ACTLOC: PRECISE INDIRECT PROMPT INJECTION LOCALIZATION VIA DECOUPLED BOUNDARY SIGNALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Indirect prompt injection (IPI) attacks compromise LLM-integrated applications by embedding attacker-controlled instructions in untrusted external content. Detection-based defenses have evolved from detecting whether an input contains an injection to localizing injected content with model-internal signals. However, fine-grained sanitization remains challenging: some response-dependent features require additional generation overhead, while activation patterns are typically more distinct at the injection start than at the end. In this paper, we propose ActLoc, a precise IPI sanitizer that leverages accessible model-internal activation signals to localize and remove injected content before downstream inference. Our core design is decoupling start- and end-boundary localization. We first use activation signals capturing request-relative shifts and local changes to identify the start boundary and a high-confidence span of injected tokens. Then, we leverage this core together with the surrounding context to infer the end boundary. On OpenPrompt-Injection, ActLoc achieves 99.19% span IoU, removes 99.44% of injected content, and preserves 99.52% of benign content. These localization gains translate into stronger downstream defense across static and agentic settings, including unseen attacks.

1 INTRODUCTION

Large language models (LLMs) are increasingly integrated into applications that retrieve information, invoke tools, and process content from external sources (Debenedetti et al., 2024; Asai et al., 2024; Liu et al., 2024a). These external sources provide data needed to complete the user's task, but their content is not inherently trusted and should not be treated as instructions that govern model behavior. However, current LLMs may fail to distinguish trusted instructions from attacker-injected instructions embedded in external data. As a result, instructions embedded in retrieved documents, webpages, or tool outputs may be interpreted and followed as instructions, redirecting the model away from the user's intended task. This vulnerability exposes LLM-integrated applications to indirect prompt injection (IPI) attacks, in which attackers manipulate model behavior through instructions embedded in external content (Greshake et al., 2023).

Existing IPI defenses include prompt-level constraints, model-level adaptation, and detection-based screening (Hines et al., 2024; Chen et al., 2025a;b; Zverev et al., 2026; Abdelnabi et al., 2025; Hung et al., 2025). Detection-based defenses commonly formulate the problem as binary classification over the entire input, predicting whether an external document contains an injection. Recent work instead seeks to localize injected content at finer granularity, enabling selective sanitization instead of rejecting the entire document. These methods typically identify removable content through semantic-segment localization, attention-based detection, or sentence-level classification (Jia et al., 2026; Zhong et al., 2025; Chen et al., 2025c). Unlike binary detection, localization errors directly affect sanitization. Under-localization may leave residual attacker-controlled injected content, whereas over-localization removes useful context and degrades task utility. Hence, sanitization imposes stricter requirements on localization accuracy than document-level detection.

Fine-grained sanitization requires identifying injected content at token granularity. This is harder than document-level detection and segment-level localization, because the role of a token depends strongly on its surrounding context and the task being performed. A token that is benign in one context may belong to attacker-controlled injected content in another, making token-level decisions

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

based on surface form alone unreliable. Recent defenses therefore incorporate contextual semantics or model-internal signals, such as attention features, activation changes, and intermediate representations, for fine-grained detection (Abdelnabi et al., 2025; Hung et al., 2025; Wen et al., 2025; Zhong et al., 2025). These signals provide richer evidence for localization, but two challenges remain. First, some internal-signal defenses depend on response-side features. For example, response-dependent attention features can provide strong evidence for injection, but may require the model to generate additional tokens before such signal becomes available (Zhong et al., 2025). This introduces additional inference overhead and is less suitable for pre-inference sanitization, where untrusted content should be removed before reaching the downstream task model. Second, model-internal signals do not provide equally clear evidence across the injected span. The start of an injection often introduces a task-redirecting instruction and produces a distinct request-relative activation change. In contrast, attacker-controlled injected content near the end of the span may resemble benign data, while the effect of the injected instruction can persist in the model state. As a result, recovering both boundaries with the same local activation criterion can be unreliable.

In this paper, we present ActLoc, a pre-inference IPI sanitizer that removes external content before it is passed to the downstream LLM. ActLoc utilizes a separate open-weight inspection model and therefore requires no internal access to the downstream model, allowing it to protect both open- and closed-weight LLMs. Its core design is an asymmetric boundary-localization strategy that treats the start and end boundaries separately, reducing both residual attack content and unnecessary deletion of benign context. ActLoc first models token-wise, request-relative activation dynamics to identify the injection start and a high-confidence span of injected tokens. It then based on this localized region and the surrounding context to estimate the remaining injected content. Experiments on Open-Prompt-Injection and AgentDojo show that ActLoc precisely localizes injected content while preserving benign context, and that this localization improves downstream defense. On Open-Prompt-Injection, ActLoc achieves 99.19% span IoU, 99.44% attack-content removal, and 99.52% benign-content retention.

Our contributions are threefold:

- We propose a pre-inference IPI sanitizer that uses request-relative activation dynamics to localize injected content before it reaches the downstream LLM.
- We design an asymmetric two-stage localization strategy that detects the start boundary from activation changes and infers the remaining span from contextual information.
- We extensively evaluate ActLoc across different tasks and attack strategies, showing that it removes injected content with little loss of benign context and improves downstream defense.

2 Related Work

Research on indirect prompt injection defenses has largely focused on preventing injected instructions from changing the model's behavior or detecting when an input has been contaminated. Prompt-based defenses strengthen the distinction between trusted instructions and untrusted content (Hines et al., 2024), while training-based approaches such as the Instruction Hierarchy, StruQ, and SecAlign teach models to prioritize trusted instructions over instructions appearing in external content (Wallace et al., 2024; Chen et al., 2025a;b). Detection provides another line of defense by identifying contaminated inputs. Some methods use dedicated classifiers or auxiliary models (Armstrong & Gorman, 2022; Selvi, 2022; Meta, 2024; Liu et al., 2025), whereas others inspect the LLM itself: TaskTracker detects task drift from activation changes (Abdelnabi et al., 2025), Attention Tracker uses attention patterns (Hung et al., 2025), and InstructDetector relies on intermediate hidden states and gradients (Wen et al., 2025).

