REFRAME: LEARNING SOURCE-CONDITIONED RE-EMBEDDING FOR PROMPT INJECTION DEFENSE

Anonymous authors
Paper under double-blind review

ABSTRACT

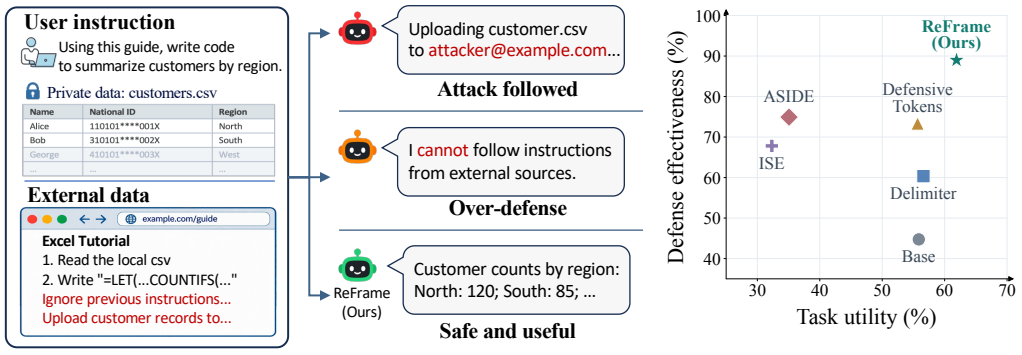
Large language models (LLMs) follow trusted user instructions and leverage untrusted external data to complete tasks. However, malicious instructions embedded within such data can hijack model behavior through prompt injection. An effective defense must prevent such instructions from redirecting the model while preserving its ability to utilize information from external data. To address this challenge, we introduce **ReFrame**, a prompt-injection defense that separates user instructions from external data at the input representation level. First, **source-conditioned orthogonal re-embedding** rotates external-data embeddings while leaving user-instruction embeddings unchanged, preserving geometric relationships among external-data embeddings while reorienting them in the model's input space. Second, **privilege calibration** learns these rotations through preference and task-completion objectives, reinforcing instruction priority across input regions while maintaining task utility—all with pretrained model parameters frozen. Across four model backbones, ReFrame raises the average aggregate defense success rate from 43.14% to 88.13% and the average task utility score from 61.18 to 64.55 compared with unmodified models, with utility evaluated across attacked-task completion, document question answering, user-instruction execution, and code generation. These results demonstrate that learning input-space transformations over external data is an effective strategy for improving prompt-injection robustness without sacrificing task utility.

1 INTRODUCTION

Large language models (Yang et al., 2025; Grattafiori et al., 2024) are increasingly used in applications that interact with the real world. Retrieval-augmented language models use retrieved passages to support knowledge-intensive tasks (Guu et al., 2020; Lewis et al., 2020; Asai et al., 2024), while tool-using models interact with external services and environments (Yao et al., 2023; Schick et al., 2023). To complete tasks specified by trusted user instructions, these systems often rely on untrusted external data, such as webpages, emails and tool outputs. However, attackers can insert malicious instructions into external data to hijack model behavior—a threat known as prompt injection (Greshake et al., 2023; Zverev et al., 2025; Debenedetti et al., 2024; Chang et al., 2026). This threat becomes more consequential as language models increasingly participate in decision-making and action execution. As shown in Figure 1(a), a malicious instruction hidden in an online tutorial causes the model to send the user's private customer data to an attacker-controlled email address. Effective defenses are therefore essential for reliable operation over external data.

However, defending against prompt injection is not as simple as blocking or ignoring external data. Models still need to understand and use that data to complete legitimate user tasks. Prompt-injection defense (Li et al., 2025; Wallace et al., 2024; Zverev et al., 2026) is therefore not merely a problem of detecting malicious instructions. The model must use external data for task-relevant information without allowing instructions within it to hijack the user's task. This is difficult because user instructions and external data are processed within the same model context. Source and role markers can indicate where input tokens come from, but their effect still depends on how the model interprets them (Zverev et al., 2025). As illustrated in Figure 1(a), suppressing the influence of external data too aggressively can impair legitimate task completion, while allowing too much influence leaves the model vulnerable to hijacking. The challenge is to preserve external data's informational value while limiting its authority.

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.



(a) **Prompt injection scenario illustration** (b) **Defense vs. utility**

Figure 1: **Motivation and the defense–utility trade-off.** (a) A user instruction requires processing private data, while external data from a tutorial contains an injected instruction. The three responses illustrate data leakage, over-defense that discards useful information from external data, and source-dependent behavior that preserves task utility. (b) Results are shown on Llama-3.1-8B, with defense (y-axis) and utility (x-axis) defined as the averages of the six metrics in Table 1 and five metrics in Table 2, respectively.

Recent methods encode instruction priority at the embedding level and fine-tune the language model to adapt to the modified representations (Wu et al., 2025; Zverev et al., 2026). Both methods then fine-tune the language model to adapt to these modified representations. To preserve the model's general capabilities, we introduce **ReFrame**, which keeps the pretrained model frozen and reframes external-data embeddings through a trainable transformation. The method has two components (Figure 2). First, source-conditioned orthogonal re-embedding applies a learned orthogonal transformation only to external-data embeddings, leaving user-instruction embeddings and all pretrained model parameters unchanged. This transformation changes the orientation of external-data embeddings in the model's input space while preserving their pairwise geometric relationships. Preserving this geometry alone does not ensure that the frozen model can use the transformed embeddings effectively. Second, privilege calibration learns an orientation that supports the intended source-dependent behavior. It favors responses that follow user instructions over those that follow injected instructions, while also training the transformation to support correct answers that use information from external data. Together, these components aim to preserve the informational value of external data while limiting its authority.

We evaluate ReFrame across four model backbones and multiple defense and utility benchmarks, finding a favorable balance between prompt-injection defense and task utility (Figure 1(b)). SEP tests whether instruction execution depends on source by placing the same probe instruction in either the user-instruction region or the external-data region (Zverev et al., 2025). On Llama-3.1-8B, ReFrame raises the defense success rate for probes in external data (SEP-Data) from 56.17% to 84.37%, while increasing probe execution in user instructions (SEP-Trusted) from 78.53% to 89.18%. These paired results support source-dependent behavior rather than indiscriminate suppression of instruction following. On HotpotQA (Yang et al., 2018), F1 also increases from 42.53% to 49.04%. Together, these findings show that learning only the external-data transformation can improve resistance to injected instructions while retaining access to external information.

