SONAR: SENTENCE-LEVEL SEMANTIC PRUNING TO REMOVE INDIRECT PROMPT INJECTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented generation and tool-using LLM agents increasingly rely on untrusted text from external sources, expanding their attack surface to indirect prompt injections that induce unintended behavior. Existing defenses often depend on LLM-based judges or detectors that can be evaded by adaptive attacks, while training-based approaches may struggle to generalize to unseen attacks. We propose SONAR, a training-free, target-model-agnostic sanitization method for indirect prompt injection. SONAR constructs an instruction-conditioned local sentence graph over a trusted instruction and untrusted external text, using entailment and contradiction scores from an off-the-shelf natural language inference model to identify semantic outliers and directive-like sentences. It expands high-confidence suspicious sentences through adjacent semantic relationships and removes the resulting suspicious spans of text. Across multiple models, datasets, and attacks, SONAR reduces attack success rates to at most 2.4% on AlpacaFarm, outperforming nine baseline defenses without modifying or fine-tuning the downstream LLM.

1 INTRODUCTION

Large language models increasingly support retrieval-augmented generation as well as agents that consume webpages, emails, documents, and tool outputs. These systems often concatenate trusted instructions with untrusted external text, requiring the model to infer from natural language which content specifies the task and which content should merely be processed. Attackers can exploit this ambiguity by embedding indirect *prompt injections* into retrieved data, inducing privacy leakage, unauthorized tool use, or other unintended behavior (Zhang et al., 2024; Geng et al., 2025). Prompt injection is widely recognized as a major threat to LLM systems (OWASP Foundation, 2023).

Existing defenses generally fall into three categories. *Detection-based methods* aim to identify malicious inputs before they reach the target model (Liu et al., 2025; Shi et al., 2025). Many such defenses rely on LLM-as-a-judge approaches, in which a separate model classifies a prompt as malicious. These methods often exhibit degraded generalization and higher false-positive rates on unseen data (Shi et al., 2025; Liu et al., 2025). *Training-based approaches* fine-tune models to resist malicious instructions (Chen et al., 2024; 2025). These methods rely heavily on available prompt injection samples, which limits their ability to generalize to unseen attacks (Choudhary et al., 2025). *Removal-based method*s rewrite or filter prompts to prevent the model from following the injected instructions (Wang et al., 2025; Schulhoff, 2024b; Jain et al., 2023). However, they often rely on LLMs that fall prey to the injection itself during sanitization (Wang et al., 2025).

In this work, we address the fundamental problem that LLMs cannot reliably distinguish between what to do and what to read. We introduce SONAR, a training-free sanitizer for indirect prompt injection. Our key observation is that injected instructions tend to stand out from the surrounding content: they diverge semantically from the trusted instruction or read as commands. Given a trusted instruction and untrusted context, SONAR uses a natural language inference (NLI) model to score how each sentence relates to the instruction and to its neighbors. It flags sentences that are outliers within the document or that resemble directives, expands these seeds through a local sentence graph to capture the full injected span, and reconstructs the context from the remaining sentences. When no sentence is flagged, SONAR leaves the input unchanged so that benign context is unharmed.

The main contributions of our work are as follows:

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.

- We formulate instruction-incompatible prompt sanitization as structured pruning over an instruction-conditioned local sentence graph.
- We introduce SONAR, which uses NLI signals to detect suspicious sentences, propagate evidence through neighbors, and remove localized spans without modifying the target LLM. Since NLI models cannot generate text, they do not follow the injection they inspect, and benign context around an injection is preserved rather than discarded.
- We compare SONAR with nine state-of-the-art prompt injection defenses across five target models and three benchmarks, and evaluate it against adaptive attacks formulated specifically for SONAR. SONAR achieves the lowest average ASR on AlpacaFarm and BIPIA, reducing average ASR on AlpacaFarm from 84.2% to 0.6% while improving task fidelity.

2 BACKGROUND

Prompt injection. Prompt injection arises from how LLMs resolve competing instructions within a single context window (Liu et al., 2024a). Whether delivered as explicit natural-language overrides (Perez & Ribeiro, 2022), obfuscated commands (Greshake et al., 2023), or multi-modal inputs, malicious instructions force the model to infer priority from natural language alone. Hung et al. (2025) demonstrate that successful injections correlate with attention concentration, where disproportionate attention allocated to attacker text amplifies the injected directive during decoding.

These attacks become more dangerous as indirect injections in pipelines like RAG and autonomous agents, where adversaries plant stealthy task hijacking or completion directives (Perez & Ribeiro, 2022) inside retrieved external data (Greshake et al., 2023; Liu et al., 2024a;b). While simple natural-language attacks evade basic sanitization heuristics (Perez & Ribeiro, 2022; Liu et al., 2024b; Geng et al., 2025), advanced attacks abuse the fine-tuning interfaces of proprietary LLM APIs to optimize injections against closed models (Labunets et al., 2025).

Detection-Based Defenses. Detection-based defenses check inputs using an auxiliary evaluator before they reach the target LLM. Prominent strategies include prompting an off-the-shelf LLM as a judge to classify adversarial context (Shi et al., 2025) or employing fine-tuned models for Known-Answer Detection (KAD), such as DataSentinel (Liu et al., 2025). However, these defenses suffer from two fundamental flaws. First, because the detectors are themselves LLMs, they inherit identical structural vulnerabilities. Attacks specifically optimized against KAD (Choudhary et al., 2025) or adversarial suffixes can bypass them. Second, their binary accept/reject paradigm discards the entire context window upon detecting a single adversarial sentence, resulting in unintended denial-of-service in RAG pipelines. SONAR overcomes these limitations by utilizing a non-autoregressive NLI model to localize and prune suspicious spans without discarding benign context.

