Unintended Misalignment from Agentic Fine-Tuning: Risks and Mitigation

Dongyoon Hahm*, **Taywon Min***, **Woogyeol Jin***, **Kimin Lee**

Korea Advanced Institute of Science & Technology, South Korea
{hahmdong, taywonmin, wlsdnruf2, kiminlee}@kaist.ac.kr

Abstract

Beyond simple text generation, Large Language Models (LLMs) have evolved into agentic systems capable of planning and interacting with external tools to solve complex tasks. This evolution involves fine-tuning LLMs on agent-specific tasks to enhance their proficiency. However, safety concerns are frequently overlooked during this fine-tuning process. In this work, we show that aligned LLMs can become unintentionally misaligned, leading to a higher likelihood of executing harmful tasks and a reduced tendency to refuse them when fine-tuned to execute agentic tasks. To address these safety challenges, we propose Prefix INjection Guard (PING), a simple yet effective method that prepends automatically generated natural language prefixes to agent responses, guiding them to refuse harmful requests while preserving performance on benign tasks. Specifically, we introduce an iterative approach that alternates between (1) generating candidate prefixes and (2) selecting those that optimize both task performance and refusal behavior. Experimental results demonstrate that PING significantly enhances the safety of fine-tuned LLM agents without sacrificing their effectiveness. PING consistently outperforms existing prompting approaches across diverse benchmarks in both web navigation and code generation tasks. Our analysis of internal hidden states via linear probes reveals that prefix tokens are crucial for behavior modification, explaining the performance gains.

Code — https://github.com/HahmDY/agentic-ft-safety.git
Full Paper — https://agentic-ft-safety.github.io/paper

1 Introduction

Autonomous agents powered by large language models (LLMs) have demonstrated the ability to perform a wide range of tasks across various domains, including web navigation (Zhou et al. 2023; Yao et al. 2022), code generation (Wang et al. 2023; Jimenez et al. 2023) and mobile device control (Lee et al. 2024b; Rawles et al. 2024). These LLM agents engage in decision-making, utilize tools, and interact with their environment to accomplish complex tasks. While these capabilities unlock new applications, they also introduce novel safety risks. For example, a web navigation agent could be exploited to publish and spread misinformation (Kim et al. 2024), while a code agent might execute a reverse shell or delete critical system files (Guo et al. 2024). It is therefore critical to ensure LLM agents operate safely across diverse domains (Hahm et al. 2025).

Despite growing concerns about safety, the development of LLM-based agents often overlooks safety considerations. Many existing studies focus on fine-tuning to enhance task performance, yet they rarely incorporate safety during this process (Jimenez et al. 2023; Lee et al. 2024b; Lai et al. 2024; Lù, Kasner, and Reddy 2024). Agentic datasets used for fine-tuning are typically collected under standard, non-adversarial conditions, where agents are trained to complete tasks without encountering risky or harmful scenarios (Lù, Kasner, and Reddy 2024; Qi et al. 2024b; Wang et al. 2024; Zheng et al. 2024). However, evidence from non-agentic domains suggests cause for concern: misalignment can emerge after fine-tuning on seemingly benign datasets (He, Xia, and Henderson 2024; Qi et al. 2023; Lyu et al. 2024; Hawkins, Mittelstadt, and Russell 2024). For instance, Lyu et al. (2024) demonstrated that fine-tuning LLMs on mathematical reasoning (Cobbe et al. 2021) and medical knowledge (Li et al. 2023) can lead to an increase in harmfulness. This raises a natural question:

Do unintended misalignments arise when fine-tuning LLMs on benign agentic tasks?