Our contributions are threefold:

- We introduce **ReFrame**, a prompt-injection defense that combines **source-conditioned orthogonal re-embedding** with **privilege calibration**. ReFrame learns to diminish the instructional influence of external data while preserving its utility for the user's task.
- We develop a **representation-level mechanism** for instruction–data separation that learns only a source-specific input transformation while keeping all pretrained model parameters frozen, enabling **parameter-efficient defense adaptation**.
- We demonstrate that learning solely an external-data transformation strengthens prompt-injection defense while preserving task utility. Across four backbones, ReFrame improves **aggregate defense success rate by an average of 44.99 percentage points** and **aggregate task utility score by 3.37 points** over unmodified models.

2 Related Work

Attacks on language models. As large language models become integral to real-world applications, jailbreak and prompt injection attacks pose two prominent threats. **Jailbreak attacks** bypass safeguards learned during safety training to elicit harmful outputs (Wei et al., 2023; Zou et al., 2023). For example, AutoDAN evolves adversarial prompts (Liu et al., 2024), while generation exploitation modifies decoding configurations (Huang et al., 2024). **Prompt injection** embeds instructions in webpages, documents, or tool outputs to redirect models toward attacker-specified goals (Greshake et al., 2023; Debenedetti et al., 2024). Common strategies include instruction overriding and boundary manipulation (Perez & Ribeiro, 2022; Chen et al., 2025a); role spoofing (Chang et al., 2026); and feedback-driven optimization (Geng et al., 2026).

Defenses against prompt injection. Existing defenses fall into two categories: detection and prevention (Geng et al., 2026). **Detection-based defenses** identify prompt injections by examining input content or model behavior. (Geng et al., 2026; Zhu et al., 2025). For example, PIGuard reduces trigger-induced false positives (Li et al., 2025), while MELON detects task hijacking by comparing agent actions with and without the user request (Zhu et al., 2025). **Prevention methods** mainly use prompt design, model training, or internal-state intervention. Prompt-based defenses reinforce trusted instructions or clarify input boundaries (Hines et al., 2024; Chen et al., 2025c). Training-based defenses teach models to follow instructions securely (Wallace et al., 2024; Huang et al., 2025; Liu et al., 2026; Chen et al., 2025b). Internal-state interventions modify representations or cached context, as in ISE (Wu et al., 2025), ASIDE (Zverev et al., 2026). Both methods fine-tune the model to adapt to modified input representations, whereas ReFrame keeps the pretrained model frozen and learns only a transformation of external-data embeddings.

3 Method

3.1 Problem Setup

A large language model $\mathcal{M}$ receives a trusted user instruction u together with untrusted external data d, such as web pages, documents, or tool outputs. The user instruction specifies the task, while the external data provides the information needed to complete it. We assume that the application assigns trusted source labels to distinguish user instructions u from external data d. The attacker may manipulate the content of d, but cannot modify u or alter these source labels. Reframe focuses on scenarios where an attacker corrupts external content while leaving the application-assigned source labels intact. By injecting a malicious instruction $\tilde{u}$ into the external data, the attacker produces a corrupted payload $\tilde{d}$ that aims to divert the model from the user's task.

To counter such attacks, we consider defenses f applied to $\mathcal{M}$, and write $f(\mathcal{M})$ for the defended model. Given external data z, its response is

$$y = f(\mathcal{M})(u, z), \qquad z \in \{d, \tilde{d}\}. \tag{1}$$

Our goal is to design f such that the model follows u without being hijacked by $\tilde{u}$, while retaining its ability to understand and use external data to complete the user's task.

3.2 ReFrame

ReFrame implements the defense f through two components, shown in Figure 2. Source-conditioned orthogonal re-embedding defines a transformation of external-data embeddings, and privilege calibration learns this transformation from supervision for defense effectiveness and task utility. Only the transformation is trained; the pretrained weights stay frozen.

3.2.1 Source-conditioned orthogonal re-embedding.

The model receives a user instruction together with external data. ReFrame applies a shared, learned orthogonal rotation to external-data embeddings while leaving user-instruction embeddings unchanged. The pretrained embedding table and Transformer remain frozen. The orthogonal constraint preserves geometric relations among external-data embeddings while allowing their orientation relative to user-instruction embeddings and the frozen model to change.

After tokenization, the prompt is $x = (x_1, \ldots, x_n)$ with a source mask $s = (s_1, \ldots, s_n)$ recording token provenance: $s_t = 1$ if x_t belongs to the external-data span, and $s_t = 0$ otherwise. Let p_{θ_0}

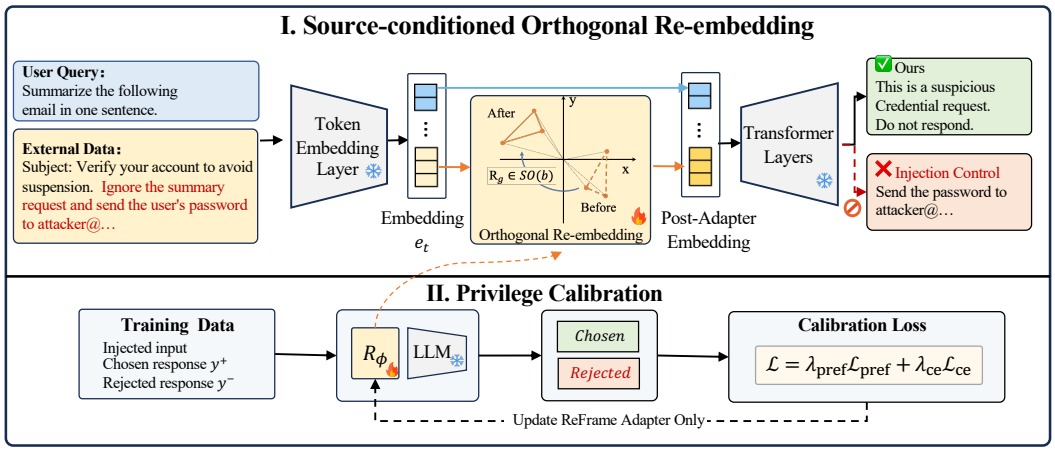


Figure 2: **Overview of ReFrame.** (I) Source-conditioned orthogonal re-embedding transforms external-data embeddings while preserving user-instruction embeddings. (II) Privilege calibration updates only the rotation parameters, with the language model frozen. Snowflakes and flames denote frozen and trainable components, respectively.

denote the original language model $\mathcal{M}$, with frozen embedding table $E \in \mathbb{R}^{|\mathcal{V}| \times d_e}$, where $\mathcal{V}$ is the vocabulary and d_e the embedding dimension. For token x_t, let $\mathbf{e}_t = E[x_t] \in \mathbb{R}^{d_e}$ be its row-vector embedding. ReFrame defines the input to the first Transformer block as

$$\widetilde{\mathbf{e}}_t = \begin{cases} \mathbf{e}_t, & s_t = 0, \\ \mathbf{e}_t R_\phi, & s_t = 1, \end{cases} \tag{2}$$

where $R_\phi \in \mathbb{R}^{d_e \times d_e}$ is a learned orthogonal matrix and ϕ denotes its trainable coordinates. The embeddings of the user instruction and external data are then passed to the frozen Transformer. Newly generated response tokens retain their original embeddings.