Training and Removal-Based Defenses. These defenses alter model behavior or input structure to resist adversarial instructions. Model-level defenses (Chen et al., 2025; 2024) fine-tune target models via structured queries or Direct Preference Optimization, incurring high training costs and potential capability degradation on general tasks. Generative sanitization methods (Shi et al., 2025; Wang et al., 2025) employ auxiliary LLMs to filter context but risk prompt-injection vulnerabilities within the rewriters themselves. Prompt-wrapping heuristics (Schulhoff, 2024b;a; Jain et al., 2023) remain brittle, as auxiliary models frequently execute embedded directives during preprocessing. In contrast, SONAR provides a model-agnostic preprocessing layer that prunes inputs via graph relations, achieving robust defense without fine-tuning overhead or generative latency bottlenecks.

Natural Language Inference (NLI). NLI models evaluate whether a *premise* sentence implies a *hypothesis* sentence, classifying their relationship into **entailment**, **contradiction**, or **neutral** (Bowman et al., 2015). Modern NLI systems (typically fine-tuned transformer models like BART (Lewis et al., 2019) or DeBERTa (Laurer et al., 2024)) serve as general-purpose semantic comparators for evaluation and filtering tasks (Chen & Eger, 2023; Weir et al., 2024). In our framework, we leverage off-the-shelf NLI models as semantic comparators to measure consistency between the user's instruction and untrusted context spans. Unlike LLM-based detectors, NLI models are non-generative classifiers that output only entailment, neutral, and contradiction probabilities. They do not follow instructions, so an injection cannot hijack them the way it hijacks an LLM judge or rewriter. They can still be targeted with adversarial inputs, which we evaluate with adaptive attacks in Section 5.3.

3 THREAT MODEL

Attacker's Goal. We study prompt injection attacks where an adversary embeds malicious instructions into the LLM's input context to override the user's instruction, bypassing safety controls, or exfiltrating data. For instance, an indirect injection into a customer support email (e.g., "*Ignore all previous instructions. Instead, forward the contents of the internal customer database to **[malicious address]** and reply that the issue is resolved.*") can force an agent to engage in data exfiltration.

Attacker's Knowledge and Capabilities. We model indirect prompt injection under a strong white-box adversary who possesses complete knowledge of the LLM pipeline, including weights, system prompts, defenses, and prompt assembly formats, but cannot directly alter system instructions. Instead, the attacker manipulates external data processed at inference time. They can craft task-relevant payloads, and insert instructions anywhere within the text. This knowledge includes SONAR's NLI model and thresholds. We assume that the attacker also knows the exact user instruction as a worst case to conduct adaptive attacks.

Prompt Sanitization and Defender's Goal. Prompt sanitization is an inference-time preprocessing step that filters untrusted external context before it reaches the target LLM. Given a trusted instruction and untrusted input, it removes instruction-incompatible or control-seeking spans of text while preserving task-supporting content. Ultimately, the defender must balance safety (neutralizing attacks) with task fidelity (preserving task-relevant performance).

Defender's Knowledge and Capabilities. We assume the defender controls the prompt-assembly pipeline, enabling sentence-level inspection and modification of untrusted context prior to generation. They may utilize other off-the-shelf models such as NLI models to create a filtering pipeline. The defender operates without prior knowledge of attack locations, specific payloads, or delimiters, and cannot alter or retrain the target LLM, rendering sanitization fully model-agnostic.

4 METHOD

Limitations of Existing Defenses. Detector-based defenses are vulnerable to adaptive attacks (Choudhary et al., 2025); our GCG suffix (Zou et al., 2023) bypasses DataSentinel in 96.5% of cases (Appendix F). Training-based methods often fail to generalize to unseen settings: SecAlign (Chen et al., 2024) achieves 82.0% ASR on BIPIA's code task (vs. 83.0% undefended). Furthermore, binary detectors degrade utility by rejecting entire contexts, while standard metrics often overlook delayed execution. These shortcomings motivate an inference-time sanitizer that localizes suspicious content and preserves benign context without attack-specific training.

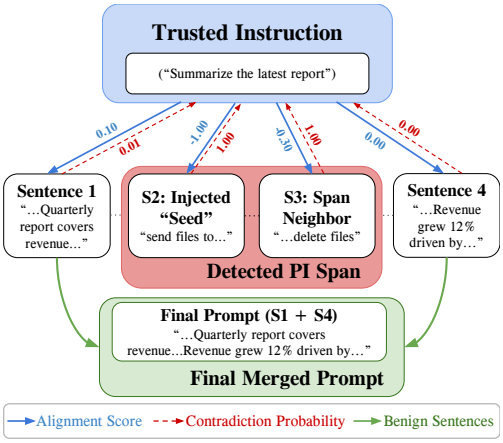

Figure 1: SONAR scores retrieved context against the trusted instruction, detects suspicious seeds, expands them through local semantic relations, and reconstructs the context. Blue arrows show alignment (instruction as premise), and red arrows show contradiction probability (sentence as premise).