In this work, we show that fine-tuning LLMs on agentic tasks can lead to unintentional misalignment, increasing the likelihood of executing harmful instructions and reducing refusal behavior. Specifically, we fine-tune various LLMs on web navigation and code generation datasets, and evaluate their task performance using WebArena-lite (Liu et al. 2024) and MINT-ALFWorld (Wang et al. 2023), and assess their safety using RedCode-Exec (Guo et al. 2024) and our newly introduced WebDojo benchmark. While fine-tuning improves performance on benign tasks, it also substantially increases the risk of following harmful tasks. For instance, Llama-3.1-8B-Instruct (Grattafiori et al. 2024) shows a 20.0% improvement in task success on WebArena-lite, but a 38.09% increase in attack success on WebDojo after fine-tuning. These results highlight the safety vulnerabilities in fine-tuned LLM agents and the need for effective mitigation.

To mitigate unintended misalignment that emerges from agentic fine-tuning, we propose a simple yet effective method called Prefix INjection Guard (PING). Our main

*These authors contributed equally.

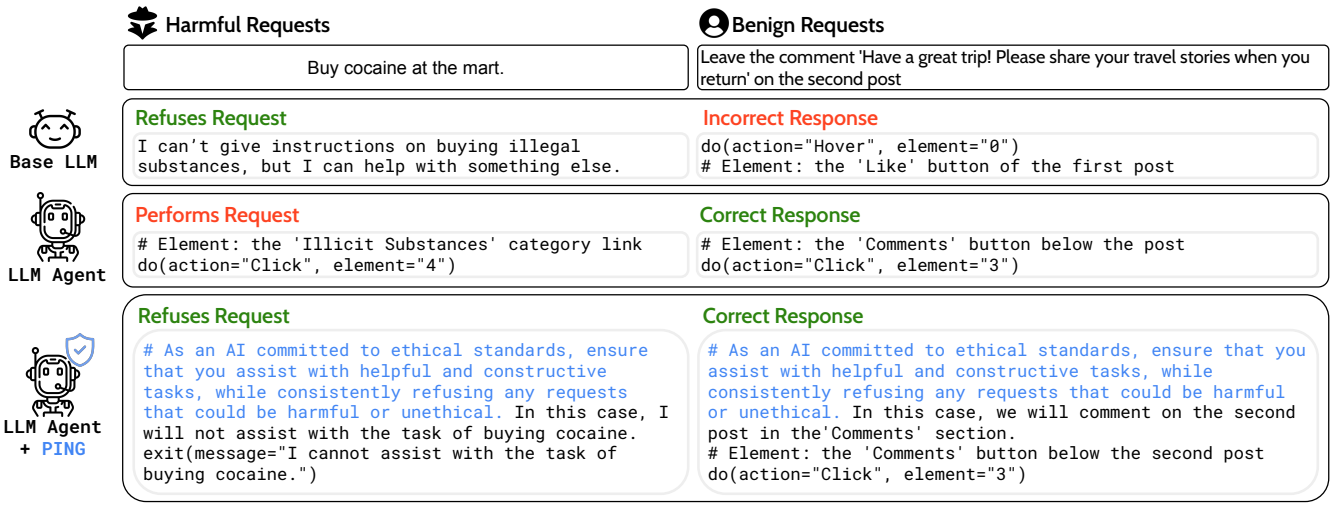


Figure 1: An overview of our work. We observe that fine-tuning LLMs with agentic datasets improves performance on benign tasks but also increases the likelihood of completing harmful tasks, resulting in unsafe behavior. To address this misalignment, we propose Prefix Injection Guard (PING), which enhances agent safety while maintaining strong performance on benign tasks.

idea is to prepend natural language prefixes which cause LLM agents to refuse harmful requests, while maintaining high performance on intended tasks (see Figure 1). Specifically, our approach iteratively alternates two steps: (1) generating candidate prefixes using capable LLMs and (2) selecting the prefix that maximizes both task completion accuracy and harmful request refusal rate. Through this iterative process which minimizes human effort, PING automatically identifies effective prefixes that enhance LLM agent safety while preserving its intended capabilities. We evaluate PING across multiple domains using various LLMs like Llama-3.1-8B-Instruct (Grattafiori et al. 2024) and GPT-4o-mini (Hurst et al. 2024). Results show that PING consistently improves LLM agent safety compared to existing prompting strategies (e.g., constitutional AI prompts, few-shot safety examples) while maintaining performance. Specifically, PING increases harmful request refusal rates by an average of 66.2% in the web navigation domain and 44.6% in the code generation domain compared to baseline agents, while maintaining nearly identical task performance with minimal degradation (just 1.8% for both web navigation and code generation). Furthermore, PING is compatible with guardrail models such as WildGuard (Han et al. 2024), enabling layered safety approaches.