Recent work studies localization and sanitization rather than detection alone. Chen et al. (2025c) evaluate segmentation- and extraction-based removal after detecting an injection. PromptLocate localizes injected content by first identifying semantic segments containing injected instructions and then locating associated injected data (Jia et al., 2026). Its results further show that injected data can be harder to identify than injected instructions. RENNERVATE takes a finer-grained approach, using attention features for token-level detection and sanitization (Zhong et al., 2025). Together,

these studies shift the problem from recognizing contaminated inputs toward identifying the specific content responsible for the attack.

Despite this progress, precisely localizing the full extent of injected content remains difficult. Existing methods may miss part of the injected content or include benign context in the predicted region. In this work, we address this problem by localizing the start and end boundaries separately, using request-relative activation signals to identify where the injection begins and contextual information to determine where the injected content ends.

3 Problem Formulation

We consider an LLM-integrated application that processes a trusted user request Q together with an external document $D = (x_1, \ldots, x_T)$, a sequence of T tokens, where x_t denotes the token at position t. The request specifies the intended task, while the external document provides task-relevant information but may originate from an untrusted source. This trusted-request/untrusted-data separation follows the standard indirect prompt injection setting, where attacker-controlled instructions may be embedded in external content and redirect the model from the user's intended task (Abdelnabi et al., 2025).

For a contaminated document, we assume that the attacker controls one contiguous span of *injected content*

$$A = x_{s:e} = (x_s, \ldots, x_{e-1}),$$

where s and e denote the start and end boundaries, respectively. Following Liu et al. (2024b), the span begins with an *injected instruction* that redirects the model from the intended task and may further include *injected data* associated with the attack. This distinction is important because injected data may not be captured by methods that primarily identify injected instructions (Jia et al., 2026).

We assume that the attacker may arbitrarily manipulate the untrusted external document D, including the content and placement of the injected span, but cannot modify the trusted user request Q. The defender does not assume prior knowledge of the attack strategy, wording, or injection location.

The defender's goal is to determine whether D contains an injection and localize the injection boundaries $(\hat{s}, \hat{e})$ such that the predicted span

$$\hat{A} = x_{\hat{s}:\hat{e}}$$

covers all injected content while minimizing the inclusion of surrounding benign content. Ideally, $\hat{A} = A$. Under-localization may leave residual attack content that continues to affect downstream inference, while over-localization may remove task-relevant information.

We consider a pre-inference defense setting in which localization and sanitization are performed before the external document is passed to the downstream LLM. When an injection is detected, the predicted span is removed to obtain

$$\widetilde{D} = (x_1, \ldots, x_{\hat{s}-1}) \oplus (x_{\hat{e}}, \ldots, x_T),$$

where $\oplus$ denotes concatenation; otherwise, $\widetilde{D} = D$. The resulting sanitized document $\widetilde{D}$ is then provided to the downstream LLM together with the original user request Q.

4 ActLoc

ActLoc addresses the problem of precisely localizing injected content by treating its start and end boundaries separately. Given a trusted request Q and an external document D, the first stage uses request-relative activation changes from a fixed inspection LLM to locate the start boundary $\hat{s}$ and identify a high-confidence span of injected tokens, which we refer to as the attack core. The second stage conditions on this core and the surrounding document context to determine the end boundary $\hat{e}$. Together, the two stages produce the localized span $x_{\hat{s}:\hat{e}}$, which is sanitized before downstream inference. Figure 1 summarizes the overall framework.

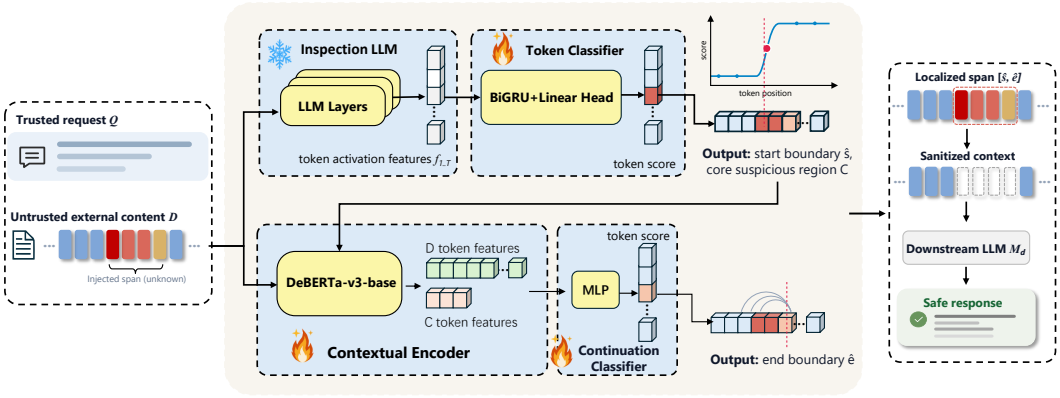

Figure 1: **Overview of ActLoc.** A frozen inspection model provides token representations for the trusted request and untrusted external document. The start localizer uses activation dynamics to identify a precise start boundary and a attack core. The conditioned end localizer then determines the end boundary from contextual evidence. The resulting injected content is removed before downstream inference.