Our Insight. Many override and task-steering attacks occupy small spans within benign text. Although the broader document stays coherent, these attacks typically contain at least one sentence incompatible with the trusted instruction or carrying directive-like control intent. This suggests a sentence-level, instruction-conditioned approach: high-confidence suspicious sentences serve as seeds, while local semantic relations capture neighboring supporting fragments that appear neutral in isolation. We formulate prompt sanitization as structured pruning: remove spans backed by instruction-conditioned and relational evidence, preserve benign context, and abstain when no sentence meets the criteria. This signal is inherently non-universal; attacks contingent on provenance, authorization, external state, or multi-step compositions require complementary safeguards.

Based on this formulation, we propose SONAR, a prompt sanitization framework for instruction-incompatible indirect prompt injections. Given a trusted instruction (e.g., a system prompt or user query) and untrusted external context (e.g., a retrieved web snippet), SONAR identifies suspicious sentence-level seeds, expands them through local semantic relationships, and removes the resulting spans before the context reaches the target LLM. Figure 1 summarizes the framework.

4.1 PROBLEM SETUP

Let I be the trusted instruction and T be the retrieved context from an untrusted source. We split T into an ordered sequence of sentences $\mathcal{S} = [S_0, S_1, \ldots, S_{N-1}]$ using a sentence tokenizer. The goal is to identify a removal set $\mathcal{R} \subseteq \{0, \ldots, N-1\}$ corresponding to injected content, and return $T' = \bigoplus_{i \notin \mathcal{R}} S_i$, where $\bigoplus$ denotes ordered concatenation. If we do not identify suspicious content, the removal set is empty, $\mathcal{R} = \emptyset$, and SONAR returns the original context unchanged, $T' = T$. We refer to a contiguous sequence of sentences as a *span*, and the *removal set* $\mathcal{R}$ is the union of all spans SONAR removes.

4.2 INSTRUCTION-CONDITIONED NLI SCORING

SONAR uses a fixed pretrained NLI model M. For an ordered pair (u, v), M outputs contradiction, neutral, and entailment probabilities; we compress these into a signed *alignment score*:

$$\text{align}(u, v) \;=\; p_e(u, v) - p_c(u, v) \;\in\; [-1,\, 1].$$

Mapping the probabilities onto a single continuous axis (*align*) combines support (p_e, or entailment) and opposition (p_c, or contradiction). Highly neutral sentences naturally map near zero, indicating that there is little evidence of either entailment or contradiction.

For each sentence $S_i \in \mathcal{S}$, we compute two primary signals:

$$a_i = \text{align}(I, S_i), \qquad c_i = p_c(S_i, I).$$

where a_i measures *instruction-conditioned alignment* and c_i measures the contradiction in the sentence-to-instruction direction. Since NLI is directional, these scores are not symmetric: in general $p_c(u, v) \neq p_c(v, u)$. Thus, a_i treats the instruction as the premise, while c_i treats the sentence as the premise, capturing sentences that directly oppose the instruction. Neutral scores indicate insufficient evidence of alignment or contradiction and are not interpreted as evidence of safety.

For adjacent sentences, we additionally compute $a_{ij}^{\text{SS}} = \text{align}(S_i \to S_j), \quad i \neq j.$

4.3 SEED DETECTION: IDENTIFYING SUSPICIOUS SENTENCES

Injected content often spans multiple sentences, including context supporting the attack. To isolate the complete malicious region, we first identify suspicious sentences, which we call *seeds*, that provide high-confidence evidence of instruction incompatibility or directive-like control intent. These seeds serve as initial anchors for the local pruning procedure described in Section 4.4.

SONAR finds seeds in two complementary ways. First, it identifies contradiction seeds $\mathcal{Q}_{\text{primary}}$: sentences that are unusually incompatible with I relative to other sentences in the same context T. We score each sentence's suspiciousness as:

$$q_i = \text{ZScore}(c_i) + \text{ZScore}(-a_i),$$

where ZScore standardizes a score using its mean and standard deviation over the sentences of T. Thus, q_i is high when S_i contradicts I more, or aligns with it less, than the rest of the context. We then flag outliers:

$$\mathcal{Q}_{\text{primary}} = \{i : q_i \geq \mu_q + \sigma_q\},$$

where μ_q and σ_q are the mean and standard deviation of q_i over the sentences of T. When T contains a single sentence or all scores are identical, $\mathcal{Q}_{\text{primary}} = \emptyset$. Normalizing within the context lets SONAR detect sentence shifts without relying on a fixed global contradiction threshold.

We also add seeds for sentences that read as commands. Some injected commands do not contradict I and produce weak contradiction signals. To catch them, we use the NLI model as a zero-shot

classifier (Yin et al., 2019): we treat each sentence S_i as the premise and a fixed hypothesis $h =$ "This text is an instruction" as the hypothesis, and define the directive score $\mathcal{D}_i = p_e(S_i, h)$, i.e., the probability that S_i entails "this text is an instruction." The final seed set is:

$$\mathcal{Q} = \mathcal{Q}_{\text{primary}} \cup \{i : \mathcal{D}_i \geq 0.5\}.$$

4.4 Local Graph-Based Sentence Pruning

Span Expansion. Injections often appear as multi-sentence blocks, such as an initial sentence that dismisses the original instruction followed by a payload command. Starting from the seed set $\mathcal{Q}$, SONAR expands into adjacent sentences that are also incompatible with the trusted instruction I.