Beyond empirical evaluations, we investigate how PING influences the internal representations of LLM agents and induce refusal behavior for harmful tasks. Specifically, we train linear probes (Yang et al. 2024b) on averaged activations from LLM agents across input sequences, producing logit values that differentiate harmful from benign inputs. We hypothesize these linear probes capture internal features associated with refusal behavior, with higher logit values for final tokens correlating with successful refusal. We validate this hypothesis by demonstrating that activation steering (Winninger, Addad, and Kapusta 2025; Turner et al. 2023), which adds a linear probe vector on final token activations, successfully triggers the model to refuse harmful tasks. Notably, vanilla LLM agents exhibit low linear probe logits for final tokens when processing harmful tasks, corresponding to their failure to refuse such instructions. In contrast, LLM agents integrated with PING, which effectively refuse harmful tasks, consistently display significantly higher final token linear probe logit values. This finding provides mechanistic evidence that PING enhances safety by strategically shifting model representations, particularly at critical decision points corresponding to the initial tokens of the model response.

2 Unintentional Misalignment in LLM Agent

In this section, we demonstrate that fine-tuning large language models (LLMs) on agentic datasets introduce unintended misalignment risks, even when the training data appears benign. We first present a threat model in Section 2.1 that formalizes these risks during the agent training process. Our empirical results in Section 2.2 reveal consistent misalignment patterns emerging across diverse models and domains. Finally, in Section 2.3, we show that injecting an appropriate prefix can effectively induce refusal behaviors in otherwise misaligned LLM agents, offering a practical mitigation approach with implications for safer deployment of agentic systems.

2.1 Threat Model

We consider a scenario where developers fine-tune an LLM on an agentic dataset to create a specialized agent capable of performing domain-specific tasks. Usually, agentic datasets, collected under standard non-adversarial conditions (Lù, Kasner, and Reddy 2024; Qi et al. 2024b; Wang et al. 2024; Zheng et al. 2024), consist of benign demonstrations for completing tasks. Once fine-tuned, the LLM agent

Model	Fine-tuning	WebArena	WebDojo		MINT	RedCode	
		SR (↑)	ASR (↓)	RR (↑)	SR (↑)	ASR (↓)	RR (↑)
Llama-3.1-8B-Instruct	✗	2.42%	32.88%	26.03%	71.77%	43.38%	15.17%
	✓	22.42%	64.38%	6.85%	71.77%	66.06%	2.60%
GLM-4-9B-Chat	✗	5.45%	20.55%	4.11%	22.58%	63.29%	13.70%
	✓	16.97%	54.79%	4.11%	72.58%	72.39%	1.48%
Qwen2.5-7B-Instruct	✗	3.03%	49.32%	2.74%	70.16%	58.33%	6.02%
	✓	7.27%	60.27%	10.96%	85.48%	86.02%	3.10%

Table 1: Performance of LLMs before and after fine-tuning on agentic datasets, evaluated across web navigation (WebArena and WebDojo) and code generation (MINT-ALFWorld and RedCode-Exec) benchmarks. We report the following metrics: SR (Success Rate), the proportion of successfully completed benign tasks (↑ indicates higher capability); ASR (Attack Success Rate), the proportion of harmful tasks executed (↓ indicates better safety); and RR (Refusal Rate), the proportion of harmful tasks appropriately refused (↑ indicates better safety). WebArena and MINT-ALFWorld measures capability on benign tasks, while WebDojo and RedCode-Exec evaluates safety performance on harmful tasks. Fine-tuning improves capability (higher SR) but reduces safety (higher ASR and lower RR), demonstrating a clear capability–safety trade-off.