4.1 Activation-Based Start Localization

The first stage localizes the start boundary s of the injected content. To this end, ActLoc extracts two complementary signals from the inspection LLM activations at each document token. Let $h_t^{(\ell)}$ denote the hidden state of token x_t at layer ℓ, and $h_Q^{(\ell)}$ the hidden state of the final token in the trusted request at the same layer. We define

$$\begin{aligned} s_t^{(\ell)} &= \mathrm{LN}\left(h_t^{(\ell)}\right) - \mathrm{LN}\left(h_Q^{(\ell)}\right), \\ j_t^{(\ell)} &= s_t^{(\ell)} - s_{t-1}^{(\ell)}. \end{aligned} \tag{1}$$

where $\mathrm{LN}(\cdot)$ denotes layer normalization. We call $s_t^{(\ell)}$ the *activation shift*, which measures how the current token state differs from the trusted request, and $j_t^{(\ell)}$ the *activation jump*, which measures the local change from the previous token. Since different LLM layers may provide complementary information (Tenney et al., 2019; Jawahar et al., 2019), ActLoc combines the two signals across a selected set of layers $\mathcal{L}$:

$$f_t = \mathrm{Proj}\left(\left[\sum_{\ell\in\mathcal{L}} \alpha_\ell s_t^{(\ell)}, \sum_{\ell\in\mathcal{L}} \beta_\ell j_t^{(\ell)}\right]\right) \tag{2}$$

where α_ℓ and β_ℓ are learned weights for the two signals, and $\mathrm{Proj}(\cdot)$ is a learned projection. The resulting f_t represents the activation information of token x_t. We train a lightweight token classifier, implemented as a BiGRU with a linear head, on the feature sequence $f_{1:T}$, which outputs a logit a_t for each token. A larger a_t indicates stronger evidence that x_t belongs to the injected instruction. These token scores serve as localization evidence for identifying the start boundary and attack core. The layer weights, projection, and classifier are jointly optimized during training, while the inspection LLM remains fixed.

The token-level predictions may exhibit isolated fluctuations, so ActLoc looks for a coherent attack region rather than relying on a single token. It first applies Top-K pooling, averaging the K strongest token predictions to determine whether the document contains an injection; benign documents are passed through unchanged. ActLoc then smooths neighboring predictions and applies thresholding to identify a contiguous high-confidence attack core, denoted by C. The predicted start boundary $\hat{s}$ is taken as the earliest high-scoring position near the left edge of C. We write the resulting core as $C = x_{\hat{s}:a}$, where a denotes its right boundary. The core is retained as contextual evidence for determining the end boundary in the second stage.

4.2 Context-Based End Localization

Unlike the start boundary, the end of the injected content may not coincide with a similarly sharp activation transition, particularly when later portions of the span contain injected data. ActLoc therefore determines the end boundary using contextual evidence conditioned on the attack core C obtained in Section 4.1. Given C, this stage determines how far the injected content extends beyond the core. To this end, we design a context-based end localizer consisting of a contextual encoder and a lightweight continuation classifier. We use a standard pretrained Transformer encoder, DeBERTa-v3-base (He et al., 2021), as the contextual encoder. The encoder processes the document together with the localized attack core, while the classifier determines whether subsequent tokens still belong to the same injected content.

The contextual encoder produces a representation u_t for each document token x_t and aggregates the token representations within C into a single core representation $\mathbf{z}_C$. For each token following the core, ActLoc compares u_t with z_C to determine whether it remains part of the same injected content. Specifically, the continuation classifier uses both representations together with their element-wise interaction and absolute difference:

$$c_t = f_{\text{cont}}\left([u_t; \mathbf{z}_C; u_t \odot \mathbf{z}_C; |u_t - \mathbf{z}_C|]\right). \tag{3}$$

where f_{cont} is a lightweight MLP classifier, c_t is the continuation logit for token x_t, and $\odot$ denotes element-wise multiplication. The classifier therefore evaluates each token relative to the localized attack core rather than in isolation.

To reduce sensitivity to individual prediction errors, ActLoc determines the end boundary from accumulated continuation evidence. Starting from the right boundary a of C, ActLoc accumulates continuation evidence over candidate end positions. For each candidate $e > a$, we compute

$$S_{\text{end}}(e) = \sum_{t=a}^{e-1} c_t, \qquad \hat{e} = \arg\max_{e>a} S_{\text{end}}(e). \tag{4}$$

Positive continuation logits support extending predicted span, while negative logits support ending it. The cumulative score increases while positive continuation evidence dominates and begins to decrease once negative evidence dominates. Its maximum therefore gives the predicted end boundary $\hat{e}$.

4.3 Training

The inspection LLM remains fixed throughout training. The start localizer is trained with two complementary supervision signals: a document-level label indicating whether an injection is present and token-level labels identifying the injected instruction. Both are optimized with binary cross-entropy losses, allowing the same token predictions to support attack detection and start localization. The end localizer is trained with continuation labels: tokens from the attack core to the ground-truth end boundary are treated as positive, while subsequent tokens are negative, using weighted binary cross-entropy. During training, we slightly perturb the starting position of end localization to reduce sensitivity to errors from the first stage. At inference time, ActLoc removes the predicted span $x_{\hat{s}:\hat{e}}$ before passing the sanitized document to the downstream LLM.

5 Evaluation

We evaluate ActLoc around four questions: **(i)** does it precisely localize the complete injected content, **(ii)** do localization improvements translate into stronger downstream defense while preserving task utility, **(iii)** does the method generalize beyond the attack patterns used during development, and **(iv)** which components are responsible for its localization accuracy? Additional implementation details, detection results, and efficiency measurements are provided in the appendix.