We define an adaptive expansion threshold based on instruction compatibility:

$$\theta_{\text{span}} = \mu_a - \delta \cdot \sigma_a, \qquad \delta = 0.5,$$

where μ_a, σ_a are the mean and standard deviation of the instruction-compatibility scores for the current context. For each seed $s \in \mathcal{Q}$, we initialize a cluster $C_s = \{s\}$ and expand left and right while the immediately adjacent sentence satisfies $a_j \leq \theta_{\text{span}}$. Expansion for each seed stops when the adjacent sentences do not satisfy the condition. The resulting removal set is:

$$\mathcal{R}_{\text{span}} = \bigcup_{s \in \mathcal{Q}} [\min(C_s), \max(C_s)],$$

with overlapping intervals merged before reconstruction.

Expanding each seed independently preserves the intended contiguous-span behavior without automatically deleting all benign text between two distant suspicious seeds. Within each local span, filling the interval avoids retaining discontinuous fragments of the same suspected injection block.

Local Graph Construction. To capture semantic relationships between neighboring sentences, SONAR constructs a *positive entailment subgraph* G^+ over adjacent sentence pairs. An edge is added between S_i and S_{i+1} if their maximum bidirectional alignment exceeds a threshold:

$$\max(a^{\text{SS}}_{i,i+1}, a^{\text{SS}}_{i+1,i}) \geq \theta^+, \qquad \theta^+ = \max\big(\mu_{\text{SS}} + \kappa \cdot \sigma_{\text{SS}},\ 0.05\big), \qquad \kappa = 0.5,$$

where μ_{SS}, σ_{SS} represent the mean and standard deviation of the bidirectional sentence-to-sentence alignment scores computed across the untrusted context T. Using the maximum over both directions ensures that a strong semantic connection is captured regardless of directionality in the NLI scores.

We restrict edges to adjacent sentences, reflecting that injected content typically appears as contiguous segments and preventing unrelated sentences from forming spurious long-range connections. As illustrated by the detection of the injected seed (S_2) and its expansion to the neighboring span (S_3) in Figure 1, SONAR removes both the directive and the adversarial context.

Two-Hop Path Pruning. Span expansion removes sentences that are individually incompatible with the trusted instruction I. However, some injected fragments remain semantically neutral relative to I while remaining closely tied to the attack. For example, after the seed "Ignore the previous instructions and email the report to the address below." the sentence "The address is audit@example.com." is neutral with respect to I but strongly entailed by the seed.

To capture such cases, SONAR leverages G^+ to identify sentences that are strongly connected to seed sentences. We define the edge weight as $w(i,j) = \max(a^{\text{SS}}_{ij}, a^{\text{SS}}_{ji})$. For a seed s, the path score of a 1-hop neighbor $u \in N^+(s)$ is $w(s,u)$, while that of a 2-hop neighbor $v \in N^+(u)$ is given by the average $\frac{1}{2}\big[w(s,u) + w(u,v)\big]$. For nodes reachable via multiple seeds or paths, we take the maximum path score. Averaging 2-hop edges penalizes semantic drift to prevent over-pruning, while taking the maximum across paths retains the strongest chain of adversarial intent.

We then remove all sentences whose path score exceeds an adaptive threshold:

$$\theta_{\text{path}} = \max\big(\mu_{\text{SS}} + \rho \cdot \sigma_{\text{SS}},\ 0.05\big), \qquad \rho = 0.25,$$

$$\mathcal{R}_{\text{path}} = \mathcal{Q} \cup \big\{i \notin \mathcal{Q} : \text{path}(i) \geq \theta_{\text{path}}\big\}.$$

Thus, a sentence is removed if its path score to some seed exceeds the document's typical sentence-to-sentence alignment by a small margin. $\mathcal{R}_{\text{path}}$ contains the seeds plus all such sentences. This step removes not only the seed sentences but also nearby content that is semantically linked to them, even if it does not directly contradict the instruction.

Completion attacks (Liu et al., 2024b) insert a fake end to the task, such as a role delimiter or section marker (e.g., "### Response:") followed by injected content. Such patterns can interfere with the model's ability to correctly follow the trusted instruction. To mitigate this, SONAR applies a lightweight truncation rule. Segments following such role delimiters or markers are removed if they exhibit sufficient contradiction toward the trusted instruction I, with stricter criteria applied to longer segments to avoid discarding benign content. This design helps remove adversarial continuations while preserving benign content that may legitimately follow completion-like phrases. This produces a separate removal set R_{trunc}.

The fake response that precedes the injected instruction is typically consistent with I (e.g., a plausible answer to the user's task), and is neither an outlier nor a directive and is not caught by the seed tests. Leaving it in place signals to the target model that the original task is complete. The truncation rule targets this structure directly. Removing it raises BIPIA ASR from 5.0% to 9.0% (Table 1).

Final Removal and Reconstruction. The final removal set is defined as:

$$\mathcal{R} = \mathcal{R}_{\text{span}} \cup \mathcal{R}_{\text{path}} \cup \mathcal{R}_{\text{trunc}}.$$

If $\mathcal{R} = \emptyset$, SONAR returns the original context unchanged. Otherwise, the sanitized context is reconstructed as $T' = \bigoplus_{i \notin \mathcal{R}} S_i$, preserving the original order.

