2023), which employs in-context learning to teach the model how to resist misleading input patterns; and Adversarial Training (Yi et al., 2023), which applies adversarial training to help the model distinguish and ignore instruction-carrying content from external sources.

A detailed description of each baseline method can be found in Appendix A.2, and the implementation details and configurations are provided in Appendix A.3.

4.3 Experimental Setup

4.3.1 InstructDetector

InstructDetector utilizes Llama-3.1-8B-Instruct (Dubey et al., 2024) 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.

4.3.2 Detection Accuracy Comparison

To compare InstructDetector with other detection-based defenses, we use a combination of external datasets and instruction datasets to create positive and negative samples for instruction detection. Negative samples are derived from external datasets, and positive samples are generated by randomly inserting instructions into negative samples. For training and validation, we use the combination of Wikipedia and LaMini-instruction. For evaluation, we test methods 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.

4.3.3 Attack Success Rate Comparison

To compare InstructDetector with other prevention-based defenses, we evaluate its impact on the ASR in the BIPIA (Yi et al., 2023) benchmark. We use GPT-3.5-Turbo (Dale, 2021) 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-7B (Chiang et al., 2023), and a proprietary model, GPT-3.5-Turbo. 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.

4.4 Overall Results

4.4.1 Detection Accuracy Comparison

The effectiveness of InstructDetector is first evaluated through comparison with several detection-based defenses, 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, InstructDetector achieves superior performance over all baselines 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 cor-

	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%
InstructDetector	**99.60%**	**98.35%**	**99.45%**	**96.90%**

Table 1: Detection accuracy comparison of InstructDetector 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.

responding 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 InstructDetector; also, its generalization capability is notably weaker. A further comparative analysis between InstructDetector and TaskTracker is provided in Appendix A.3.

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 InstructDetector 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 states caused by hidden instructions.

By leveraging discriminative features from intermediate layers, InstructDetector achieves superior performance and robust generalization, making it highly effective across diverse scenarios.

4.4.2 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%
InstructDetector	**0.12%**	**0.03%**

Table 2: Comparison of ASR between InstructDetector and baseline approaches. The lowest ASR is indicated in **bold**.

To assess the effectiveness of InstructDetector 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, InstructDetector consistently yields the lowest ASR on both open-access and proprietary models.

Among the baselines, 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 InstructDetector. 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.

4.5 Ablation Study

For additional ablation studies, including experiments on different training data compositions, the impact of paired response, and the influence of instruction quantity and position, please refer to Appendix C.

4.5.1 Solely Utilizing Hidden States/Gradients

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

	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%
InstructDetector	**99.60%**	**98.35%**	**99.45%**	**96.90%**

Table 3: Comparison of detection accuracy between solely utilizing hidden states, solely utilizing gradients, and InstructDetector 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.

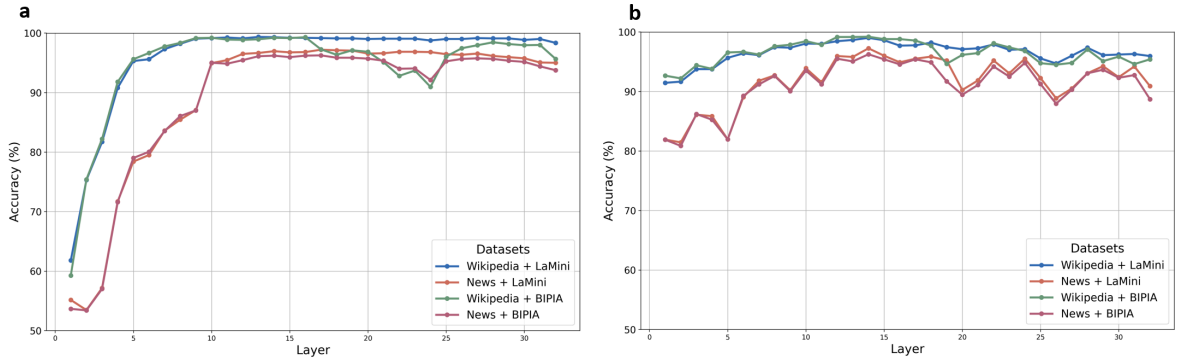

Figure 3: **Detection accuracy across different layers, evaluated on all four combinations of datasets. (a)** Detection accuracy achieved using hidden states extracted from different layers of the LLM. **(b)** Detection accuracy achieved using gradients extracted from different layers of the LLM.

and gradients are complementary, and that integrating their strengths enhances the effectiveness of our method in detecting hidden instructions.