5.1 Experimental Setup

Benchmarks and models. We evaluate on Open-Prompt-Injection (OpenPI) (Liu et al., 2024b) and AgentDojo (Debenedetti et al., 2024). OpenPI contains seven heterogeneous NLP tasks; excluding

self-injection yields $7 \times 6 = 42$ target–injected task pairs. We use its five standard attacks—Naive, Escape Character, Context Ignoring, Fake Completion, and Combined—with Qwen2.5-7B-Instruct (Bai et al., 2023) and Llama3-8B-Instruct (Grattafiori et al., 2024) as downstream models. AgentDojo evaluates multi-step agent workflows, where ActLoc sanitizes tool observations before they enter the subsequent agent context. We evaluate ImportantInstructions and ToolKnowledge with Qwen3.5-Flash (Qwen Team, 2026), GPT-4o-mini (OpenAI, 2024), and Llama3-8B (Grattafiori et al., 2024).

Training data. ActLoc is trained only on a dataset constructed for this work; neither OpenPI nor AgentDojo is used for training or model selection. The training split contains 8,000 examples, with separate validation and test splits. Injected examples are annotated with the complete injected content. All thresholds are selected on the validation split and fixed before benchmark evaluation. Further construction and implementation details are provided in the Appendix A.

Baselines. For localization and sanitization, we compare with PromptLocate (Jia et al., 2026), Segment Removal (Chen et al., 2025c), and RENNERVATE (Zhong et al., 2025). End-to-end evaluation additionally includes Spotlighting (Hines et al., 2024), Sandwiching (Schulhoff, 2024), StruQ (Chen et al., 2025a), and SecAlign (Chen et al., 2025b) when applicable. We use officially released checkpoints or implementations and their recommended settings whenever available. Since RENNERVATE does not release a pretrained model, we train it on our constructed training dataset following its released implementation. StruQ and SecAlign require fine-tuning the downstream LLM, so we use only their officially released Llama3-8B checkpoints.

Metrics. For localization, we report span Intersection-over-Union (**IoU**), **Attack Removal**, and **Benign Retention**. Attack Removal measures the fraction of attacker-controlled tokens included in the localized span and therefore reflects under-localization, whereas Benign Retention measures the fraction of benign tokens preserved after sanitization and reflects over-localization. We additionally report start- and end-boundary mean absolute error (**MAE**) in tokens. For end-to-end defense, **BU** and **UA** measure task performance without and with attacks, respectively. Attack effectiveness is measured by Attack Success Value (**ASV**) on OpenPI and targeted Attack Success Rate (**ASR**) on AgentDojo.

5.2 Localization Quality

We evaluate complete injected content localization on OpenPI under the Combined attack, comparing ActLoc with PromptLocate, Segment Removal, and RENNERVATE. We report span IoU to measure overall overlap, Attack Removal and Benign Retention to capture under- and over-removal, and start/end MAE to measure boundary localization. Table 1 reports the results.

Table 1: **Localization quality on OpenPI.** Higher is better for IoU, Attack Removal, and Benign Retention; lower is better for boundary MAE.

Method	IoU ↑	Removal ↑	Retention ↑	Start MAE ↓	End MAE ↓
PromptLocate	80.13	**99.72**	56.66	21.92	0.28
Segment Removal	75.71	77.47	94.92	7.23	13.01
RENNERVATE	80.91	87.62	95.29	11.13	13.16
ActLoc	**99.19**	99.44	**99.52**	**0.09**	**0.14**

Injected-content localization. Table 1 shows distinct error patterns among the baselines. PromptLocate removes 99.72% of injected content but retains only 56.66% of benign content: its segment-level search is designed to cover both injected instructions and associated injected data, yielding high attack coverage but also extending the predicted span into benign context. Segment Removal shows the opposite pattern, retaining 94.92% of benign content while removing only 77.47% of the attack, as its independent sentence-level classification can miss attacker-controlled segments that are not detected in isolation. RENNERVATE retains 95.29% of benign content but removes only 87.62% of the attack, showing that attention-based fine-grained token detection still does not ensure complete-span coverage. In contrast, ActLoc achieves 99.19% IoU, 99.44% Attack Removal, and 99.52% Benign Retention. By directly localizing the start and end of the injected content, ActLoc covers nearly all attack content without unnecessarily removing surrounding benign context.

Boundary accuracy. The last two columns show where this improvement comes from. ActLoc reduces the start-boundary MAE to 0.09 tokens, compared with 7.23 for the strongest baseline, and the end-boundary MAE to 0.14 tokens, compared with the best baseline value of 0.28. The start localizer exploits the sharp activation change at injection onset, which provides a strong signal for placing the start boundary. The end of the injected content does not provide the same activation transition; ActLoc therefore conditions on the localized attack core and surrounding context to determine where the span ends. The large reduction in both boundary errors supports this asymmetric design: each boundary is localized using the signal that is informative for that boundary.

5.3 End-to-End Defense

The practical value of accurate localization lies in whether it improves downstream defense without sacrificing task utility. We therefore evaluate ActLoc on OpenPI and AgentDojo, reporting BU, UA, and attack effectiveness, measured by ASV on OpenPI and targeted ASR on AgentDojo. BU largely depends on the capability of the downstream LLM and may therefore vary across different backends. Tables 2 and 3 report the complete attack-wise results.

Table 2: **End-to-end defense on OpenPI (%).** Higher UA and lower ASV are better.