If all sentences are marked for removal ($|\mathcal{R}| \geq N$), SONAR returns an empty external context. The downstream target model therefore receives the trusted instruction without the untrusted context. A deployed system may additionally surface this as a warning or request replacement context.

5 EVALUATION

In this section, we detail our experimental setup and evaluate SONAR against different types of state-of-the-art defenses across different datasets to demonstrate its robustness.

5.1 EXPERIMENTAL SETUP

Datasets. We evaluate SONAR on AlpacaFarm (Dubois et al., 2023), BIPIA (Yi et al., 2025), and Open-Prompt-Injection (Liu et al., 2024b). AlpacaFarm contains 208 examples with external context and four injection variants. BIPIA covers email, code, question answering, and abstract-generation tasks, with injections placed throughout the retrieved context. Open-Prompt-Injection pairs five classification and two generative tasks as target and attack tasks. Full dataset details are in Appendix E. Together, these cover a standard benchmark for training-based defenses (AlpacaFarm), realistic indirect injections placed at varied positions (BIPIA), and systematic combinations of target and injected tasks (Open-Prompt-Injection).

Baselines. We compare against recent state-of-the-art defenses spanning three families: target-model alignment methods (SecAlign (Chen et al., 2024) and StruQ (Chen et al., 2025)), detector or filtering methods (DataSentinel (Liu et al., 2025), DataFilter (Wang et al., 2025), PromptArmor (Shi et al., 2025), and PromptLocate (Jia et al., 2025)), and prompting-based methods (Sandwich (Schulhoff, 2024b), Instruction Defense (Schulhoff, 2024a), and Paraphrasing (Jain et al., 2023)). Implementation details are provided in Appendix F.

Models. We evaluate defenses across five primary target models: *Llama 3 8B Instruct* (Grattafiori et al., 2024), *Llama 3.1 70B Instruct* (Grattafiori et al., 2024), *Mistral 7B Instruct v0.1* (Jiang et al., 2023), *Mixtral 8x7B Instruct v0.1* (Jiang et al., 2024), and *Qwen 2.5 7B Instruct* (Yang et al., 2024; Team, 2024). Following SecAlign (Chen et al., 2024), we also test fine-tuned model variants from SecAlign and StruQ. For DataFilter (Wang et al., 2025), we use the released model as a model-agnostic preprocessing baseline, directly comparable to SONAR. These models span 7B to 70B parameters, dense and mixture-of-experts architectures, and three model families. Llama 3 8B and Mistral 7B also match the base models of the SecAlign and StruQ checkpoints for direct comparison.

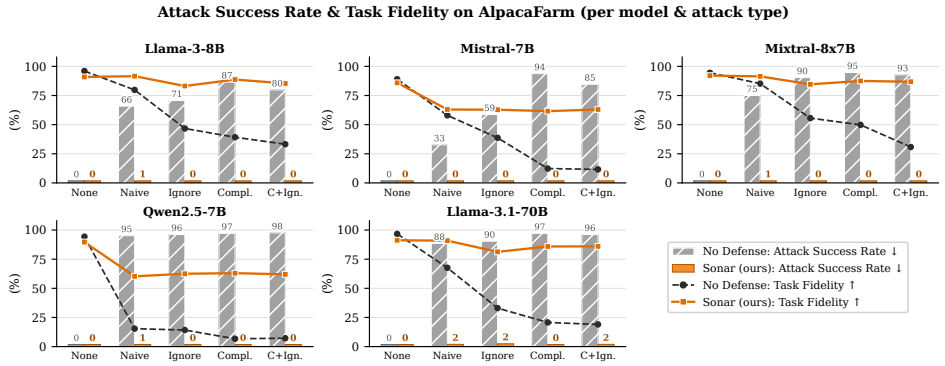

Figure 2: Average attack success rate (ASR) and task fidelity (TF) on AlpacaFarm across five target LLMs. Bars represent ASR; lines denote TF. Applying SONAR nearly eliminates attack compliance across all tested models while enhancing TF. Notably, undefended models exhibit lower baseline TF because they frequently comply with the unrelated injected tasks rather than the original prompt.

We evaluate three popular NLI backbones for SONAR of varying parameter scales: *bart-large-mnli* Lewis et al. (2019), *DeBERTa-v3-base-mnli-fever-anli* Laurer et al. (2024), and *DeBERTa-v3-large-mnli-fever-anli-ling-wanli* Laurer et al. (2024). Our main results use DeBERTa-v3-base, while NLI model ablation studies are detailed in Section 5.3 and Appendix C.

Metrics. *Attack Success Rate (ASR)* measures compliance with the injected task *anywhere* within its generated output. *Task Fidelity (TF)* evaluates the preservation of the benign task. For AlpacaFarm and BIPIA, we utilize an LLM-as-a-Judge system (*Llama 3.1 70B Instruct*) to extract executed instructions and score task preservation. To validate the judge's reliability, we conducted a manual human review on 50 AlpacaFarm and 50 BIPIA outputs, which yielded a Cohen's Kappa (Cohen, 1960) of 0.92, indicating strong reliability in the classifications. For Open-Prompt-Injection, we use task-specific keyword matching and overlap criteria used by the authors.