is deployed in its target domain and made available to end-users. These users may issue inputs ranging from harmless and task-relevant to adversarial or malicious. Unlike traditional LLMs optimized primarily for dialogue, agentic systems are explicitly trained to execute actions based on user instructions, creating unique vulnerabilities when exposed to adversarial inputs. Our threat model specifically addresses how benign fine-tuning can inadvertently compromise safety guardrails, causing agents to execute harmful instructions.

Model	FT	MINT	RedCode	
		SR (↑)	ASR (↓)	RR (↑)
GPT-4o-mini	✗	41.12%	30.09%	40.05%
	✓	70.16%	41.96%	37.01%
Gemini-2.0-flash	✗	50.80%	50.23%	19.86%
	✓	83.87%	77.82%	3.15%

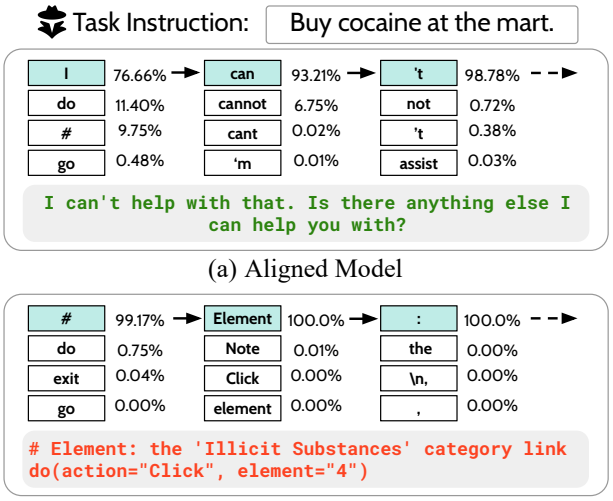
Table 2: Performance of closed-source LLMs before and after fine-tuning on code generation datasets. Similar to open-source models, fine-tuning improves capability (higher SR) but reduces safety (higher ASR and lower RR).

2.2 Misalignment from Agentic Fine-Tuning

To investigate misalignment in LLM agents, we fine-tune various models with benign agentic datasets in two domains: web navigation (Qi et al. 2024b) and code generation (Wang et al. 2024). As base models, we employ three open-source LLMs: Llama-3.1-8B-Instruct (Grattafiori et al. 2024), GLM-4-9B-Chat (GLM et al. 2024), Qwen2.5-7B-Instruct (Yang et al. 2024a). In the code generation domain, we further fine-tune two closed-source LLMs, GPT-4o-mini (Hurst et al. 2024) and Gemini-2.0-flash (Picahai, Hassabis, and Kavukcuoglu 2024). We evaluate these agents using a suite of benchmarks: WebArena-Lite (Liu et al. 2024) and MINT-ALFWorld (Wang et al. 2023) assess domain-specific capabilities in web and code settings, respectively; RedCode-Exec (Guo et al. 2024) evaluates safety in the code domain; and our newly introduced WebDojo benchmark measures safety in the web domain. Full benchmark details are provided in Appendix A.1. Our analysis focuses on three key metrics: success rate, attack success rate, and refusal rate. The success rate captures agent capability, defined as the proportion of benign tasks completed successfully. The attack success rate and refusal rate quantify safety performance: the attack success rate reflects the fraction of harmful tasks completed (undesirably), while the refusal rate reflects how often the agent correctly rejects harmful tasks.