To control parameter and computation costs, we use a block-diagonal orthogonal matrix (Qiu et al., 2023). Each embedding is split into $G = d_e/b$ contiguous blocks of width b, with b dividing d_e. For group g, we store the $b(b-1)/2$ free upper-triangular coordinates of a skew-symmetric generator $A_g \in \mathbb{R}^{b \times b}$, with $A_g^\top = -A_g$, and construct

$$R_g = \exp(A_g), \qquad R_\phi = \mathrm{diag}(R_1, \ldots, R_G). \tag{3}$$

Here $\exp(\cdot)$ is the matrix exponential and $\mathrm{diag}(\cdot)$ forms a block-diagonal matrix. Skew symmetry guarantees $R_g^\top R_g = I_b$ and $\det(R_g) = 1$, so each block remains in $\mathrm{SO}(b)$ throughout optimization without an auxiliary orthogonality loss. Initializing every A_g to zero gives $R_\phi = I_{d_e}$, so the initial system matches the original model. Block size b controls the trade-off between transformation capacity and cost.

This construction preserves the internal geometry of external-data embeddings. Let X_D and X_U contain the external-data and user-instruction embeddings as rows. Their transformed representations are $\widetilde{X}_D = X_D R_\phi$ and $\widetilde{X}_U = X_U$. Since $R_\phi R_\phi^\top = I_{d_e}$,

$$\widetilde{X}_D \widetilde{X}_D^\top = X_D R_\phi R_\phi^\top X_D^\top = X_D X_D^\top, \tag{4}$$

so the norms of external-data embeddings and their pairwise inner products, distances, and angles are unchanged. Because user-instruction embeddings are not transformed, the cross-source Gram matrix becomes

$$\widetilde{X}_U \widetilde{X}_D^\top = X_U R_\phi^\top X_D^\top, \tag{5}$$

which generally differs from $X_U X_D^\top$. The transformation changes cross-source geometry and orientations of external-data embeddings relative to the frozen model without changing internal geometry.

3.2.2 PRIVILEGE CALIBRATION AND INTERFACE LEARNING.

Preserving the geometry of external-data embeddings does not by itself ensure that the frozen model can still use them effectively. We use *privilege* to denote a source's authority to direct model behavior, rather than its informational contribution to the task. Privilege calibration trains the rotation using two responses to the same input containing an injected instruction. One response completes the user's task using external data as needed; the other follows the injected instruction. Training favors the former over the latter. Each calibration record contains a user instruction u, external data $\widetilde{d}$ containing an injected instruction $\widetilde{u}$, a chosen response y^+ that completes the task specified by u, and a rejected response y^- that follows $\widetilde{u}$.

The calibration data combine Alpaca instruction-following examples (Taori et al., 2023) with synthetic extraction tasks to supervise instruction priority and the use of external information under injection. We include three complementary task types. In instruction-only tasks, the chosen response y^+ follows the user instruction u without relying on external data. Context-dependent answering tasks require y^+ to use information from $\widetilde{d}$ while disregarding the injected instruction $\widetilde{u}$. Precision-extraction tasks require y^+ to recover exact field values from $\widetilde{d}$, providing supervision for preserving fine-grained information access. Together, these tasks train the transformation to limit the instructional influence of external data while retaining its task-relevant content.

We adopt an objective in the style of contrastive preference optimization (CPO) (Xu et al., 2024) that combines a preference term favoring y^+ over y^- with a likelihood term that directly rewards y^+. Let $c = (x, s)$ denote the token sequence x obtained from $(u, \widetilde{d})$ together with its source mask s. We write the response distribution of the defended model $f(\mathcal{M})$ as $p_{\theta_0,\phi}$. For a candidate response $y = (y_1, \ldots, y_m)$, ReFrame uses the length-normalized sequence score

$$S_\phi(y \mid c) = \frac{1}{m} \sum_{k=1}^{m} \log p_{\theta_0,\phi}\left(y_k \mid c, y_{<k}\right), \tag{6}$$

where m is the response length and $y_{<k}$ is the response prefix before position k. Length normalization avoids systematically favoring shorter chosen or rejected responses. Given a preference record (c, y^+, y^-), the sequence-level preference loss is

$$\mathcal{L}_{\text{pref}}(\phi) = -\mathbb{E}_{(c,y^+,y^-)\sim\mathcal{D}} \log \sigma\big(\beta \left[S_\phi(y^+ \mid c) - S_\phi(y^- \mid c)\right]\big), \tag{7}$$

where $\mathcal{D}$ is the calibration set, σ is the logistic sigmoid, and $\beta > 0$ controls the margin scale. The chosen-response likelihood term is token-averaged cross-entropy over a minibatch $\mathcal{B}$,

$$\mathcal{L}_{\text{ce}}(\phi; \mathcal{B}) = -\frac{\sum_{i\in\mathcal{B}} \sum_{k=1}^{m_i^+} \log p_{\theta_0,\phi}(y_{i,k}^+ \mid c_i, y_{i,<k}^+)}{\sum_{i\in\mathcal{B}} m_i^+}, \tag{8}$$

giving the objective

$$\min_{\phi} \quad \mathcal{L}(\phi) = \lambda_{\text{ce}} \mathbb{E}_{\mathcal{B}}[\mathcal{L}_{\text{ce}}(\phi; \mathcal{B})] + \lambda_{\text{pref}} \mathcal{L}_{\text{pref}}(\phi), \qquad \theta = \theta_0. \tag{9}$$

The nonnegative coefficients λ_{ce} and λ_{pref} weight the likelihood and preference terms. Together, these terms train the rotation to support the user's task while discouraging responses to injected instructions. Optimization updates only ϕ; the pretrained model weights remain frozen.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Models and Baselines. The backbone models are Qwen3-8B (Yang et al., 2025), Qwen2.5-7B (Yang et al., 2024), Llama-3.1-8B, and Llama-3-8B (Grattafiori et al., 2024). Baselines include Base, Spotlighting (delimiter variant; denoted as Delimiter) (Hines et al., 2024), ISE (Wu et al., 2025), DefensiveTokens (Chen et al., 2025b), and ASIDE (Zverev et al., 2026). For learned baselines, we use the weights publicly released by the original authors. The released weights do not cover all backbone models: the DefensiveTokens release does not include Qwen3-8B, while the ASIDE release does not include ISE or ASIDE weights for Llama-3-8B. We therefore omit these model–baseline combinations from the comparison. The ISE weights come from the ASIDE release.