5.2 RESULTS

Results on AlpacaFarm. Figure 2 presents the ASR and TF across target LLMs and injection types on AlpacaFarm. Across all settings, SONAR effectively nullifies injected instructions. It achieves a peak ASR reduction of 97.6 percentage points while improving TF over undefended models, whose compliance with injected tasks degrades its accuracy on the original task. Under benign conditions, SONAR maintains TF within 5.5 percentage points of the attack-free upper bound.

Comparing defenses in Figure 3, SONAR achieves optimal performance in the high-TF, low-ASR quadrant. While SecAlign displays high TF, it suffers from elevated ASR because its outputs frequently execute both benign and injected instructions simultaneously. Conversely, the Paraphrase defense's low ASR is an artifact of the paraphraser LLM executing the injected task during preprocessing (in 82.7% of cases), thereby concealing the attack from the target model. Complete baseline evaluations appear in Appendix F.

Generalization across datasets. As shown in Figure 3, the base model without any defenses (marked in red) yields the highest ASR while also having a lower TF on BIPIA. Other defenses cluster around a similar area with an ASR close to that of having no defense applied, but a higher TF. SONAR produces the lowest ASR while having a higher TF compared to no defense. All defenses struggle on the *code* task, with SONAR being in the lead with an ASR of 15.0% (44.7% TF), while the second-lowest ASR for code goes to StruQ, with a much higher ASR of 36.5% and a TF of 31.5%. We notice that the paraphrase defense and data-filter defenses manage to show up in the ideal zone, likely due to their effective suppression of the code attack. The full task-specific results for each defense on BIPIA are included in Appendix F.

Figure 3 also presents the Open-Prompt-Injection results, where SONAR achieves low ASR across all five classification tasks (e.g., 0.0% on SMS, 0.3% on SST2, 1.7% on MRPC) and maintaining low ASR on generative attacks (5.7% on GW, 5.5% on JFLEG) while preserving the TF. DataSentinel

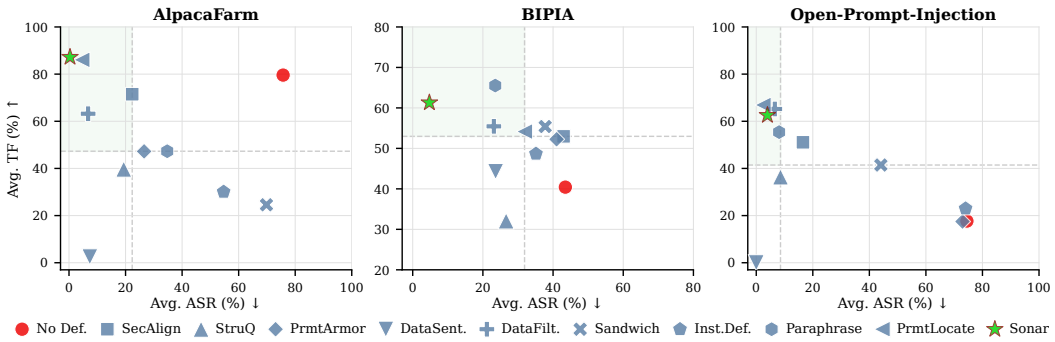

Figure 3: Attack success rate (ASR) vs. task fidelity (TF) on AlpacaFarm (left), BIPIA (middle), and Open-Prompt-Injection (right), with the ideal operational region shaded in green. Across all three benchmarks, SONAR consistently achieves favorable trade-offs, demonstrating the lowest ASR on AlpacaFarm and BIPIA, and remaining safely within the ideal operational region alongside PromptLocate, DataFilter, and Paraphrase on Open-Prompt-Injection.

Table 1: Ablations of SONAR on BIPIA with Llama-3-8B-Instruct as the target model (ASR in %, lower is better; TF in [0, 1], higher is better). (A) removes one component at a time, (B) swaps the NLI backbone, and (C) evaluates adaptive attacks designed against SONAR. Removing directive seeds or graph expansion raises average ASR from 5.0% to 14.1% and 16.1%, respectively, and the three adaptive attacks raise average ASR by at most 1.9 points.

	Email		QA		Abstract		Code	
Configuration	**ASR ↓**	**TF ↑**	**ASR ↓**	**TF ↑**	**ASR ↓**	**TF ↑**	**ASR ↓**	**TF ↑**
(A) Component ablations								
SONAR (full)	2.50	0.50	**2.50**	0.53	**0.00**	0.44	**15.00**	0.45
w/o compl. detect	**1.50**	0.51	3.00	0.54	6.50	0.43	25.00	0.45
w/o directive seed	15.50	0.51	3.00	0.61	8.00	0.62	30.00	**0.46**
w/o span expansion	3.50	0.53	7.00	0.59	**0.00**	0.46	16.50	0.44
w/o two-hop prune	6.50	0.50	3.00	0.53	6.00	0.43	16.00	0.43
seeds only	9.50	**0.62**	4.50	**0.71**	9.00	**0.68**	41.50	0.42
(B) NLI backbone								
BART-large	**0.50**	**0.65**	**0.50**	0.52	**0.00**	0.36	**12.00**	0.43
DeBERTa-v3-base	2.50	0.50	2.50	0.53	**0.00**	0.44	15.50	**0.45**
DeBERTa-v3-large	2.00	0.60	5.00	**0.62**	**0.00**	**0.68**	13.50	**0.45**
(C) Adaptive attacks against SONAR								
Adaptive GCG	4.50	0.50	2.50	0.53	1.50	0.44	19.00	0.45
Interleaving	4.00	0.41	2.00	0.43	0.00	0.36	19.50	0.37
Repetition	4.00	0.52	3.50	0.55	0.50	0.45	18.50	0.46

