Turn a Blind Eye: Defending Indirect Prompt Injection via Oracle-Guided Self Attention Alignment

Anonymous ACL submission

Abstract

Agents with tool calling are vulnerable to Indirect Prompt Injection, where malicious instructions embedded in untrusted data hijack the model's control flow. Instead of relying on external supervision or extensive adversarial training, we propose **Oracle-Guided Self Attention Alignment** in this work. Our method leverages the model's own behavior under "Oracle Conditions"—where malicious instructions are masked—as the ground truth. By forcing the model's attention patterns on corrupted data to mimic this oracle state, we effectively teach the model to ignore the indirect prompt injections.

1 Introduction

The rise of Agentic AI allows LLMs to use tools, but introduces *Indirect Prompt Injection* (IPI): adversaries embedding malicious instructions in retrieved data to hijack control flow. Current defenses fall into three main categories: (1) **Prompt-based** methods (Hines et al., 2024; Schulhoff), cost-effective but fragile against adaptive attacks; (2) **Architecture-based** methods (Wu et al., 2024; Liu et al., 2025), which physically segregate data but incur high deployment costs; and (3) **Tuning-based** methods (Chen et al., 2025a,b), utilizing adversarial training or latent interventions.

Within tuning, Representation Engineering (RepE) approaches (Hung et al., 2025; Zhang et al., 2024; Shi et al., 2025) typically focus on *boosting* attention to user instructions. However, we argue this overlooks a more robust path: leveraging the model's intrinsic capacity to ignore noise. We propose a paradigm shift to **Oracle-Guided Self Attention Alignment**. Instead of externally suppressing signals, we guide the model to *restore* its native attention dynamics—using its behavior under safe, clean conditions (the "Oracle" state) as the ground truth.

Our framework comprises two key contributions:

- **Signal-Based Head Identification:** We employ Signal Detection Theory (Macmillan, 2002) on clean data to pinpoint "instructional heads" specific to command fetching, avoiding the noise of adversarial gradients.
- **Oracle-Guided Self Attention Alignment:** We fine-tune these heads via self attention alignment, teaching them to mimic the Oracle's masked attention patterns and treat data-embedded instructions as non-executable noise.

Experiments on TopicAttack (Chen et al., 2025c) show our method significantly outperforms SOTA defenses (e.g., SecAlign) with negligible parameter updates ($\sim$0.2%). Crucially, when applied to **native tool tokens**, it achieves near-perfect defense (11.22% ASR), validating the power of architectural alignment.

2 Methodology

Our framework is designed to immunize agents against Indirect Prompt Injection (IPI) by aligning the model's attention dynamics with its intrinsic behavior under safe conditions. Rather than imposing external constraints.

2.1 Problem Formulation

Consider an agentic interaction where the input sequence X comprises a user instruction I_u, the tool execution output D (encapsulated by tool separators T_{start}, T_{end}), and potentially a hidden attacker instruction I_a. The input is formalized as:

$$X = [I_u, T_{start}, D \oplus I_a, T_{end}] \quad (1)$$

where $\oplus$ denotes insertion at an arbitrary position. In a standard IPI attack, the model generates a response Y following I_a rather than I_u. Our objective is to optimize the model parameters θ such that the conditional probability $P(Y|X)$ remains invariant to the presence of I_a, ensuring the tool output is treated strictly as passive data.