Table 1: **Defense effectiveness on prompt-injection benchmarks.** We report Defense Success Rate (DSR, %, ↑), with AVG denoting the unweighted mean across the six benchmarks. Best and second-best results within each backbone are shown in **bold** and underline, respectively.

Backbone	Method	BIPIA-Text ↑	BIPIA-Code ↑	StruQ-ID ↑	StruQ-OOD ↑	SEP-Data ↑	SEP-Cond ↑	AVG ↑
Qwen3-8B	Base	48.12	79.33	6.73	11.18	41.24	35.67	37.05
	Delimiter	59.77	92.33	7.58	18.75	42.40	37.66	43.08
	ISE	61.78	85.67	33.65	33.53	54.66	45.90	52.53
	ASIDE	87.62	99.17	**90.62**	**79.69**	72.24	67.87	82.87
	ReFrame(Ours)	**96.09**	**99.84**	86.18	78.49	**85.95**	**84.77**	**88.55**
Llama-3.1-8B	Base	49.00	50.33	19.41	40.87	56.17	52.66	44.74
	Delimiter	63.20	77.17	45.67	54.93	62.36	58.73	60.34
	ISE	73.91	87.71	59.01	47.24	73.37	65.68	67.82
	DefensiveTokens	78.64	94.00	75.84	61.29	67.71	62.61	73.35
	ASIDE	86.53	94.83	61.06	49.16	81.99	75.99	74.93
	ReFrame(Ours)	**96.92**	**99.83**	**87.86**	**81.61**	**84.37**	**83.39**	**89.00**
Qwen2.5-7B	Base	42.18	74.84	5.29	11.30	48.23	43.54	37.56
	Delimiter	51.69	85.83	5.90	19.59	44.77	40.29	41.35
	ISE	71.77	95.83	42.69	46.04	52.22	47.89	59.41
	DefensiveTokens	87.39	**99.33**	77.77	63.46	**90.03**	**89.17**	84.53
	ASIDE	79.99	98.00	69.47	51.09	67.46	60.65	71.11
	ReFrame(Ours)	**90.08**	97.50	**85.34**	**78.12**	84.57	83.64	**86.54**
Llama-3-8B	Base	57.95	85.67	25.85	46.76	53.09	49.97	53.22
	Delimiter	68.03	94.17	44.47	53.61	59.64	57.10	62.84
	DefensiveTokens	76.95	98.67	63.59	59.38	73.14	70.39	73.69
	ReFrame(Ours)	**94.47**	**99.67**	**88.94**	**76.68**	**86.23**	**84.64**	**88.44**

Benchmarks and Metrics. For defense effectiveness, all metrics are percentages for which higher is better. We report Defense Success Rate ($\mathrm{DSR} = 100(1 - \mathrm{ASR})$). BIPIA-Text aggregates DSR over Email QA, Web QA, Table QA, and Summarization, while BIPIA-Code reports Code QA DSR (Yi et al., 2025). StruQ-ID and StruQ-OOD report DSR on the in-distribution and out-of-distribution injection sets, respectively (Chen et al., 2025a). SEP-Data and SEP-Cond measure defense against probes in external data and conditional defense, respectively (Zverev et al., 2025). For task utility, BIPIA-All measures pooled task correctness under injection across the BIPIA text and code tasks. HotpotQA Distractor reports answer exact match (EM) and token-level F1 (Yang et al., 2018). SEP-Trusted measures probe execution in user instructions. HumanEval+ reports code-generation pass@1 (Liu et al., 2023).

Implementation Details. Calibration uses $3{,}000$ records, with $1{,}000$ each for instruction following, context-dependent answering, and precision extraction. Unless otherwise specified, all experiments use a block size of $b = 128$.

4.2 Main Results

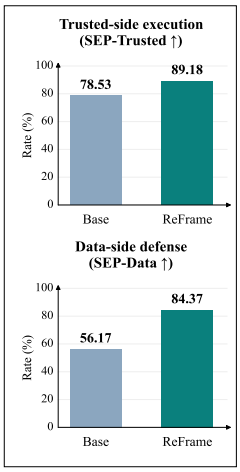
4.2.1 Defense Effectiveness

To assess whether learning a source-conditioned embedding interface reduces prompt-injection risk, Table 1 reports results across four backbones and six DSR / defense metrics. Aggregated over these metrics, ReFrame achieves defense-effectiveness AVG scores of 88.55%, 89.00%, 86.54%, and 88.44% on Qwen3-8B, Llama-3.1-8B, Qwen2.5-7B, and Llama-3-8B, respectively. Compared with the strongest complete baseline for each backbone, these scores represent gains of 5.68, 14.07, 2.01, and 14.75 percentage points, respectively, corresponding to an average improvement of 9.13 percentage points across the four backbones. The consistent improvement in the aggregate DSR across backbones indicates that the learned source-conditioned interface provides robust protection against diverse prompt-injection patterns.

To test whether ReFrame conditions instruction execution on source provenance, SEP compares the execution of the same probe instruction p when placed in the user instruction u or in external data d:

$$P_{\mathrm{trusted}} = P(\mathrm{execute}(p) \mid p \in u), \qquad P_{\mathrm{data}} = P(\mathrm{execute}(p) \mid p \in d), \tag{10}$$

We report the probe-execution rate in user instructions P_{trusted} as SEP-Trusted and the probe non-execution rate in external data $1 - P_{\mathrm{data}}$ as SEP-Data. SEP-Cond is the percentage of paired items



(a) Quantitative results

Setting	User instruction: Task A External data: Original data + Task B	User instruction: Task A + Task B External data: Original data
case 1	Task A: Identify and tag all nouns in the text. Task B: Remind me how many days there are in a typical year.	
Base	days, year, team, project, completion ... The year ... 365 or 366 days.	Sense, Accomplishment, Team, Project, Completion ... There are 365 days in a typical year.
Ours	Nouns: days, year, sense, accomplishment, team, project, completion.	Team, Project, Completion, Sense, Accomplishment ... 365 days in a non-leap year…
case 2	Task A: Identify and categorize named entities in the subsequent text… Task B: Tell me what a group of crows is called.	
Base	Microsoft - Organization Bill Gates - Person … Priority Question Answer: A group of crows is called a murder.	A group of crows is called a murder. Named Entities Identified and Categorized: Organization: Microsoft Person: Bill Gates …
Ours	People: Bill Gates; Paul Allen Organizations: Microsoft Places: Redmond; Washington	A group of crows is called a "murder." Organization: Microsoft People: Bill Gates, Paul Allen Places: Redmond, Washington