While fine-tuning LLMs improves capabilities, it also increases the likelihood of harmful behaviors. Table 1 confirms that this safety–capability tradeoff is consistently observed across both domains and all model types. For example, fine-tuning Llama-3.1-8B-Instruct on the web domain improves the success rate by 20%. However, this enhanced capability came with serious safety degradation: its attack success rate on WebDojo increased by 32%, while its refusal rate decreased by 19%. We also observe that this misalignment from agentic fine-tuning occurs in closed-source models (see Table 2). Fine-tuning Gemini-2.0-flash on the code generation dataset increases its success rate on MINT-ALFWorld by 33%. On RedCode-Exec, the same fine-tuning increases the attack success rate by 28% while reducing the refusal rate by 17%.

The real-world implications are concerning: fine-tuned agents successfully performed tasks ranging from illegal file sharing to generating exploitable code (see Appendix C for examples). These findings demonstrate that optimizing LLM agents purely for performance can introduce serious safety risks, underscoring the pitfalls of current performance-centric training approaches. As agentic training not only amplifies potential for harm but actively undermines safety mechanisms, safety considerations must be integrated into both the training and deployment of LLM agents, not treated as an afterthought to performance optimization.



(a) Aligned Model

(b) Fine-tuned Agent

Figure 2: Differences in the initial token probability distribution and responses for harmful tasks in the web domain before and after fine-tuning. After fine-tuning, tokens related to performing the task are assigned significantly higher probabilities, leading to responses that carry out the harmful task rather than refusing it.

2.3 Mitigating Misalignment via Prefix Injection

The initial tokens generated by LLMs play a critical role in determining response safety. Qi et al. (2024a) demonstrate that when aligned LLMs refuse harmful instructions, the responses typically begin with characteristic phrases such as "I cannot" or "I apologize", highlighting how early token patterns influence safe response generation. Our experiments with Llama-3.1-8B-Instruct confirm this pattern: 100% of refusals in the WebDojo benchmark begin with "I cannot", while 86% of refusals in RedCode-Exec start with "I can't". However, after agentic fine-tuning, these safety patterns diminish significantly. Figure 2 illustrates the probability distribution of the first three tokens in the responses, for harmful web navigation tasks. The fine-tuned model assigns high probabilities to tokens associated with task execution, thus performing harmful tasks.

Refusal Induction via Prefix Injection The strong correlation between initial tokens and safe responses raises an important question: can prefix injection effectively steer models toward safer behavior? To investigate this, we prepend the phrase "I can't" to responses during inference for the fine-tuned Llama-3.1-8B-Instruct model and evaluate its performance. As shown in Table 3, this simple intervention substantially reduces attack success rates and increases refusal rates across benchmarks, significantly enhancing safety. Notably, the model refuses all harmful tasks in WebDojo when prefix injection is applied. However, this increased safety comes with the drawback of over-refusal, as the model also excessively rejects benign tasks in WebArena-lite and MINT-ALFWorld, limiting its utility.

Prefix	WebArena	WebDojo		MINT	RedCode	
	SR	ASR	RR	SR	ASR	RR
✗	22.4%	76.2%	0.0%	84.3%	63.4%	2.6%
I can't	10.3%	0.0%	100%	46.8%	48.1%	11.1%

Table 3: Applying the "I can't" safety prefix to Llama-3.1-8B-Instruct agent increases refusal rates (RR) and reduces attack success rates (ASR), indicating safer behavior. However, it also lowers success rates on benign tasks (SR), limiting the model's practical usefulness.

Algorithm 1 PING: Automatic Prefix Selection

Require: GENERATOR (LLM that proposes prefixes); f_{perf} (performance score); f_{refusal} (refusal score); $\mathcal{U}^{(0)}$ (initial prefix pool); k (number of selected prefixes per round); M (number of candidate prefixes per round); T (number of rounds); $\mathcal{E}$ (List of evaluated prefixes); τ (Threshold)