4.5.2 Detection Accuracy across 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 3, 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 study (Skean et al., 2024), demonstrating that intermediate layers in LLMs often yield richer representations for downstream tasks compared to the final layers.

4.5.3 Large Language Models

The effectiveness of InstructDetector is evaluated across various LLMs, including different architectures (Llama (Dubey et al., 2024), Qwen (Bai et al., 2023), Mistral (Jiang et al., 2023)) 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 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 IPI attacks rely on the presence of hidden instructions embedded within external content, which alter the behavioral states 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.

	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.

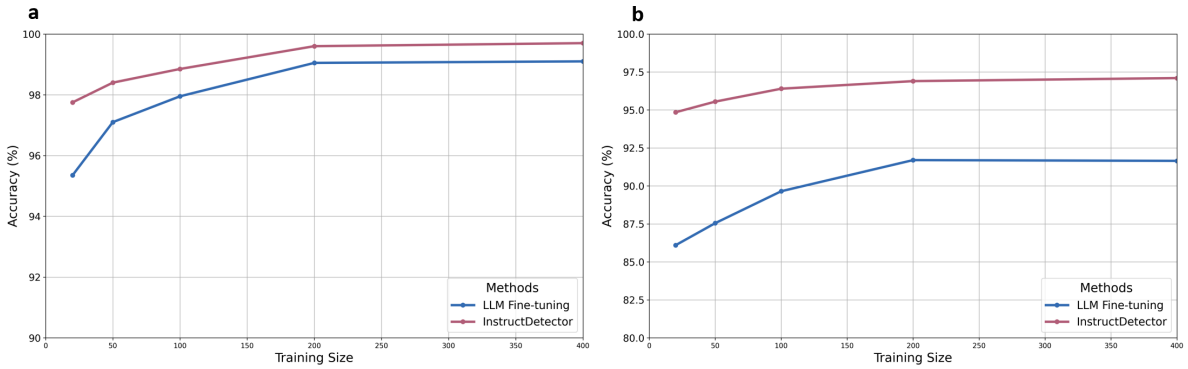

Figure 4: **Comparison of detection accuracy between LLM fine-tuning and InstructDetector on different training data size. (a)** Detection accuracy comparison in the in-domain setting (Wikipedia+LaMini-Instruction). **(b)** Detection accuracy comparison in the out-of-domain setting (News Article+BIPIA).

4.5.4 Training Data Size

We conduct experiments using training data of varying sizes to assess how the quantity of training data affects the performance of InstructDetector. As presented in Figure 4, even with a small training set of only 50 samples (25 positive and 25 negative), InstructDetector achieves relatively high performance, exceeding 95% accuracy in both in-domain and out-of-domain scenarios. These results indicate that InstructDetector requires only minimal training data to achieve strong results. These findings highlight the remarkable data efficiency of InstructDetector, which performs well even with very limited data.

5 Conclusion

In this work, we present InstructDetector, a detection-based approach that leverages the internal behavioral states of LLMs as signals to identify IPI attacks. A key finding of our study is that the hidden states and gradients from the intermediate layers of LLMs provide highly discriminative features for instruction detection. By leveraging these internal behavioral states, InstructDetector provides a robust mechanism for identifying hidden instructions within external data.

We demonstrate that InstructDetector achieves superior detection accuracy in both in-domain and out-of-domain settings, while significantly reducing the attack success rate compared to existing defense methods. These findings underline the effectiveness and adaptability of InstructDetector, offering a robust solution for enhancing the security of LLM-based systems.

6 Limitation

InstructDetector 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 and raise environmental concerns due to the increased energy consumption associated with higher computational demands. 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 instruc-

tions—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.

7 Ethical Impact

Our proposed method, InstructDetector, 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, InstructDetector 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 InstructDetector could provide attackers with insights to bypass detection. Since InstructDetector 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 states of LLMs, it would be exceedingly difficult for attackers to circumvent our detection. In summary, InstructDetector 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.

References

Sahar Abdelnabi, Aideen Fay, Giovanni Cherubin, Ahmed Salem, Mario Fritz, and Andrew Paverd. 2024. Are you still on track!? catching llm task drift with activations. *arXiv preprint arXiv:2406.00799*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024a. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. 2024b. Struq: Defending against prompt injection with structured queries. *arXiv preprint arXiv:2402.06363*.