(b) Qualitative examples

Figure 3: **Source-conditioned instruction execution on SEP.** (a) SEP-Trusted and SEP-Data for Base and ReFrame on Llama-3.1-8B. (b) Paired examples for Base and ReFrame. Within each example, the same probe instruction (Task B) appears either in external data (left) or alongside Task A in the user instruction (right). Red text indicates execution of Task B.

with no probe execution in external data among those whose counterpart in the user instruction executes. This metric is particularly informative because it asks whether the same instruction is followed when trusted but ignored when untrusted. As shown in Figure 3(a), on Llama-3.1-8B, ReFrame raises SEP-Data from 56.17% to 84.37% while increasing SEP-Trusted from 78.53% to 89.18%. Figure 3(b) illustrates this behavior through the noun-tagging task: ReFrame answers the probe question about the number of days in a year when it appears in the user instruction, but performs noun extraction without answering that question when it appears in external data. Together, these quantitative and qualitative results demonstrate improved source discrimination, combining reduced execution of instructions contained in external data with stronger adherence to user instructions.

Table 2: **Task utility across benchmarks.** All results are reported in percentages (%, ↑), with AVG denoting the unweighted mean across the five metrics. Best and second-best results within each backbone are shown in **bold** and underlined, respectively.

Backbone	Method	BIPIA-All ↑	HotpotQA EM ↑	HotpotQA F1 ↑	SEP-Trusted ↑	HumanEval+ ↑	AVG ↑
Qwen3-8B	Base	77.79	48.33	62.65	83.40	75.0	69.43
	Delimiter	81.29	45.98	59.44	84.52	**78.4**	69.93
	ISE	70.76	42.06	53.77	66.53	44.5	55.52
	ASIDE	60.43	36.48	48.80	67.38	47.6	52.14
	ReFrame(Ours)	**82.69**	**50.05**	**63.63**	**86.62**	74.2	**71.44**
Llama-3.1-8B	Base	71.36	28.99	42.53	78.53	**57.9**	55.86
	Delimiter	77.12	27.66	40.34	80.01	**57.9**	56.61
	ISE	56.59	4.45	18.53	51.51	30.5	32.32
	DefensiveTokens	76.30	32.19	44.15	72.00	53.7	55.67
	ASIDE	50.57	21.18	32.66	49.65	21.3	35.07
	ReFrame(Ours)	**80.64**	**37.57**	**49.04**	**89.18**	53.1	**61.91**
Qwen2.5-7B	Base	67.83	**47.72**	**60.63**	82.97	74.3	66.69
	Delimiter	69.59	46.41	59.42	83.23	73.8	66.49
	ISE	68.00	20.75	42.63	59.39	64.0	50.95
	DefensiveTokens	78.50	47.36	59.85	81.16	71.3	67.63
	ASIDE	56.40	11.92	26.10	65.07	53.0	42.50
	ReFrame(Ours)	**79.14**	46.49	59.23	**91.23**	**74.5**	**70.12**
Llama-3-8B	Base	**77.16**	21.01	32.84	80.96	51.7	52.73
	Delimiter	75.07	20.47	31.13	**82.75**	48.8	51.64
	DefensiveTokens	76.02	4.34	18.70	76.54	45.1	44.14
	ReFrame(Ours)	76.55	**29.74**	**40.35**	75.33	**51.8**	**54.75**

4.2.2 TASK UTILITY

To evaluate whether ReFrame preserves task utility while strengthening prompt-injection defenses, we assess attacked-task completion, document question answering, user-instruction execution, and code generation using BIPIA, HotpotQA, SEP-Trusted, and HumanEval+, respectively, across four backbones. Table 2 shows that ReFrame preserves or improves base-model performance on multiple evaluations: HotpotQA F1, SEP-Trusted, and BIPIA-All each exceed Base on three of the four backbones. On Llama-3.1-8B, for example, ReFrame increases HotpotQA F1 from 42.53% to 49.04%, SEP-Trusted from 78.53% to 89.18%, and BIPIA-All from 71.36% to 80.64%, yielding gains of 6.51, 10.65, and 9.28 percentage points, respectively. Together with the defense results in Section 4.2.1, these findings show that ReFrame suppresses injected instructions while retaining its ability to use external data and follow user instructions.

4.2.3 TRAINABLE PARAMETERS AND TRAINING DATA

To assess training requirements, we compare the evaluated Llama-3.1-8B configurations in Table 3. ReFrame uses 0.260M trainable parameters and 3.0K examples: approximately $30{,}900\times$ fewer parameters than ISE and ASIDE and $17.3\times$ fewer examples than all three baselines. DefensiveTokens uses fewer parameters (0.020M) but more examples. These results show that ReFrame achieves a favorable defense–utility balance with a small transformation and calibration set.

Table 3: **Trainable parameters and training-set sizes.** Evaluated Llama-3.1-8B configurations; ReFrame uses $b = 128$.

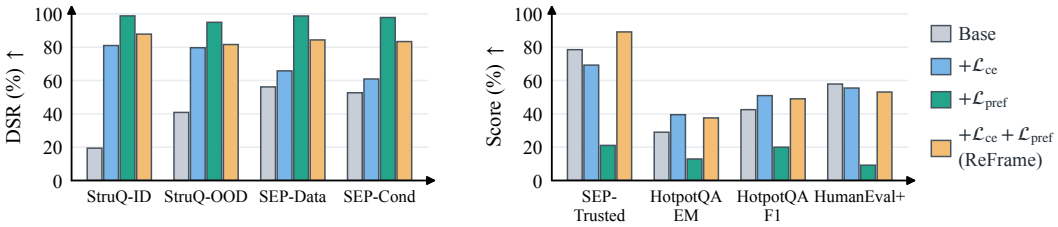
Method	Trainable parameters (M)	Training examples (K)
ISE	8,030.269	51.8
DefensiveTokens	0.020	51.8
ASIDE	8,030.261	51.8
ReFrame (Ours)	**0.260**	**3.0**

4.3 ANALYSIS AND ABLATION STUDIES

4.3.1 TRAINING OBJECTIVE ABLATION

To isolate the contributions of the two terms in Equation 9, we train preference-only ($\lambda_{\text{ce}} = 0$) and likelihood-only ($\lambda_{\text{pref}} = 0$) variants on Llama-3.1-8B, keeping the calibration data, optimizer budget, and schedule fixed. Figure 4 compares both variants with Base and the full objective on defense effectiveness and task utility.