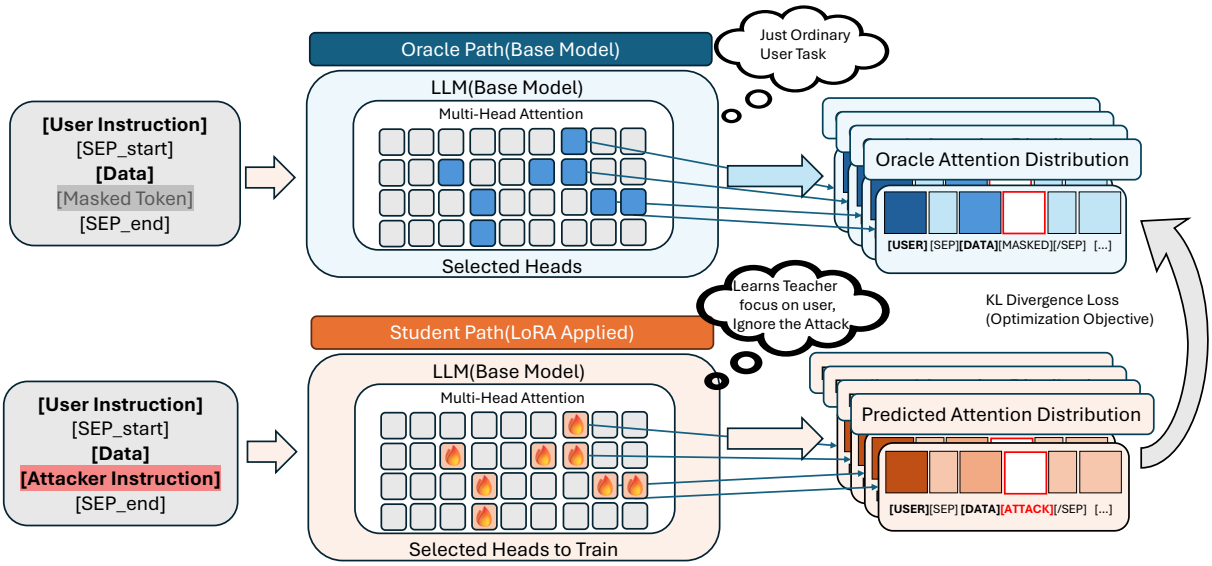

Figure 1: Overview of **Oracle-Guided Self-Alignment**. The student model's specific instructional heads learn to mimic the oracle's masked attention pattern, effectively rendering the attacker instruction embedded between separators transparent to the decision-making process.

To validate generalization, we investigate two separator configurations:

- **Prompt-Based Separators:** Explicit text tags (e.g., <data>) commonly used in standard RAG prompts.
  In this experiment, We use ordinary token sequence <data> to verify our approach can make the model recognize our predefined boundary. But in real-world application, we should always use special token as the boundary, which should be always removed by ReAct framework
- **Native Tool Separators:** Special tokens pretrained into tool-calling capable models (e.g., Llama-3.1-Instruct, Qwen3), which we hypothesize possess stronger structural priors.

2.2 Signal-Based Head Identification

Existing approaches to improving instruction following and robustness can be broadly categorized into: (1) attention steering methods that directly modify attention weights (e.g., PASTA, INSTABOOST), and (2) head selection and adaptation methods that identify and selectively optimize a subset of attention heads (e.g., FocalLoRA).

Our method belongs to the second category, but differs fundamentally in how instructional heads are identified and utilized for security purposes.

Unlike prior head identification methods (e.g., FocalLoRA), which rely on adversarial or conflict-driven signals, our approach formulates head identification as a signal detection problem based solely on instruction-following data via a two-stage process: scoring and thresholding.

Diagnostic Scoring. We first construct a diagnostic dataset consisting of clean instruction-data pairs (I_u, D). To decouple semantic sensitivity from positional bias, we create inputs $X_{diag} = D \oplus I_u$ where the insertion position is randomly chosen. We define a ground truth binary mask M_{pos} indicating token ownership by I_u. For each attention head h, we calculate the ROC-AUC score between its attention distribution and M_{pos}, prioritizing heads that consistently distinguish instructions from data. We define the Instructional Score S_h as the Lower Confidence Bound (LCB) of the AUC to penalize variance:

$$S_h = \mu(\text{AUC}_h) - \lambda_{LCB} \cdot \sigma(\text{AUC}_h) \quad (2)$$

where λ_{LCB} controls the penalty strength.