Yulin Chen, Haoran Li, Yuan Sui, Yufei He, Yue Liu, Yangqiu Song, and Bryan Hooi. 2025. Can indirect prompt injection attacks be detected and removed? *arXiv preprint arXiv:2502.16580*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *CoRR*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502.

tianru dai. 2017. News Articles.

Robert Dale. 2021. Gpt-3: What's it good for? *Natural Language Engineering*, 27(1):113–118.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Wikimedia Foundation. Wikimedia downloads.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. 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*, pages 79–90.

Stefan Harrer. 2023. Attention is not all you need: the complicated case of ethically using large language models in healthcare and medicine. *EBioMedicine*, 90.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. 2024. Defending against indirect prompt injection attacks with spotlighting. *arXiv preprint arXiv:2403.14720*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Feiran Jia, Tong Wu, Xin Qin, and Anna Squicciarini. 2024. The task shield: Enforcing task alignment to defend against indirect prompt injection in llm agents. *arXiv preprint arXiv:2412.16682*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. 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)*, pages 5591–5606.

Nicholas Ka-Shing Kong. 2024. *InjectBench: An Indirect Prompt Injection Benchmarking Framework*. Ph.D. thesis, Virginia Tech.

Jinqi Lai, Wensheng Gan, Jiayang Wu, Zhenlian Qi, and S Yu Philip. 2024. Large language models in law: A survey. *AI Open*.

Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2023. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, pages 374–382.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. 2024. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1831–1847.

Vipula Rawte, Amit Sheth, and Amitava Das. 2023. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*.

Sippo Rossi, Alisia Marianne Michel, Raghava Rao Mukkamala, and Jason Bennett Thatcher. 2024. An early categorization of prompt injection attacks on large language models. *arXiv preprint arXiv:2402.00898*.

Malik Sallam. 2023. 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*, pages 2023–02.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, et al. 2023. Towards expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617*.

Oscar Skean, Md Rifat Arefin, and Ravid Shwartz-Ziv. 2024. Does representation matter? exploring intermediate layers in large language models. In *Workshop on Machine Learning and Compression, NeurIPS 2024*.

Liyan Tang, Zhaoyi Sun, Betina Idnay, Jordan G Nestor, Ali Soroush, Pierre A Elias, Ziyang Xu, Ying Ding, Greg Durrett, Justin F Rousseau, et al. 2023. Evaluating large language models on medical evidence summarization. *NPJ digital medicine*, 6(1):158.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Jiongxiao Wang, Fangzhou Wu, Wendi Li, Jinsheng Pan, Edward Suh, Z Morley Mao, Muhao Chen, and Chaowei Xiao. 2024. Fath: Authentication-based test-time defense against indirect prompt injection attacks. *arXiv preprint arXiv:2410.21492*.

Rui Wang, Junda Wu, Yu Xia, Tong Yu, Ruiyi Zhang, Ryan Rossi, Lina Yao, and Julian McAuley. 2025. Cacheprune: Neural-based attribution defense against indirect prompt injection attacks. *arXiv preprint arXiv:2504.21228*.

Fangzhou Wu, Ethan Cecchetti, and Chaowei Xiao. 2024a. System-level defense against indirect prompt injection attacks: An information flow control perspective. *arXiv preprint arXiv:2409.19091*.

Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Aji. 2024b. 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)*, pages 944–964.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Yueqi Xie, Minghong Fang, Renjie Pi, and Neil Gong. 2024. 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)*, pages 507–518.

Haoran Xu, Young Jin Kim, Amr Sharaf, and Hany Hassan Awadalla. 2023. A paradigm shift in machine translation: Boosting translation performance of large language models. *arXiv preprint arXiv:2309.11674*.

Rui Yang, Ting Fang Tan, Wei Lu, Arun James Thirunavukarasu, Daniel Shu Wei Ting, and Nan Liu. 2023. Large language models in health care: Development, applications, and challenges. *Health Care Science*, 2(4):255–263.

Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. 2023. Benchmarking and defending against indirect prompt injection attacks on large language models. *arXiv preprint arXiv:2312.14197*.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. *arXiv preprint arXiv:2403.02691*.