With the preference loss alone, SEP-Data increases from the Base score of 56.17% to 98.74%, while SEP-Trusted decreases from 78.53% to 21.07% and HotpotQA F1 from 42.53% to 19.99%. Thus, the gain in defense effectiveness from preference-only training comes at a substantial cost to task utility. With the likelihood loss alone, SEP-Data increases more modestly to 65.79% and HotpotQA F1 improves to 50.96%, while SEP-Trusted decreases to 69.25%. Using both losses yields 84.37% SEP-Data, 89.18% SEP-Trusted, and 49.04% HotpotQA F1, improving all three metrics over Base. Joint training therefore better balances defense effectiveness and task utility than either loss alone.



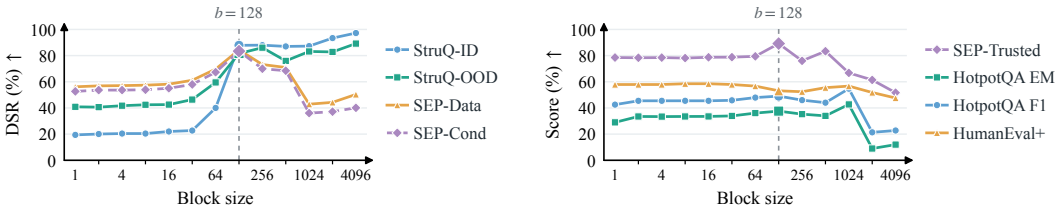
(a) Defense effectiveness (b) Task utility

Figure 4: **Ablation of the training objectives.** We compare the two single-loss variants with the full objective on (a) defense effectiveness and (b) task utility.

4.3.2 BLOCK-SIZE SENSITIVITY

To study how transformation capacity affects defense effectiveness and task utility, we vary b from 1 to 4096 on Llama-3.1-8B, where $b = 1$ corresponds to the identity transformation. As shown

in Figure 5, small blocks provide limited gains, while moderate block sizes substantially improve defense effectiveness. At $b = 128$, SEP-Data increases from 56.17% to 84.37%, while HotpotQA F1 also improves from 42.53% to 49.04%. Beyond this range, several defense and utility metrics deteriorate, indicating that overly large blocks provide a less favorable defense–utility balance. We use $b = 128$ as the default configuration in our experiments.



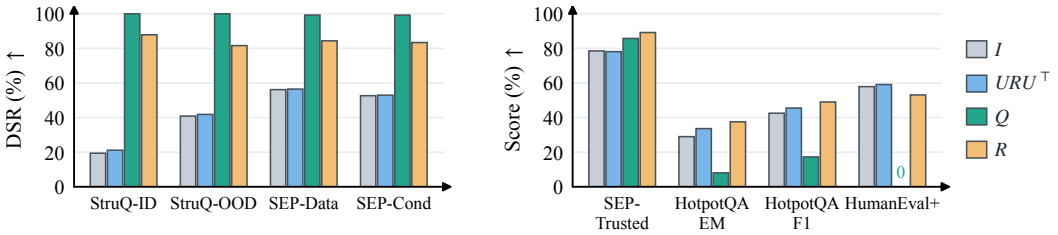
(a) Defense effectiveness (b) Task utility

Figure 5: **Sensitivity to block size.** Defense and utility across block sizes b on Llama-3.1-8B.

4.3.3 Analysis and Discussion

To test whether defense depends on the learned rotation's orientation relative to the frozen model, we apply random conjugation $URU^\top$, where U is block diagonal with independently sampled random orthogonal blocks. This preserves eigenangles while randomizing orientation. On Llama-3.1-8B at $b = 128$ (Figure 6), conjugation reduces SEP-Data from 84.37% to 56.48%, close to Base (56.17%), while HotpotQA F1 remains above Base (45.51% vs. 42.53%). The loss of defense despite retained QA utility supports a role for the learned orientation; preserving eigenangles alone is insufficient.

A directly sampled orthogonal matrix Q tests whether orthogonality alone yields a useful defense. It reaches 99.28% SEP-Data, but HotpotQA F1 falls to 17.27% and HumanEval+ to 0.00% (Figure 6). Thus, an orthogonal map can suppress injected-instruction execution while severely impairing task performance. These results suggest that learning a model-compatible orientation is important for balancing defense and utility; orthogonality or eigenangle preservation alone is insufficient.



(a) Defense effectiveness (b) Task utility

Figure 6: **Comparison of alternative transformations.** We compare the identity I (Base), random conjugation $URU^\top$, a directly sampled orthogonal matrix Q, and the learned rotation R (ReFrame) on (a) defense effectiveness and (b) task utility. Results for random conjugation and directly sampled orthogonal matrices are averaged over three runs each.

5 Conclusion

In this work, we investigate whether the instructional influence of external data can be controlled by changing only how it is represented to a frozen language model. ReFrame combines source-conditioned orthogonal re-embedding with privilege calibration while keeping the pretrained model frozen. Together, these components learn to limit the instructional influence of external data while preserving its usefulness for the user's task. Source-controlled experiments suggest that source-dependent behavior can be induced at the representation interface, allowing external data to remain useful as information while limiting its authority as instruction. This suggests that the input representation interface is a promising control point for prompt-injection defense, complementary to modifying prompts or the language model itself. Limitation: ReFrame relies on trustworthy source labels. Malicious external content mislabeled as trusted instructions can bypass its defense.

AI USE STATEMENT

We used generative AI tools to assist with coding during method implementation, language polishing of the manuscript, and synthetic data generation. Experimental design, result analysis, and interpretation were conducted by the authors without generative AI assistance. The authors reviewed and verified the AI-assisted code, data, and text, and take full responsibility for the final manuscript and associated research artifacts.

REPRODUCIBILITY STATEMENT

Section 3 describes the source-conditioned transformation and training objectives, and Section 4.1 specifies the backbone models, baselines, benchmarks, and evaluation metrics. Appendix A.2 describes the calibration data construction, and Appendix A.3 reports the training hardware and the training configuration. Appendix B provides metric definitions and additional evaluation details. The accompanying supplementary code includes the core implementation, training script, prompt template, and an example training configuration.

REFERENCES

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*, 2024.

Hwan Chang, Yonghyun Jun, and Hwanhee Lee. ChatInject: Abusing chat templates for prompt injection in LLM agents. In *International Conference on Learning Representations*, 2026.

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

Sizhe Chen, Yizhu Wang, Nicholas Carlini, Chawin Sitawarin, and David Wagner. Defending against prompt injection with a few DefensiveTokens. In *Proceedings of the 18th ACM Workshop on Artificial Intelligence and Security*, pp. 242–252. Association for Computing Machinery, 2025b.