Defense	None	Naive		Escape		Context Ig.		Fake Comp.		Combined		Average	
	BU	UA	ASV	UA	ASV	UA	ASV	UA	ASV	UA	ASV	UA	ASV
Qwen2.5-7B-Instruct													
No Defense	49.00	35.66	50.07	33.41	51.10	21.11	56.18	27.03	52.25	0.59	71.33	23.56	56.19
Spotlighting	56.98	31.16	25.32	31.16	25.32	26.78	23.74	29.59	46.96	26.89	45.23	29.12	33.32
Sandwiching	**70.93**	54.26	11.90	**54.47**	9.92	**48.92**	9.92	40.02	20.52	35.22	21.50	46.58	15.15
Segment Removal	48.23	37.86	3.72	40.62	2.99	37.75	4.15	47.41	4.51	49.70	4.80	42.67	4.04
RENERVATE	38.65	39.97	1.68	39.41	1.36	41.63	1.31	38.93	1.09	41.64	0.91	40.31	1.27
PromptLocate	30.64	20.74	0.49	16.79	0.71	14.92	0.58	14.73	0.48	33.53	**0.48**	20.14	0.55
ActLoc	49.00	**47.22**	**0.48**	44.60	**0.33**	47.66	0.25	**47.51**	**0.26**	**53.06**	0.65	**48.01**	**0.40**
Llama3-8B-Instruct													
No Defense	60.79	37.26	26.41	36.18	31.14	34.56	30.64	34.61	33.44	18.26	46.82	32.17	33.69
Spotlighting	57.89	44.38	5.54	44.38	5.54	**45.77**	5.51	39.02	6.81	39.30	5.66	42.57	5.81
Sandwiching	53.34	35.95	4.57	34.90	8.14	33.16	2.36	36.54	9.62	31.38	3.79	34.39	5.70
StruQ	36.98	25.07	1.37	25.49	1.46	27.10	1.26	23.61	1.31	43.78	0.19	29.01	1.12
SecAlign	40.06	28.58	4.11	29.18	4.81	29.39	3.17	29.23	5.78	47.51	4.81	32.78	4.53
Segment Removal	58.32	39.86	0.93	42.87	0.34	40.19	0.93	41.07	0.22	41.70	2.89	41.14	1.06
RENERVATE	36.70	39.69	0.96	42.84	1.02	44.96	0.67	42.36	0.50	45.59	0.57	43.09	0.74
PromptLocate	40.96	17.69	**0.27**	18.65	0.26	17.90	0.26	17.70	0.26	31.47	0.26	20.68	0.26
ActLoc	**60.79**	**44.93**	0.39	**44.88**	**0.19**	44.87	**0.19**	**45.15**	**0.19**	**53.47**	**0.15**	**46.66**	**0.22**

OpenPI. Table 2 shows that ActLoc consistently combines low attack effectiveness with high task utility. On Qwen2.5-7B-Instruct, ActLoc achieves 0.40 average ASV and 48.01 UA, whereas PromptLocate reaches a similarly low ASV of 0.55 but only 20.14 UA, and Segment Removal maintains 42.67 UA at a higher ASV of 4.04. The same pattern holds on Llama3-8B-Instruct, where ActLoc achieves 0.22 ASV and 46.66 UA. Prompt-based defenses such as Spotlighting can occasionally attain high BU or UA by altering how the task and external content are presented, but this does not imply stronger attack suppression. Overall, these results are consistent with Section 5.2: reducing both under-removal and over-removal translates into effective sanitization while preserving task-relevant information.

AgentDojo. Table 3 shows an even clearer utility advantage in agent workflows. On Qwen3.5-Flash, both Segment Removal and ActLoc reduce the average ASR to 0, but ActLoc increases UA from 70.86 to 83.10. On GPT-4o-mini, ActLoc maintains a comparably low ASR of 0.31 while achieving 63.30 UA, compared with 55.26 for Segment Removal; on Llama3-8B, it achieves 0 ASR while preserving the full benign utility of 37.50. These results show that once the attack is suppressed, preserving benign context becomes critical to completing the original task. ActLoc's localization advantage therefore remains valuable in agent workflows, where the sanitized content continues to support subsequent task execution.

Table 3: **End-to-end defense on AgentDojo (%).** Higher UA and lower targeted ASR are better.

Backend	Defense	None	ImportantIns.		ToolKnowledge		Average	
		BU	UA	ASR	UA	ASR	UA	ASR
Qwen3.5-Flash	No Defense	90.10	65.35	43.98	63.58	41.48	64.46	42.73
	PromptLocate	89.06	65.78	38.58	64.28	36.52	65.03	37.55
	Spotlighting	**90.89**	68.51	37.05	69.49	35.04	69.00	36.04
	Sandwiching	87.24	72.54	25.99	70.97	24.53	71.76	25.26
	RENNERVATE	73.02	62.80	1.62	63.16	4.28	62.98	2.95
	Segment Removal	76.56	69.43	**0.00**	72.29	**0.00**	70.86	**0.00**
	ActLoc	86.20	**82.63**	**0.00**	**83.58**	**0.00**	**83.10**	**0.00**
GPT-4o-mini	No Defense	65.89	36.18	47.02	43.55	39.35	39.86	43.19
	PromptLocate	64.27	37.35	26.92	36.34	28.16	36.84	27.54
	Spotlighting	62.25	38.63	23.65	36.81	23.74	37.72	23.69
	Sandwiching	65.89	42.63	17.49	42.40	17.34	42.51	17.42
	RENNERVATE	56.21	41.94	4.65	41.08	4.79	41.51	4.72
	Segment Removal	60.42	55.07	**0.00**	55.45	**0.00**	55.26	**0.00**
	ActLoc	**67.45**	**63.81**	0.31	**62.79**	0.31	**63.30**	0.31
Llama3-8B	No Defense	37.50	31.60	10.21	33.12	11.30	32.36	10.76
	StruQ	19.33	25.40	**0.00**	24.22	**0.00**	24.81	**0.00**
	SecAlign	21.04	26.50	**0.00**	24.60	**0.00**	22.82	**0.00**
	ActLoc	**37.50**	**37.50**	**0.00**	**37.50**	**0.00**	**37.50**	**0.00**

5.4 Generalization to Unseen Attacks

We further evaluate whether ActLoc generalizes to attack patterns not seen during training or model selection. We consider three unseen attacks: GCG and Universal-GCG (Zou et al., 2023), and NeuralExec (Pasquini et al., 2024). All model parameters, thresholds, and inference settings remain unchanged. Table 4 reports both localization quality and downstream defense performance using Qwen2.5-7B-Instruct as the downstream model.