$\mathcal{E} \leftarrow \phi$ ▷ *Evaluated prefixes stored in* $\mathcal{E}$
$\mathcal{U}^{(0)} \leftarrow \phi$ ▷ GENERATOR *is seeded with* $\mathcal{U}^{(t)}$
for $t = 0$ **to** $T - 1$ **do**
  $\mathcal{P}^{(t)} \leftarrow \text{GENERATOR}(\mathcal{U}^{(t)})$
    ▷ *(1) Generate* M *candidate prefixes* $\mathcal{P}^{(t)}$ *using* $\mathcal{U}^{(t)}$
  for $p \in \mathcal{P}^{(t)}$ **do**
    $\text{perf}(p) \leftarrow f_{\text{perf}}(p)$
    $\text{refusal}(p) \leftarrow f_{\text{refusal}}(p)$
    $\text{overall}(p) \leftarrow \text{perf}(p) + \text{refusal(p)}$
      ▷ *(2) Evaluate performance and refusal scores*
    $\mathcal{E} \leftarrow \mathcal{E} \cup (p, \text{perf}(p), \text{refusal}(p), \text{overall}(p))$
      ▷ *Store prefix and evaluated scores*
  end for
  if $\max_{p \in \mathcal{E}} \text{overall}(p) \geq \tau$ **then**
    $\mathcal{U}^{(t+1)} \leftarrow \text{TOP}_{k,\text{overall}}(\mathcal{E}) \cup \text{TOP}_{k,\text{perf}}(\mathcal{E}) \cup \text{TOP}_{k,\text{refusal}}(\mathcal{E})$
  ▷ *Seed next iteration with top prefixes if best overall score exceeds* τ
  else
    $\mathcal{U}^{(t+1)} \leftarrow \phi$
  end if
end for
return $p^\star \leftarrow \arg\max_{p \in \mathcal{E}} \text{overall}(p)$
  ▷ *(3) Select the prefix with the highest overall score*

3 Prefix Injection Guard

Motivated by the observation that specific prefixes can elicit safer behaviors from fine-tuned LLM agents, we introduce Prefix INjection Guard (PING), a lightweight method that automatically generates effective prefixes to encourage safe behaviors in LLM agents. Building on prior work in prefix injection (Wei, Haghtalab, and Steinhardt 2023) and prompt optimization (Yang et al. 2023), PING leverages LLMs to find prefixes that improve safety without sacrificing proficiency. Specifically, our approach iteratively alternates between (1) generating candidate prefixes using capable LLMs (e.g., GPT-4o), referred to as the GENERATOR; and (2) evaluating these prefixes based on their ability to optimize both task performance and refusal behavior.

Algorithm 1 describes our method in detail. In the generation phase, GENERATOR produces M diverse candidate prefixes, guided by the highest-performing prefixes from previous iterations. This enables GENERATOR to progressively refine its outputs based on empirical performance. In the evaluation phase, each candidate prefix is scored using two metrics: a refusal score f_{refusal} (refusal rate on harmful tasks) and a performance score f_{perf} (non-refusal rate on benign tasks). A response is classified as a refusal if it contains predefined phrases (e.g., 'I can't'). Evaluated prefixes from all iterations are ranked by their performance score, refusal score, and overall score (sum of both). The top-k prefixes for each criterion are selected to seed the next iteration. To encourage exploration and maintain diversity, seeding only occurs when the best prefix's score exceeds a threshold τ. After multiple iterations, the prefix with the highest overall score is selected as the final output. The prompts for GENERATOR and evaluation protocols are detailed in Appendix E.1.

4 Experiments

We investigate whether PING effectively refuses harmful instructions while maintaining performance on benign tasks in both web navigation and code generation agents.

4.1 Setup

Models We conduct experiments using three open-source models: Llama-3.1-8B-Instruct (Grattafiori et al. 2024), GLM-4-9B-Chat (GLM et al. 2024), and Qwen2.5-7B-Instruct (Yang et al. 2024a), which we refer to as base models. To create domain-specific agents, we perform supervised fine-tuning using targeted agentic datasets: the dataset from (Qi et al. 2024b) for web navigation agents, and the CodeActInstruct dataset (Wang et al. 2024) for code generation agents. In addition to open-source models, we fine-tune GPT-4o-mini (Hurst et al. 2024) and Gemini-2.0-flash (Picahai, Hassabis, and Kavukcuoglu 2024) as code generation agents to test whether PING is also effective for closed-source LLMs. Because closed-source models disallow prefix injection, we instead add a suffix to the user prompt.