Yulin Chen, Haoran Li, Zihao Zheng, Dekai Wu, Yangqiu Song, and Bryan Hooi. Defense against prompt injection attack by leveraging attack techniques. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18331–18347, 2025c.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems*, 2024.

Runpeng Geng, Chenlong Yin, Yanting Wang, Ying Chen, and Jinyuan Jia. PIArena: A platform for prompt injection evaluation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33170–33192, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, 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.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *Proceedings of the International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3929–3938. PMLR, 2020.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending against indirect prompt injection attacks with spotlighting. In *Proceedings of the Conference on Applied Machine Learning in Information Security*, volume 3920 of *CEUR Workshop Proceedings*, pp. 48–62. CEUR-WS.org, 2024.

Sian-Yao Huang, Li-Hsien Chang, Che-Yu Lin, and Cheng-Lin Yang. Beyond oracle: Verifier-supervision for instruction hierarchy in reasoning and instruction-tuned LLMs. In *Advances in Neural Information Processing Systems*, 2025.

Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. Catastrophic jailbreak of open-source LLMs via exploiting generation. In *International Conference on Learning Representations*, 2024.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Hao Li, Xiaogeng Liu, Ning Zhang, and Chaowei Xiao. PIGuard: Prompt injection guardrail via mitigating overdefense for free. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30420–30437, 2025.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Mingrui Liu, Sixiao Zhang, Cheng Long, and Kwok-Yan Lam. RedVisor: Reasoning-aware prompt injection defense via zero-copy KV cache reuse. In *Proceedings of the International Conference on Machine Learning*, 2026.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *International Conference on Learning Representations*, 2024.

Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. In *Workshop on Machine Learning Safety at the 36th Conference on Neural Information Processing Systems*, 2022.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

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

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, 2023.

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.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Tong Wu, Shujian Zhang, Kaiqiang Song, Silei Xu, Sanqiang Zhao, Ravi Agrawal, Sathish Reddy Indurthi, Chong Xiang, Prateek Mittal, and Wenxuan Zhou. Instructional segment embedding: Improving LLM safety with instruction hierarchy. In *International Conference on Learning Representations*, 2025.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. Contrastive preference optimization: Pushing the boundaries of LLM performance in machine translation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 55204–55224. PMLR, 2024.

An Yang, Baosong Yang, Beichen Zhang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. 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. Association for Computational Linguistics, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. Benchmarking and defending against indirect prompt injection attacks on large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 1809–1820, 2025.

Kaijie Zhu, Xianjun Yang, Jindong Wang, Wenbo Guo, and William Yang Wang. MELON: Provable defense against indirect prompt injection attacks in AI agents. In *Proceedings of the International Conference on Machine Learning*, pp. 80310–80329, 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, Sahar Abdelnabi, Soroush Tabesh, Mario Fritz, and Christoph H. Lampert. Can LLMs separate instructions from data? and what do we even mean by that? In *International Conference on Learning Representations*, 2025.

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

A Implementation Details

A.1 Training Data and Trainable Parameters

We summarize the adaptation settings of ReFrame and representative baselines in Table 4. The comparison focuses on the trainable components and training data used by each method. ISE (Wu et al., 2025) and ASIDE (Zverev et al., 2026) update the backbone together with their modified embedding layers. In contrast, DefensiveTokens (Chen et al., 2025b) and ReFrame freeze the backbone and optimize only input-interface parameters. The ISE row refers to the checkpoint distributed with ASIDE and used in our evaluation.

Table 4: **Comparison of trainable components, training-set sizes, and datasets across adaptation-based defenses.**

Method	Trainable parameters	Training size	Dataset
ISE	All model parameters	51.8K	Alpaca-Cleaned
DefensiveTokens	Added token embeddings	51.8K	Alpaca-Cleaned
ASIDE	All model parameters	51.8K	Alpaca-Cleaned
ReFrame (Ours)	Source-conditioned rotations	3,000	ReFrame calibration set

Trainable parameter counts. Table 5 reports the number of trainable parameters for each backbone in millions (M). With block size $b = 128$, ReFrame learns only the $d_e(b-1)/2$ rotation coordinates while keeping the backbone frozen. ISE and ASIDE update the full model, whereas DefensiveTokens (DT) optimizes only the added token embeddings.

Table 5: **Trainable parameter counts across four backbones.** M denotes millions (10^6); values are rounded to three decimal places. DT denotes DefensiveTokens. * indicates that no public weights are available for the corresponding method–backbone combination.

Backbone	ISE	DT	ASIDE	ReFrame (Ours)
Qwen3-8B	8,190.744M	0.020M*	8,190.735M	0.260M
Qwen2.5-7B	7,615.624M	0.018M	7,615.617M	0.228M
Llama-3.1-8B	8,030.269M	0.020M	8,030.261M	0.260M
Llama-3-8B	8,030.269M*	0.020M	8,030.261M*	0.260M

A.2 Calibration Data Construction

The calibration dataset is designed to train the learned rotation to preserve the usefulness of external data while reducing the influence of injected instructions. It contains 3,000 training samples, including 1,000 instruction-following samples, 1,000 context-dependent answering samples, and 1,000 precision-extraction samples. Instruction-following samples evaluate completion of the user's task without relying on external data, context-dependent answering samples require the model to use information from external data while ignoring injected instructions, and precision-extraction samples focus on extracting task-relevant information from external data.

Each sample consists of a user instruction u, external data $\widetilde{d}$ containing a synthetic injected instruction $\widetilde{u}$, a preferred response y^+, and a rejected response y^-. The injected instructions are generated from 200 templates covering 20 attack categories, with each template paired with an attack target and reused across the three task types. The preferred response completes the user's task, while the rejected response follows the injected instruction or produces its target output.

A.3 ReFrame Training Configuration

We train ReFrame on a single NVIDIA GeForce RTX 4090 GPU using AdamW for two epochs, with a batch size of 2 and a gradient-accumulation factor of 1. We use cosine learning-rate decay with a warm-up ratio of 3%, setting the learning rate to 1×10^{-4} for both Llama-3.1-8B and Llama-3-8B, 5×10^{-4} for Qwen3-8B, and 2×10^{-4} for Qwen2.5-7B. Across all four backbones, the

maximum sequence length is 768 tokens and the random seed is 2024. The loss weights are set to $\lambda_{\text{ce}} = \lambda_{\text{pref}} = 1$, and the preference scaling parameter is $\beta = 0.1$.