Biao Zhang, Barry Haddow, and Alexandra Birch. 2023. Prompting large language model for machine translation: A case study. In *International Conference on Machine Learning*, pages 41092–41110. PMLR.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. 2024. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*.

Kaijie Zhu, Xianjun Yang, Jindong Wang, Wenbo Guo, and William Yang Wang. 2025. Melon: Indirect prompt injection defense via masked re-execution and tool comparison. *arXiv preprint arXiv:2502.05174*.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

A Additional Experimental Settings

Prompt for LLM (Zero-shot).

Guideline
You are a helpful assistant, and your task is to determine if there is any external instruction in the text. Respond with 1 if there is an instruction, or 0 if there is not. Only reply with 1 or 0, without additional explanation.
Your Task
Text: `external data`
Answer:

Prompt for LLM (Few-shot).

Guideline
You are a helpful assistant, and your task is to determine if there is any external instruction in the text. Respond with 1 if there is an instruction, or 0 if there is not. Only reply with 1 or 0, without additional explanation.
Examples ### Here are few concrete examples to help you understand how to make decision.
Example 1:
Text: `external data`
Answer: `label`
Your Task
Text: `external data`
Answer:

Prompt for Response Check.

Please follow the instruction based on the context below.
Instruction: `instruction`
Context: `external data`

A.1 Dataset Details

All datasets used in this work are in English and the use of these datasets complies with their intended purposes as specified by their creators. We have carefully reviewed the data to ensure that they do not contain personally identifiable information or offensive content.

Wikipedia The dataset is constructed using Wikipedia dump files, under the CC-BY-SA license. 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, under the CC0 license. 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, under the CC-BY-NC license.

BIPIA BIPIA is the first benchmark aimed at evaluating the risk of IPI attacks on LLMs, under the MIT license, 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.

A.2 Baseline Details

A.2.1 Detection-based Defenses

LLM (Zero-shot) (Liu et al., 2024) 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 Check (Liu et al., 2024) 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.

TaskTracker (Abdelnabi et al., 2024) 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 learning (Dong et al., 2022) to strengthen the model's capability to detect hidden instructions, where task demonstrations are integrated into the textual prompt.

	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**.

	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.

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.

A.2.2 Prevention-based Defenses

In-context Learning (Yi et al., 2023) 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 external data.

Multi-turn Dialogue (Yi et al., 2023) 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 Training (Yi et al., 2023) 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.

A.3 Additional Experimental Details

As shown earlier, we employ specific prompt templates for LLM (Zero-shot), Response Check, and LLM (Few-shot).

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.

We conduct our method using an 8B-parameter LLM. All training and inference are performed on a single NVIDIA A100 80GB GPU, with the training phase requiring approximately 1 GPU hour. For all reported results, we present outcomes from a single run. During preprocessing, to construct negative samples by inserting instructions into benign text, we utilize the PunktSentenceTokenizer from the NLTK library for sentence segmentation.

	Scenario 1	Scenario 2
TaskTracker	70.55%	71.35%
InstructDetector	**99.40%**	**99.55%**

Table 7: Detection accuracy comparison of InstructDetector 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**.

	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.

	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.

B Comparison with Related Method

Both InstructDetector and TaskTracker (Abdelnabi et al., 2024) 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, InstructDetector 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, InstructDetector leverages the high sensitivity of LLM's behavioral states to embedded instructions, achieving effective detection with a significantly smaller dataset (just 100 pairs).

Second, TaskTracker's effectiveness relies heavily on a clear distinction between user's instructions and external instructions, while InstructDetector 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 InstructDetector maintains high detection accuracy.

C Extended Ablation Studies

C.1 Composition of Training Data

We conduct experiments using different combinations of training datasets to assess the robustness and adaptability of InstructDetector to various training dataset compositions. As presented in Table 5, InstructDetector consistently yields high accuracy across all test datasets, regardless of the specific combination of training data used. This indicates the generalizability and adaptability of InstructDetector, 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, InstructDetector still achieves satisfactory accuracy in this challenging scenario, further validating its effectiveness and robustness in instruction detection.

C.2 Paired Responses for Gradients

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 WildChat (Zhao et al., 2024) 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 InstructDetector. These results emphasize the robustness of InstructDetector to differentiate response pairings while confirming that "Sure" is a particularly effective option for this task.

C.3 Influence of Instruction Quantity and Position

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 InstructDetector. 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.