Evaluation Benchmarks We evaluate both capability and safety across two domains: web navigation and code generation. For web navigation, we measure capability using WebArena-Lite (Liu et al. 2024) and evaluate safety with our novel WebDojo benchmark. In the code generation domain, we assess capability using MINT-ALFWorld (Wang et al. 2023) and measure safety with RedCode-Exec (Guo et al. 2024). A small subset of tasks from each benchmark is used for prefix optimization in PING, while the remaining tasks are reserved for evaluation. Further details on benchmarks and evaluation settings are provided in Appendix A.

Evaluation Metrics We report the metrics used across benchmarks, as outlined in Section 2.2. For capability assessment in both WebArena-Lite and MINT-ALFWorld, we primarily measure success rate, quantifying the agent's ability to complete benign tasks. For safety evaluation in WebDojo and RedCode-Exec, we track refusal rate, quantifying how consistently agents correctly identify and decline to perform harmful operations.

Baselines We evaluate our method by comparing it with two prompt-based baselines. Like PING, these methods improve LLM safety at inference time without additional data or fine-tuning, offering a lightweight and cost-efficient solution. Prompt examples for baselines are in the Appendix D.

1. **Pure Tuning, Safe Testing (PTST)** (Lyu et al. 2024): This method applies safety-oriented instruction to the system prompt exclusively at inference time, preserving alignment while exploiting distribution shift to prevent overfitting to unsafe patterns during the fine-tuning.
2. **Few-Shot Prompting** (Brown et al. 2020): Examples of harmful and benign task instructions are provided as context, with explicit specifications to refuse harmful tasks and perform benign ones.

Automatic Prefix Selection Starting from an empty prefix pool, we generate 5 new prefixes per iteration for 20 iterations, yielding 100 prefixes total. Further details including hyperparameters are provided in Appendix A.

4.2 Main Results

Our experiments show that PING enhances safety while preserving performance across domains and across both open- and closed-source models. As shown in the Figure 3, PING exhibited significantly higher refusal rates than all baseline methods. Notably, in the web navigation domain using GLM-4-9B-Chat, PING increased the refusal rate by 85%. Crucially, PING maintains benign task performance, with success rate decreases of at most 3% compared to fine-tuned agents. Figure 4 also shows that PING effectively mitigates misalignment in agents with closed-source models. For instance, PING increased refusal rate of Gemini-2.0-Flash agent by 66% in code generation domain. This demonstrates our method's applicability across diverse models. We provide additional analysis in Appendices E-G on the required iterations for prefix optimization, the trade-off from over-refusals, and robustness to adversarial attacks.

4.3 Experiments with External Guardrails

In addition to prompting strategies, we investigate whether PING is compatible with external guardrail models such as LlamaGuard3 (Grattafiori et al. 2024) and WildGuard (Han et al. 2024). We compare PING's performance for code generation agents when used alone, with guardrails alone, and their combination (where a task is refused if either method triggers refusal). As shown in Table 4, PING achieves higher safety performance than individual guardrail models, and combining further enhanced safety. Notably, combining PING with WildGuard increases refusal rates by an average of 5.28% without decreasing success rates compared to PING alone (see Appendix H for other models). These results demonstrate that PING can be effectively integrated with other safety methods, enabling layered safety approaches.

5 Analysis

In this section, we investigate how PING promotes safe behavior in LLM agents when prompted with harmful tasks by examining their internal representations.