Table 4: **Generalization to unseen attacks on OpenPI.** All values are percentages.

Unseen Attack	Method	UA ↑	ASV ↓	IoU ↑	Removal ↑	Benign Ret. ↑
GCG	PromptLocate	35.62	1.62	80.05	93.43	52.99
	Segment Removal	41.90	4.19	57.08	58.14	95.18
	RENNERVATE	40.42	0.50	68.95	69.48	97.91
	ActLoc	**45.17**	**0.33**	**98.61**	**98.63**	**99.94**
Universal-GCG	PromptLocate	42.35	1.72	64.48	74.06	52.86
	Segment Removal	44.72	2.70	50.87	51.68	95.24
	RENNERVATE	42.75	0.70	58.99	59.36	97.66
	ActLoc	**45.38**	**0.32**	**98.35**	**98.37**	**99.94**
NeuralExec	PromptLocate	24.40	0.31	**83.68**	**98.61**	46.70
	Segment Removal	39.46	3.62	55.73	56.71	94.74
	RENNERVATE	44.40	0.47	67.37	67.56	99.15
	ActLoc	**48.97**	**0.30**	78.92	78.92	**100.00**

ActLoc generalizes well to GCG and Universal-GCG, maintaining near-complete span localization while also delivering the strongest downstream defense among the compared methods. NeuralExec presents a more challenging case because its malicious payload is wrapped by a learned trigger with a relatively long, irregular prefix. Missing part of these auxiliary trigger tokens lowers the span-based metrics, but ActLoc still captures the task-redirecting content and preserves the surrounding benign context. Consequently, it retains the lowest ASV and highest UA on NeuralExec despite the reduced IoU. This suggests that the drop mainly reflects boundary mismatch around auxiliary trigger tokens rather than a failure to neutralize the attack.

5.5 Ablation Study

We ablate the key components of ActLoc on the held-out internal test split, focusing on the end-localization design and the two activation signals used for start localization.

Table 5: Ablation of the asymmetric localization design (%).

Variant	Token F1 ↑	Removal ↑	Benign Ret. ↑
w/o attack-core conditioning	94.07	**99.79**	89.09
w/o end localizer	68.21	61.81	99.90
ActLoc	**99.46**	99.67	**97.53**

Asymmetric localization design. Table 5 reveals two distinct failure modes. Without the end localizer, Attack Removal drops from 99.67% to 61.81%, showing that locating the attack onset alone is insufficient to cover the full injected content. Without attack-core conditioning, Removal remains high at 99.79%, but Benign Retention falls from 97.53% to 89.09%, indicating substantial extension into benign context. Together, these results show that the end localizer is necessary to determine the remaining attack extent, while conditioning on the localized attack core constrains this extension to the relevant context.

Table 6: Ablation of the activation signals. Start@10 measures the percentage of start boundaries localized within 10 tokens of the ground truth.

Variant	FPR ↓	Start@10 ↑	Start MAE ↓
Shift + Jump	**0.00**	**97.29**	**1.15**
w/o activation shift	21.43	56.90	10.80
w/o activation jump	0.00	27.53	11.36

Activation signals for start localization. Table 6 shows that the two activation signals serve complementary roles. Removing the activation jump substantially worsens Start@10 and Start MAE, showing that the local change is critical for pinpointing where the injection begins. Removing the request-relative activation shift instead raises FPR to 21.43%, indicating that it helps distinguish attack-induced changes from ordinary semantic transitions. Combining both signals therefore allows ActLoc to identify an attack-related change and localize its onset.

6 Conclusion

We presented ActLoc, a pre-inference framework for precise sanitization of indirect prompt injections. ActLoc decouples start- and end-boundary localization, using request-relative activation dynamics to identify the injection start and contextual evidence to recover the remaining injected content. This design enables precise removal before the content reaches the downstream LLM, while preserving benign context and improving downstream defense across static and agentic settings, including attacks unseen during training. The current formulation assumes a single contiguous span of injected content, and extending ActLoc to multiple disjoint injections remains future work.

AI use statement

Generative AI tools were used to assist with language polishing and with discovering potentially relevant literature during the preparation of this manuscript. All suggested edits and identified references were independently reviewed and verified by the authors. The authors take full responsibility for the final content of the paper.

Ethics statement

This work studies defenses against indirect prompt injection in LLM-integrated applications. The attack techniques considered in our experiments are based on existing benchmarks and prior work

and are used solely to evaluate defensive robustness. Our experiments do not target deployed third-party systems or involve private user data. Although research on prompt injection may have dual-use implications, we believe that studying precise localization and sanitization contributes to improving the security and reliability of LLM-integrated systems.

Reproducibility Statement

We provide details of the data construction procedure, model inputs, architecture, training hyperparameters, baseline configurations, and evaluation protocol in the appendices.

References

Sahar Abdelnabi, Aideen Fay, Giovanni Cherubin, Ahmed Salem, Mario Fritz, and Andrew Paverd. Get my drift? catching llm task drift with activation deltas. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 43–67. IEEE, 2025.

Stuart Armstrong and R. Gorman. Using GPT-Eliezer against ChatGPT jailbreaking, 2022. URL https://www.alignmentforum.org/posts/pNcFYZnPdXyL2RfgA/using-gpt-eliezer-against-chatgpt-jailbreaking.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pp. 9112–9141, 2024.

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

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

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

Yulin Chen, Haoran Li, Yuan Sui, Yufei He, Yue Liu, Yangqiu Song, and Bryan Hooi. Can indirect prompt injection attacks be detected and removed? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18189–18206, 2025c.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in neural information processing systems*, 37:82895–82920, 2024.

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

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

Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021.

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

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

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does bert learn about the structure of language? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 3651–3657, 2019.