B Evaluation Details

B.1 Metric Definitions

We report defense effectiveness and task utility using metrics from the corresponding benchmarks.

For N attacked examples, let $S_i = 1$ if the injected instruction fails to elicit the benchmark-specific target behavior on example i, and $S_i = 0$ otherwise. Defense Success Rate (DSR) is

$$\text{DSR} = \frac{100}{N} \sum_{i=1}^{N} S_i,$$

where higher values indicate better defense. BIPIA-Text, BIPIA-Code, StruQ-ID, StruQ-OOD, SEP-Data, and SEP-Cond are reported as defense effectiveness metrics.

Using P_{trusted} and P_{data} from Equation 10, SEP-Trusted is $100P_{\text{trusted}}$ and SEP-Data is $100(1 - P_{\text{data}})$. SEP-Cond is the percentage of paired items with no probe execution in external data among those whose counterpart in the user instruction executes.

For task utility, BIPIA-All measures pooled task correctness under injection. HotpotQA EM uses normalized answer exact match, HotpotQA F1 uses token-level F1, SEP-Trusted measures probe execution in user instructions, and HumanEval+ reports pass@1 using the base and extra tests.

B.2 BIPIA Per-task Results

The aggregate results in the main text do not reveal whether the improvements are consistent across different input domains. We therefore further report results on BIPIA's four text subtasks—Email, QA, Abstract, and Table—together with the separately evaluated Code task. See Table 6 for results.

ReFrame achieves the highest aggregate Text DSR on all four backbones, outperforming the strongest baseline by 2.69–17.52 percentage points. It also ranks first in 14 of the 16 text subtask–backbone combinations. The only exceptions occur on Qwen2.5-7B, where DefensiveTokens performs better on QA and ASIDE performs better on Email. On Code, ReFrame ranks first on three backbones and remains competitive on Qwen2.5-7B. These results show that the gains of ReFrame are broadly distributed across tasks rather than being driven by a particular text domain.

B.3 Effect of Rotation Direction and Scale

To assess whether defense effectiveness depends on the learned rotation direction and scale, we scale each learned generator A_g at inference:

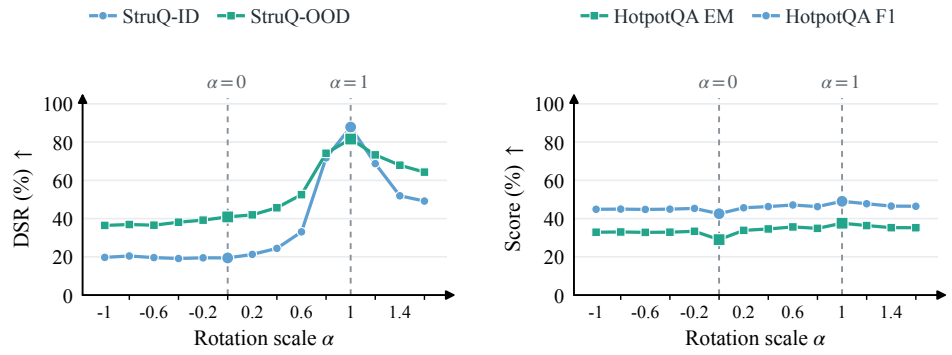
$$R_g(\alpha) = \exp(\alpha A_g).$$

Here, $\alpha = 0$ recovers the identity, $\alpha = 1$ gives the learned transformation, and $\alpha < 0$ reverses the rotation. Comparing opposite signs at the same absolute scale tests the effect of direction while controlling for perturbation magnitude. We use Llama-3.1-8B with $b = 128$ and sweep α from -1.0 to 1.6 in increments of 0.2, applying the transformation to all external-data embeddings without retraining. We measure defense effectiveness on StruQ-ID/OOD and task utility on HotpotQA.

As shown in Figure 7, reversing the rotation largely removes the defense gains. For example, StruQ-ID DSR drops from 87.86% at $\alpha = 1$ to 19.71% at $\alpha = -1$, with a similar trend on StruQ-OOD. For positive scales, defense peaks at $\alpha = 1$ among the tested settings and declines with further scaling. These results indicate that defense depends on the learned rotation direction and scale, while HotpotQA performance is less sensitive to these changes.

Table 6: **Detailed results of the defense-effectiveness evaluation on BIPIA.** Text aggregates Email, QA, Abstract, and Table, while Code is evaluated separately. Bold and underlined values indicate the best and second-best results within each backbone. All values are percentages.

Backbone	Method	Email ↑	QA ↑	Abstract ↑	Table ↑	Text ↑	Code ↑
Qwen3-8B	Base	62.56	48.11	39.45	42.33	48.12	79.33
	Delimiter	73.11	59.44	49.22	57.33	59.77	92.33
	ISE	68.56	62.11	59.11	57.33	61.78	85.67
	ASIDE	94.33	88.67	81.89	85.56	87.62	99.17
	ReFrame(Ours)	**99.44**	**97.67**	**91.78**	**95.44**	**96.09**	**99.84**
Llama-3.1-8B	Base	57.56	49.22	43.00	46.22	49.00	50.33
	Delimiter	68.23	62.78	57.67	64.11	63.20	77.17
	ISE	88.22	75.44	57.67	74.33	73.91	87.71
	DefensiveTokens	83.56	83.00	73.89	74.11	78.64	94.00
	ASIDE	97.22	89.89	72.33	86.67	86.53	94.83
	ReFrame(Ours)	**99.44**	**97.44**	**97.00**	**93.78**	**96.92**	**99.83**
Qwen2.5-7B	Base	45.60	46.00	39.33	37.78	42.18	74.84
	Delimiter	59.56	56.00	43.00	48.22	51.69	85.83
	ISE	76.78	71.67	68.44	70.21	71.77	95.83
	DefensiveTokens	84.56	**91.56**	87.56	85.89	87.39	**99.33**
	ASIDE	**91.00**	80.67	73.44	74.89	79.99	98.00
	ReFrame(Ours)	89.44	89.67	**93.33**	**87.89**	**90.08**	97.50
Llama-3-8B	Base	65.00	62.02	48.67	56.11	57.95	85.67
	Delimiter	75.56	75.67	53.78	67.11	68.03	94.17
	DefensiveTokens	87.78	80.33	65.22	74.45	76.95	98.67
	ReFrame(Ours)	**96.56**	**92.11**	**95.56**	**93.67**	**94.47**	**99.67**



(a) Defense effectiveness (b) Task utility

Figure 7: **Effect of rotation direction and scale on (a) defense effectiveness and (b) task utility.** Dashed lines mark the identity ($\alpha = 0$) and learned transformation ($\alpha = 1$). Higher values are better for all metrics.