Threshold Selection via Probe Data. To determine the optimal subset $\mathcal{H}^*$, we utilize a secondary probe dataset following the SEP protocol (Zverev et al., 2025). Let $A(r)$ denote the Attack Success Rate (ASR) when the top $r \in [0, 1]$ ratio of ranked heads are masked on $D \oplus I_a$ section. We seek to cover the majority of the model's vulnerability window with minimal intervention. We define the optimal cutoff ratio r^* as the earliest point where the cumulative area under the ASR curve saturates a threshold α (e.g., 0.9):

Attack Methods	Qwen2-7B-Instruct						Llama3.1-8B-Instruct					
	None	Sandwich	Spotlight	StruQ	SecAlign	Ours	None	Sandwich	Spotlight	StruQ	SecAlign	Ours
Naive	70.67	30.56	60.78	12.78	0.56	**0.44**	64.44	27.67	33.11	**0.11**	2.78	**0.11**
Ignore	80.11	33.11	63.67	11.22	0.22	**0.11**	77.56	23.67	54.00	1.11	4.22	**0.11**
Escape	78.89	34.11	67.44	11.11	1.33	**1.00**	76.67	39.11	46.89	0.22	4.11	**0.44**
Fakecom	96.78	52.67	97.22	78.56	**0.44**	36.15	85.78	30.89	88.56	46.22	1.89	**0.72**
Combined	92.00	52.00	96.00	82.78	0.56	**0.44**	84.00	42.22	88.33	56.00	1.67	**0.22**
TopicAttack	99.22	**68.56**	99.44	99.22	92.00	95.77	96.44	79.67	92.67	98.22	90.67	**65.77**

Table 1: The ASR results of attack methods against different defense methods on Prompt-Based Separator setup, evaluated with Inj-SQuAD dataset. Bold indicates the lowest ASR. All the results are reported in %.

$$r^* = \min \left\{ r \in [0,1] \;\middle|\; \frac{\int_0^r A(x)\,dx}{\int_0^1 A(x)\,dx} \geq \alpha \right\} \quad (3)$$

Heads within the top r^* ratio constitute our target set $\mathcal{H}^*$ for the model training stage.

2.3 Oracle-Guided Self Attention Alignment

In this step, We train the query (Q) and key (K) on selected heads for it's corresponding attention layer in previous step for student model.

The attention layer going to align is selected in step "Signal-Based Head Identification". We scoring attention heads which sensitive to instruction following sentence based on instruction tuning dataset

To restore the intrinsic safety of $\mathcal{H}^*$, we introduce an **Oracle Consistency Loss**. Our training signal is self-referenced: the teacher and student share the same base model, differing only in whether injected instructions are visible. We define two views: a *Student View* containing the full attack, and an *Oracle View* where the attack is neutralized.

In the Oracle View, we replace the tokens of I_a with padding tokens ($\vec{\varnothing}$) and apply the standard attention mask to these positions. This effectively removes the attack's semantic signal while preserving the sequence length and positional structure:

$$\begin{aligned} X_{stu} &= [I_u, T_{start}, D \oplus I_a, T_{end}] \\ X_{ora} &= [I_u, T_{start}, D \oplus \vec{\varnothing}, T_{end}] \end{aligned} \quad (4)$$

We freeze the model backbone and apply Low-Rank Adaptation (LoRA) specifically to the Query and Key projections of the selected heads $\mathcal{H}^*$ in student model. We minimize the KL divergence between the attention distributions of selected heads in student model, and corresponding heads teacher model :

$$\mathcal{L} = \sum_{h \in \mathcal{H}^*} D_{KL}(\mathbf{A}_h(X_{ora}) \parallel \mathbf{A}_h(X_{stu})) \quad (5)$$

This forces the student's instructional heads to mimic the oracle, learning to treat the visible attack tokens I_a as if they were invisible padding, effectively rendering the injection transparent to the model's decision-making process.

We focus on aligning the query (Q) and key (K) representations because the attention logits are determined solely by their scaled dot-product. Specifically, the attention weights are computed as

$$A = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right), \quad (6)$$

where A denotes the attention weight matrix. Since prompt injection primarily manifests as abnormal shifts in attention distribution, as observed in prior work, directly modifying Q and K allows us to reshape the attention weights and counteract such shifts.

In contrast, the value (V) does not affect the computation of attention weights; it only influences the final weighted aggregation once the attention distribution has been determined. Therefore, fine-tuning Q and K provides the most direct and computationally efficient way to regulate attention behavior, while leaving V unchanged helps preserve model utility.

3 Experiments

3.1 Setup and Implementation

Training Data Construction. To ensure our method learns generalized structural boundaries rather than overfitting to specific content, we utilize the whole *inj-TrivalQA* dataset from (Chen et al., 2025d) for training. We construct training samples (I_u, D, I_a) using two representative attack

templates: *Naive* and *Ignore* proposed by (Perez and Ribeiro, 2022). Following Section 2.3, we synthesize the Student View X_{stu} (containing attacks) and Oracle View X_{ora} (masked) to guide the self attention alignment.