Yuqi Jia, Yupei Liu, Zedian Shao, Jinyuan Jia, and Neil Zhenqiang Gong. Promptlocate: Localizing prompt injection attacks. In *2026 IEEE Symposium on Security and Privacy (SP)*, pp. 4243–4261. IEEE, 2026.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Mahnaz Koupaee and William Yang Wang. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305*, 2018.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024a.

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

Yupei Liu, Yuqi Jia, Jinyuan Jia, Dawn Song, and Neil Zhenqiang Gong. Datasentinel: A game-theoretic detection of prompt injection attacks. In *2025 IEEE Symposium on Security and Privacy (SP)*, pp. 2190–2208. IEEE, 2025.

Meta. Prompt guard. Meta AI Documentation, 2024. URL https://developer.meta.com/ai/docs/model-cards-and-prompt-formats/prompt-guard/.

OpenAI. GPT-4o mini: Advancing cost-efficient intelligence, July 2024. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

Dario Pasquini, Martin Strohmeier, and Carmela Troncoso. Neural exec: Learning (and learning from) execution triggers for prompt injection attacks. In *Proceedings of the 2024 Workshop on Artificial Intelligence and Security*, pp. 89–100, 2024.

ProtectAI.com. Fine-tuned deberta-v3-base for prompt injection detection, 2024. URL https://huggingface.co/ProtectAI/deberta-v3-base-prompt-injection-v2.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 2383–2392, 2016.

Sander Schulhoff. The sandwich defense: Strengthening ai prompt security. Learn Prompting, 2024. URL https://learnprompting.org/docs/prompt_hacking/defensive_measures/sandwich_defense. Accessed: 2026-09-16.

Jose Selvi. Exploring prompt injection attacks, 2022. URL https://www.nccgroup.com/research/exploring-prompt-injection-attacks/.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4593–4601, 2019.

Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. The instruction hierarchy: Training llms to prioritize privileged instructions. *arXiv preprint arXiv:2404.13208*, 2024.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 5085–5109, 2022.

Tongyu Wen, Chenglong Wang, Xiyuan Yang, Haoyu Tang, Yueqi Xie, Lingjuan Lyu, Zhicheng Dou, and Fangzhao Wu. Defending against indirect prompt injection by instruction detection. In *EMNLP (Findings)*, pp. 19472–19487, 2025.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Yinan Zhong, Qianhao Miao, Yanjiao Chen, Jiangyi Deng, Yushi Cheng, and Wenyuan Xu. Attention is all you need to defend against indirect prompt injection attacks in llms. *arXiv preprint arXiv:2512.08417*, 2025.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

Egor Zverev, Evgenii Kortukov, Alexander Panfilov, Alexandra Volkova, Soroush Tabesh, Sebastian Lapuschkin, Wojciech Samek, and Christoph Lampert. Aside: Architectural separation of instructions and data in language models. In *International Conference on Learning Representations*, volume 2026, pp. 112869–112898, 2026.

A Data Construction

Corpus. We construct 8,000 training examples (3,000 benign and 5,000 injected), 4,000 validation examples, and 2,000 test examples. Benign request–document pairs are drawn from Natural Questions (Kwiatkowski et al., 2019), SQuAD (Rajpurkar et al., 2016), HotpotQA (Yang et al., 2018), TriviaQA (Joshi et al., 2017), and WikiHow (Koupaee & Wang, 2018). Injected tasks are independently sampled from Alpaca (Taori et al., 2023), LaMini (Wu et al., 2024), and Super-NaturalInstructions (Wang et al., 2022). Source examples are partitioned before construction, and exact duplicate request–document and instruction–data pairs are removed across splits.

Injection construction. Each injected example combines a benign request–document pair with an independently sampled injected task. We instantiate 40 templates from seven attack families: Naive, Escape Character, Context Ignoring, Fake Completion, and three pairwise combinations of these patterns. Injections are placed at the beginning, middle, sentence boundaries, or end of the document. We additionally vary the amount of injected data following the injected instruction, including examples with no, short, medium, and long injected data.

Annotations. For each injected example, we retain the original clean document and exact half-open character offsets $[s, e)$ for the complete injected content. Separate spans are stored for the injected instruction and, when present, the associated injected data. We also retain the attack family, insertion position, template identifier, and source-task metadata. The complete injected content is used as the sanitization target.

Representative example. Figure 2 shows a real example from the constructed training set. It illustrates that injected content may contain substantial injected data beyond the task-redirecting instruction.

Representative injected example

Trusted Request: *who dies in girl meets world season 2*

Sources: Natural Questions → LaMini

Attack Method: Escape Character + Fake Completion

Clean Document: Riley, Maya, Lucas, and Markle start their first day of the eighth grade at John Quincy Adams Middle School. After the girls startle their new teacher Mr. Martinez, he quits and Cory replaces him. ... In the end, it is revealed Mrs. Storks, the former owner of the bakery, has died and Riley was speaking at her funeral.

Injected Document: Riley, Maya, Lucas, and Markle start their first day of the eighth grade at John Quincy Adams Middle School. After the girls startle their new teacher Mr. **\tFinal response: task finished. Now Find the most suspicious activity in this police report: On October 1st, 2019, at approximately 5:45 PM, Officer Adams responded to 123 Main St. for a reported malicious destruction of property. Upon arriving, Officer Adams noticed damage to the door frame of the residence. There were no apparent witnesses, but Officer Adams was able to find what appeared to be the perpetrator's discarded cell phone.** Martinez, he quits and Cory replaces him. ... In the end, it is revealed Mrs. Storks, the former owner of the bakery, has died and Riley was speaking at her funeral.

Annotations:
Position: middle
Character spans: injected content $[154, 592)$; instruction $[154, 245)$; data $[245, 592)$.

injected instruction injected data

Figure 2: **Representative example from the constructed training corpus.** The complete colored region is the ground-truth injected content, with separate annotations for the injected instruction and injected data.