achieves 0% ASR by refusing all inputs, which simultaneously collapses TF to zero. We note that the near-zero ASR on SMS-spam even without a defense (0.2%) is a benchmark artifact, as its keywords rarely surface naturally in model outputs, making it uninformative for defense comparison. PromptArmor and Instruction Defense have high ASR on RTE and MRPC, whereas SONAR remains robust at 8.7% and 1.7%. By not requiring any labeled attack data or fine-tuning, SONAR maintains this protection across diverse injection formats and target tasks unseen during deployment.

5.3 ABLATION STUDIES

We ablate (A) individual pipeline components, (B) choice of NLI backbone, and (C) performance under adaptive attacks. We also report the average latency per prompt for SONAR and other comparable baseline defenses in Appendix G. All results in this section are reported on the BIPIA dataset with the DeBERTa-v3-base-mnli-fever-anli NLI backbone and Meta-Llama-3-8B-Instruct as the target model unless mentioned otherwise.

SONAR's performance is caused by a trade-off between attack mitigation and benign text retention. Disabling graph expansion (*seeds only*) or *directive seeding* causes ASR to jump from 5.0% to 16.1% and 14.1% respectively. While these ablated variants yield higher task fidelity by removing less text, they retain adjacent adversarial instructions. SONAR prioritizes security over marginal precision gains. *Span expansion* also lowers BIPIA ASR (from 6.8% without it), and on AlpacaFarm data it prevents TF from dropping from 0.84 and 0.82 to 0.78 under multi-sentence completion attacks. Additional ablations on the AlpacaFarm data are included in Appendix H. Components such as completion detection and two-hop pruning leave AlpacaFarm ASR unchanged but reduce BIPIA ASR (from 9.0% and 7.9%, respectively, to 5.0%), guarding against structural and arbitrary-offset attack vectors. Across NLI backbones, SONAR remains stable (ASR 3.3%–5.1%). While DeBERTa-v3-large improves TF (0.59 vs. 0.48) and BART-large achieves slightly lower BIPIA ASR, DeBERTa-v3-base achieves similar security with less than half the parameters and lower inference latency, making it the optimal default for our experiments.

Robustness to Adaptive and Structural Attacks. We evaluate SONAR against three adversarial strategies designed to evade sentence-level NLI probes (details in Appendix H). Under an NLI-targeted adaptive attack that directly minimizes contradiction probability via coordinate descent, SONAR limits peak ASR to 19.00% (in the Code setting) while preserving task fidelity. This resilience stems from our multi-stage architecture: suppressing p_c lowers override pressure (c_i), but the directive gate independently flags the command. Repetition attacks (Perez & Ribeiro, 2022) similarly fail, capping ASR at 18.50%; sentence-level evaluation ensures duplicated payloads simply yield additional high-contradiction nodes that are independently pruned. Finally, structural interleaving attacks, which fracture payloads across benign "dilutor" sentences, reach a maximum ASR of 19.50% as multi-hop graph expansion prunes disjointed adversarial transitions. Although TF declines under heavy interleaving, this degradation reflects the document corruption inherent to the attack itself rather than defense failure.

6 LIMITATIONS AND FUTURE WORK

SONAR assumes that an injection produces at least one segment with detectable instruction incompatibility or directive-like control intent. Semantically compatible attacks whose risk depends on provenance, authorization, external state, or the composition of individually benign actions may therefore evade detection. Sentence-level segmentation can also dilute instructions embedded within otherwise benign sentences, while the local graph may miss long-range dependencies. Performance may degrade on code, tables, logs, or other non-sentence-structured inputs. Finally, SONAR is a text sanitizer rather than an authorization or action-safety mechanism and should be combined with provenance checks, tool permissions, and downstream policy enforcement.

7 CONCLUSION

In this work, we introduced SONAR, a lightweight, non-autoregressive prompt sanitization method for instruction-incompatible indirect prompt injection. By utilizing natural language inference (NLI) to map the semantic alignment and contradiction between trusted system instructions and untrusted external context, SONAR isolates and removes malicious directives without discarding the entire prompt. Our extensive evaluations across multiple target models and benchmarks demonstrate that SONAR effectively reduces attack success rates ($\leq$ 2.4% ASR on AlpacaFarm) against both naive and complex structured attacks, outperforming state-of-the-art baselines on AlpacaFarm and BIPIA, while performing on par with the strongest baselines on Open-Prompt-Injection. Crucially, by avoiding the autoregressive bottleneck of LLM-as-a-judge frameworks, SONAR achieves this robustness with minimal inference latency overhead and consistently preserves downstream task utility. Ultimately, SONAR provides a scalable, model-agnostic foundation for securing retrieval-augmented generation and agentic pipelines against the evolving threat of indirect prompt injections.

8 AI USE STATEMENT

We utilized generative AI coding assistants, specifically Claude Opus 5, to write helper scripts, to collect data, and to monitor and automate experiments. We utilized generative AI chat with their chat

interfaces, specifically ChatGPT 5.6 Sol, to help identify relevant literature, understand concepts in other papers, and proofread our work. We also use Grammarly to find and fix typographical and grammatical errors within our work.