Evaluation Protocol. We evaluate defense performance on a separate held-out dataset, *inj-SquadQA* from (Chen et al., 2025d), to test generalization. We report Attack Success Rate (ASR) across a comprehensive suite of attacks: *Naive, Ignore, Escape (Breitenbach et al., 2023), Fakecom (Willison, 2023), Combined (Liu et al., 2024),* and the context-aware *TopicAttack* (Chen et al., 2025c).

3.2 Main Result: Defense Effectiveness

We assess defense performance under two distinct separator configurations: standard textual separators and native tool tokens.

Setting 1: Prompt-Based Separator. Following standard protocols, we first employ explicit textual tags (T_{start} = `<Data Area>`, T_{end} = `</Data Area>`). As shown in Table 1, our method achieves a new state-of-the-art. Against the most challenging *TopicAttack*, where baselines like *StruQ* fail (98.22%) and *SecAlign* struggles (90.67%), our method significantly reduces the ASR to **65.77%** on Llama-3. Furthermore, we maintain near-zero ASR (<1%) on standard injection types (Naive, Ignore, etc.), demonstrating that our "Oracle-Guided" alignment provides superior robustness compared to adversarial training methods like SecAlign.

Setting 2: Native Tool Separator. To validate our framework in modern agentic workflows, we extend experiments to models with native tool-calling capabilities. Table 2 reveals a critical insight: aligning with native tokens is far more effective than textual tags. For Llama-3.1, applying our method reduces the TopicAttack ASR to a negligible **11.22%**, compared to 65.77% in the prompt-based setting. We attribute this to the *Structural Prior* hypothesis: pre-trained models already possess distinct attention patterns for special control tokens. By anchoring our Oracle mechanism to these native structures, we effectively "harden" the model's inherent boundary between executable instructions and data.

3.3 Analysis of Head Identification

A key hypothesis of this work is that **Signal-Based (SDT)** identification locates vulnerable mechanisms more precisely than activation-based baselines (e.g., FocalLoRA). To evaluate the effectiveness of our head identification strategy, we compare against FocalLoRA, which provides a representative head ranking based on conflict sensitivity. This comparison isolates the impact of head selection, independent of downstream optimization.

We analyze this via Figure 2:

Efficiency: Steeper ASR Reduction. Figure 2 (Left) plots ASR reduction against the ratio of masked heads. Our method (Blue) exhibits a much steeper drop than FocalLoRA (Orange). This indicates we successfully pinpoint the "true" instructional heads earlier, allowing us to achieve robust defense by modifying fewer parameters.

Safety: Utility Preservation. Figure 2 (Center) illustrates model utility under varying masking ratios. FocalLoRA shows rapid utility decay, suggesting its identified heads overlap significantly with general reasoning capabilities. In contrast, our curve remains nearly flat. This confirms that our selection disentangles "instruction-fetching" from "reasoning," ensuring that disabling these heads does not degrade the agent's intelligence.

Attack Methods	Utility			ASR		
	None	Sandwich	Ours	None	Sandwich	Ours
Naive	59.77	**75.88**	75.55	20.33	5.11	**0.55**
Ignore	26.33	68.00	**74.77**	41.55	10.11	**0.11**
Escape	54.33	**75.44**	**75.44**	22.11	4.55	**0.44**
Fakecom	31.27	71.94	**74.99**	31.27	7.83	**0.66**
Combined	13.88	56.88	**76.44**	57.44	21.44	**0.33**
TopicAttack	22.22	66.22	**71.44**	66.55	17.00	**11.22**

Table 2: The Utility and ASR results of attack methods against different defense methods on Native Tool Separator setup, evaluated with Inj-SQuAD dataset on Llama-3.1-8B-Instruct . Bold indicates the lowest ASR or best Utility. For the given user instruction and attacker instruction, if the model answers user instruction correctly, we count it into Utility. If the model follows attacker instruction instead, we count it in Attack Success Rate . All results are reported in %