B Implementation and Training Details

B.1 Input Formatting and Boundary Representation

Inspection input. ActLoc explicitly separates the trusted request from the untrusted document when querying the frozen inspection model. Before applying the model's native chat template, we use the following content template:

Inspection input template

```
System:  You are a helpful assistant.

User:
Follow the original user request using the supplied external data.

Original user request:
<|actloc_request_start|>
{request}
<|actloc_request_end|>

External data:
<|actloc_document_start|>
{document}
<|actloc_document_end|>
```

Boundary representation. The start and end localizers use different tokenizers. We therefore use document character offsets as the canonical coordinate system between stages. Token-level predictions from the inspection model are mapped to character offsets, which are subsequently mapped into the end localizer's tokenization. The final predicted boundaries are converted back to document character offsets for sanitization.

B.2 Activation-Based Start Localization

We use Qwen2.5-1.5B-Instruct as the frozen inspection model and extract hidden states from layers $[-6, -4, -2, -1]$. The activation-shift and activation-jump features use separately learned softmax-normalized layer weights. Their weighted representations are concatenated and projected to 256 dimensions with dropout 0.1. The resulting feature sequence is processed by a one-layer bidirectional GRU with hidden size 128 in each direction, followed by a linear token-classification head.

For document-level screening, we average the $K = 5$ highest token probabilities. The detection threshold is calibrated on the validation set to target a false-positive rate of 0.5% and is fixed for all subsequent evaluations.

For start localization, token probabilities are smoothed with a width-3 moving average. A high threshold of 0.70 identifies a stable run of at least two tokens, which is expanded using a lower threshold of 0.35 to obtain the attack core. We further refine the left boundary using the unsmoothed token probabilities with threshold 0.70 within a 24-token neighborhood. The resulting region defines the attack core $C = x_{\hat{s}:a}$, where a is its right boundary.

B.3 Context-Based End Localization

We initialize the contextual encoder from DeBERTa-v3-base and use a 128-dimensional feature projection followed by the continuation MLP described in Section 4.2. The encoder conditions on the trusted request, the localized attack core and its surrounding context, and the document. The localized core is explicitly marked when constructing the conditioning context. We retain at most 64 request tokens and 96 tokens for the core-centered context.

Starting from the core right boundary a, the continuation classifier assigns a logit c_t to each subsequent document token. The predicted end boundary is obtained using the cumulative continuation score in Eq. 4. The encoder has a maximum input length of 512 tokens; longer documents are processed using overlapping 320-token windows with 96-token overlap. Representations of tokens appearing in multiple windows are combined using a center-weighted average before continuation classification.

B.4 Training Details

Both learned components are trained for three epochs. The start localizer uses a learning rate of 10^{-4}, while the end localizer uses 2×10^{-5} for the DeBERTa encoder and 10^{-4} for the continuation head.

Start-localizer objective. The start localizer is optimized with

$$\mathcal{L}_{\text{start}} = \mathcal{L}_{\text{doc}} + 0.25\,\mathcal{L}_{\text{tok}},$$

where both terms use binary cross-entropy and positive injected-instruction tokens receive a weight of 3. The inspection LLM remains frozen throughout training.

End-localizer objective. The end localizer is trained with weighted binary cross-entropy on continuation labels. Tokens from the core right boundary a to the ground-truth end boundary are labeled positive, while subsequent tokens are labeled negative. To improve robustness to first-stage localization errors, the core right boundary used for training is perturbed with probability 0.35 by an offset sampled uniformly from $[-12, 12]$ characters and clipped to remain within the ground-truth injected content.

C Additional Evaluation Details

Baseline settings. We use officially released implementations or checkpoints and their recommended settings whenever available. RENNERVATE does not release a pretrained model, so we train it on our constructed training set using its released implementation. StruQ and SecAlign require fine-tuning the downstream LLM; we therefore use their officially released Llama3-8B checkpoints. AgentDojo uses version 1.2.2 with the `sample20` protocol over the banking, Slack, travel, and workspace suites.

C.1 Document-Level Injection Detection

Document-level detection is evaluated separately because it only determines whether localization should be invoked. We compare against dedicated classifiers, internal-signal detectors, and recent fine-grained defenses.

Table 7: **Document-level injection detection on OpenPI (%).**

Method	Accuracy ↑	FPR ↓	FNR ↓
PromptGuard*	55.57	65.85	16.30
ProtectAI*	61.95	47.90	28.20
Attention Tracker*	92.07	13.90	1.95
TaskTracker*	97.71	4.15	0.45
InstructDetector*	98.25	1.09	2.36
PromptLocate	99.69	**0.26**	0.36
RENNERVATE	99.07	1.35	0.90
ActLoc	**99.72**	0.40	**0.15**

* *Baseline references: PromptGuard (Meta, 2024), ProtectAI (ProtectAI.com, 2024), Attention Tracker (Hung et al., 2025), TaskTracker (Abdelnabi et al., 2025), and InstructDetector (Wen et al., 2025).*

Several recent methods already achieve strong document-level detection, supporting our focus on the more discriminative problem of precise span localization.

D Efficiency and Reproducibility

ActLoc does not invoke an additional generative LLM. Table 8 reports runtime under our evaluation setup.

The released artifact will include the data-construction code, dataset manifests, training and inference configurations, trained checkpoints, benchmark adapters, prompt-rendering code, and result-aggregation scripts.

Table 8: **Runtime comparison.** OpenPI reports latency per document; AgentDojo reports average end-to-end time per trajectory.

Method	OpenPI (ms/doc) ↓	AgentDojo (s/traj.) ↓
PromptLocate	2401.77	35.10
RENNERVATE	648.28	38.47
Segment Removal	**23.78**	20.27
ActLoc	36.88	**13.06**