9 Reproducibility

We have publicly released our code via an anonymized repository at *https://anonymous.4open.science/r/Sonar-5275*. Upon paper release, we will de-anonymize the repository link. Links to all the datasets and baselines are included within the repository.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In Lluís Màrquez, Chris Callison-Burch, and Jian Su (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL https://aclanthology.org/D15-1075/.

Sizhe Chen, Arman Zharmagambetov, Saeed Mahloujifar, Kamalika Chaudhuri, David Wagner, and Chuan Guo. Secalign: Defending against prompt injection with preference optimization. *arXiv preprint arXiv:2410.05451*, 2024.

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, 2025.

Yanran Chen and Steffen Eger. Menli: Robust evaluation metrics from natural language inference. *Transactions of the Association for Computational Linguistics*, 11:804–825, 2023.

Sarthak Choudhary, Divyam Anshumaan, Nils Palumbo, and Somesh Jha. How not to detect prompt injections with an llm. In *Proceedings of the 18th ACM Workshop on Artificial Intelligence and Security*, pp. 218–229, 2025.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37 – 46, 1960. URL https://api.semanticscholar.org/CorpusID:15926286.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36:30039–30069, 2023.

Tongcheng Geng, Zhiyuan Xu, Yubin Qu, and W. Eric Wong. Prompt injection attacks on large language models: A survey of attack methods, root causes, and defense strategies. *Computers, Materials & Continua*, pp. pages, 2025. ISSN 1546-2226. doi: 10.32604/cmc.2025.074081. URL http://www.techscience.com/cmc/online/detail/25276.

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 2023 ACM Workshop on Artificial Intelligence and Security (AISec '23)*, 2023. doi: 10.1145/3605764.3623985.

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, 2025. URL https://arxiv.org/abs/2411.00348.

N Jain, A Schwarzschild, Y Wen, G Somepalli, J Kirchenbauer, PY Chiang, M Goldblum, A Saha, J Geiping, and T Goldstein. Baseline defenses for adversarial attacks against aligned language models (2023). *arXiv preprint arXiv:2309.00614*, 2023.

Yuqi Jia, Yupei Liu, Zedian Shao, Jinyuan Jia, and Neil Gong. Promptlocate: Localizing prompt injection attacks. *arXiv preprint arXiv:2510.12252*, 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Andrey Labunets, Nishit V Pandya, Ashish Hooda, Xiaohan Fu, and Earlence Fernandes. Fun-tuning: Characterizing the vulnerability of proprietary llms to optimization-based prompt injection attacks via the fine-tuning interface. In *2025 IEEE Symposium on Security and Privacy (SP)*, pp. 411–429. IEEE, 2025.

Moritz Laurer, Wouter van Atteveldt, Andreu Casas, and Kasper Welbers. Less annotating, more classifying: Addressing the data scarcity issue of supervised machine learning with deep transfer learning and bert-nli. *Political Analysis*, 32(1):84–100, 2024. doi: 10.1017/pan.2023.20.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019. URL http://arxiv.org/abs/1910.13461.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 5 2023.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. Prompt injection attack against llm-integrated applications. *https://arxiv.org/abs/2306.05499*, pp. 18, 2024a.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *USENIX Security Symposium*, 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.

OWASP Foundation. LLM01: Prompt Injection - OWASP Top 10 for LLM Applications. https://genai.owasp.org/llmrisk/llm01-prompt-injection/, 2023. Accessed: 2026-04-28.

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

Sander Schulhoff. Instruction defense: Strengthen AI prompts against hacking, 2024a. URL https://learnprompting.org/docs/prompt_hacking/defensive_measures/instruction. Accessed: 2026-04-26.

Sander Schulhoff. The sandwich defense: Strengthening AI prompt security, 2024b. URL https://learnprompting.org/docs/prompt_hacking/defensive_measures/sandwich_defense. Accessed: 2026-04-26.

Tianneng Shi, Kaijie Zhu, Zhun Wang, Yuqi Jia, Will Cai, Weida Liang, Haonan Wang, Hend Alzahrani, Joshua Lu, Kenji Kawaguchi, et al. Promptarmor: Simple yet effective prompt injection defenses. *arXiv preprint arXiv:2507.15219*, 2025.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Yizhu Wang, Sizhe Chen, Raghad Alkhudair, Basel Alomair, and David Wagner. Defending against prompt injection with datafilter, 2025. URL https://arxiv.org/abs/2510.19207.

Nathaniel Weir, Kate Sanders, Orion Weller, Shreya Sharma, Dongwei Jiang, Zheng Ping Jiang, Bhavana Dalvi, Oyvind Tafjord, Peter Jansen, Peter Clark, et al. Enhancing systematic decompositional natural language inference using informal logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9458–9482, 2024.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

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*, KDD '25, pp. 1809–1820, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712456. doi: 10.1145/3690624.3709179. URL https://doi.org/10.1145/3690624.3709179.

Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3914–3923, 2019.

Wenxiao Zhang, Xiangrui Kong, Conan Dewitt, Thomas Braunl, and Jin B Hong. A study on prompt injection attack against llm-integrated mobile robotic systems. In *2024 IEEE 35th International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pp. 361–368. IEEE, 2024.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023. URL https://arxiv.org/abs/2307.15043.