4 Discussion and Conclusion

We introduced **Oracle-Guided Self-Alignment**, shifting the defense paradigm from boosting instructions to restoring intrinsic attention dynamics. Empirically, our method significantly outperforms the SOTA baseline (SecAlign) on TopicAttack, reducing ASR from 90.67% to **65.77%** in prompt-based settings. Despite being trained solely on

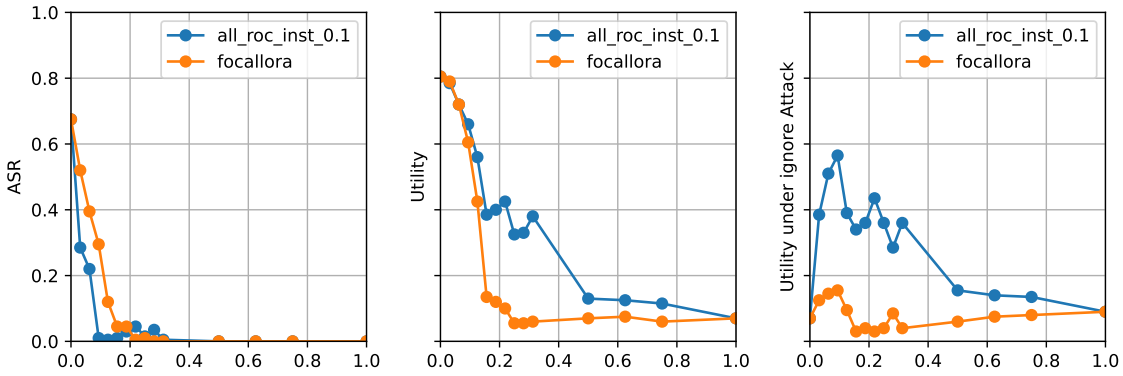

Figure 2: **Head Identification Analysis.** (Left) ASR drop curve; steeper indicates higher precision in identifying vulnerable heads. (Center) Utility retention curve; flatter indicates better safety. (Right) Utility under attack. Our Signal-Based method (Blue) defends more efficiently while preserving significantly more utility than FocalLoRA (Orange).

Scenario	**ASR AUC** (↓)		**Utility AUC** (↑)	
	Focal.	**Ours**	Focal.	**Ours**
Clean	-	-	0.1534	**0.2638**
Naive	0.0399	**0.0173**	0.1059	**0.2563**
Ignore	0.0554	**0.0316**	0.0738	**0.2322**
Escape	0.0550	**0.0258**	0.0933	**0.2478**

Table 3: Area Under Curve (AUC) analysis of head identification. **ASR AUC** (lower is better) measures defense efficiency; **Utility AUC** (higher is better) measures capability retention across masking ratios. "Focal." denotes FocalLoRA sensitivity score.

simple templates (Naive and Ignore), our model demonstrates strong generalization against unseen, complex attacks.

Although FocalLoRA is not designed for prompt injection defense, it provides a strong baseline for evaluating head-level selection strategies, which is the focus of our method.

Crucially, applying our framework to Llama-3's native tool tokens drops ASR to a negligible **11.22%**. This confirms that pre-trained models possess latent structural priors distinguishing data from instructions, yet require alignment to resist semantic jailbreaks. We conclude that robust agentic safety lies in identifying and reinforcing these *architectural* boundaries, rather than imposing arbitrary textual constraints.

Limitations

While Oracle-Guided Self-Alignment demonstrates robust defense, we acknowledge several limitations:

- **Dependency on Separator Integrity:** Our method assumes that the boundaries of untrusted data blocks are correctly delineated. It does not prevent "separator spoofing" where a user or attacker injects the separator tokens themselves to disrupt parsing. Therefore, secure deployment requires a system-level input sanitizer to ensure these structural tokens are immutable and cannot be forged within the context.
- **Architectural Constraints:** Our Signal-Based Head Identification relies on analyzing explicit attention maps. Its applicability to architectures without standard attention mechanisms (e.g., SSMs like Mamba or RWKV).
- **Modality Scope:** This work addresses textual Indirect Prompt Injection. As agents evolve into multimodal systems, investigating whether visual or audio inputs can hijack similar "instructional heads" presents a critical frontier for future research.
- **Dual-Use Concerns:** The mechanism of "teaching a model to ignore specific text segments" could theoretically be repurposed by malicious actors to suppress legitimate safety guardrails or system prompts. However, our method requires a clean "Oracle" state as a reference. As long as the alignment process is supervised by benign developers using safe data, the risk of accidental safety degradation is minimized.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceed-*

ings of the 24th International Conference on Machine Learning, pages 33–40.

Mark Breitenbach, Adrian Wood, Win Suen, and Po-Ning Tseng. 2023. Don't you (forget nlp): Prompt injection with control characters in chatgpt. Accessed: 2026-01-06.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. 2025a. {StruQ}: Defending against prompt injection with structured queries. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 2383–2400.

Sizhe Chen, Arman Zharmagambetov, Saeed Mahloujifar, Kamalika Chaudhuri, David Wagner, and Chuan Guo. 2025b. Secalign: Defending against prompt injection with preference optimization. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 2833–2847.

Yulin Chen, Haoran Li, Yuexin Li, Yue Liu, Yangqiu Song, and Bryan Hooi. 2025c. TopicAttack: An indirect prompt injection attack via topic transition. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7327–7345, Suzhou, China. Association for Computational Linguistics.

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

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*.

Kuo-Han Hung, Ching-Yun Ko, Ambrish Rawat, I-Hsin Chung, Winston H Hsu, and Pin-Yu Chen. 2025. Attention tracker: Detecting prompt injection attacks in llms. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2309–2322.

Ruofan Liu, Yun Lin, Zhiyong Huang, and Jin Song Dong. 2025. Drip: Defending prompt injection via token-wise representation editing and residual instruction fusion. *arXiv preprint arXiv:2511.00447*.

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.

Neil A Macmillan. 2002. Signal detection theory. *Stevens' handbook of experimental psychology: Methodology in experimental psychology*, 3:43–90.

Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Sander Schulhoff. Sandwich defense.

Zitong Shi, Guancheng Wan, Haixin Wang, Ruoyan Li, Zijie Huang, Wanjia Zhao, Yijia Xiao, Xiao Luo, Carl Yang, Yizhou Sun, and Wei Wang. 2025. Don't forget the enjoin: FocalloRA for instruction hierarchical alignment in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Simon Willison. 2023. Delimiters won't save you from prompt injection. Accessed: 2026-01-06.

Tong Wu, Shujian Zhang, Kaiqiang Song, Silei Xu, Sanqiang Zhao, Ravi Agrawal, Sathish Reddy Indurthi, Chong Xiang, Prateek Mittal, and Wenxuan Zhou. 2024. Instructional segment embedding: Improving LLM safety with instruction hierarchy. In *Neurips Safe Generative AI Workshop 2024*.

Qingru Zhang, Chandan Singh, Liyuan Liu, Xiaodong Liu, Bin Yu, Jianfeng Gao, and Tuo Zhao. 2024. Tell your model where to attend: Post-hoc attention steering for LLMs. In *The Twelfth International Conference on Learning Representations*.

Egor Zverev, Sahar Abdelnabi, Soroush Tabesh, Mario Fritz, and Christoph H. Lampert. 2025. Can LLMs separate instructions from data? and what do we even mean by that? In *The Thirteenth International Conference on Learning Representations*.

A Appendix A

We present our training detial here:

A.1 Native Tool Separator

For Llama-3.1-8B-Instruct and Qwen-3-8B. We align the separator with their respective chat templates:

- **Llama-3.1:**

T_{start} = <|start_header_id|>ipython <|end_header_id|>

T_{end} = <|eot_id|>

In practice, we apply the chat template directly with tool role without additional preprocessing

A.2 The hardware and hyperparameter

We uses a Nvidia RTX Pro 6000 for model training with these hyperparameters

$$\text{lr} = 5 \times 10^{-4}, \quad \beta_1 = 0.9, \quad \beta_2 = 0.999$$

$$\epsilon = 1 \times 10^{-8}, \quad \lambda_{LCB} = 0.1$$

A.3 Dataset

We utilize whole SEP (Zverev et al., 2025) for head identify, utilize the whole *inj-TrivalQA* dataset from (Chen et al., 2025d) for training and whole inj-SQuAD for evaluation.