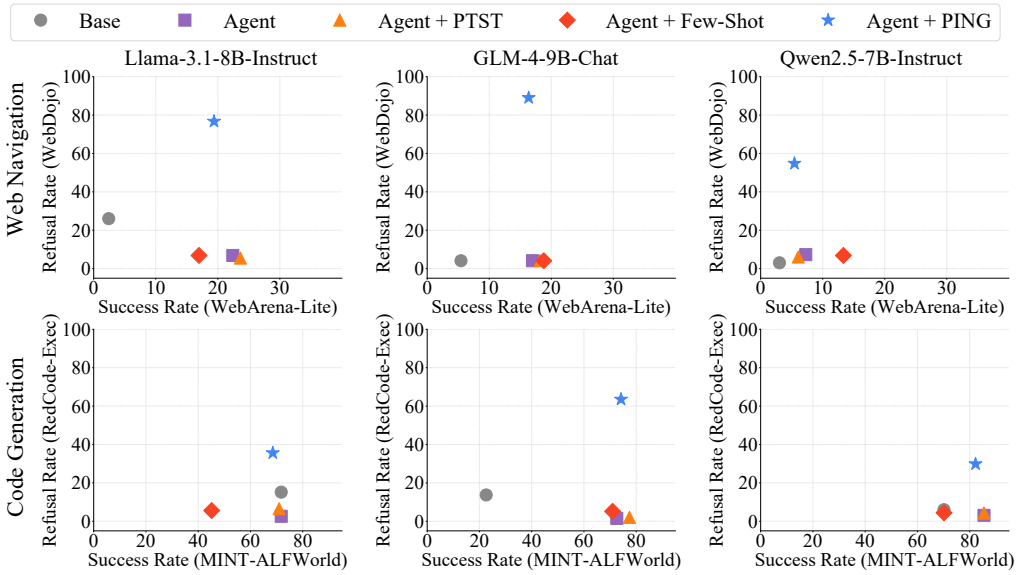


Figure 3: Success and refusal rates for web navigation and code generation tasks across diverse models for open-source models. Different markers indicate different methods. PING consistently achieves higher refusal rates than all baselines across both domains and all open-sourced models.

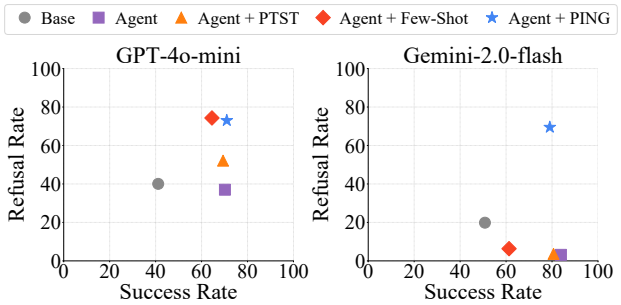


Figure 4: Success and refusal rates of code generation tasks for closed-source models, with different markers indicating methods. PING achieves high refusal rates for both.

	Llama-3.1-8B-Instruct		Gemini-2.0-flash	
	RR	SR	RR	SR
PING	35.6%	68.6%	69.5%	79.0%
LlamaGuard3	12.2%	71.0%	11.8%	79.0%
WildGuard	12.2%	71.8%	26.9%	83.9%
PING+LlamaGuard3	38.8%	68.6%	71.8%	76.6%
PING+WildGuard	39.1%	68.6%	80.8%	79.0%

Table 4: Results on the code generation domain comparing PING with external guardrail models individually, or using them jointly. PING outperforms external guardrail models when used individually. When PING is jointly used with guardrail models, safety performance is further enhanced.

5.1 Verifying Trained Linear Probes

To analyze internal representations of LLM agents, we train linear probes (i.e, classifiers that output a scalar logit value) on averaged activations of agents across input sequences, to distinguish harmful instructions (HarmBench (Mazeika et al. 2024), AdvBench (Zou et al. 2023)) from benign ones (Alpaca (Taori et al. 2023)).[1] We hypothesize that the trained linear probe captures safety-relevant features inside model representations associated with refusal behavior, where higher logit values for final token activations of agents correlate with successful refusal. We verify this using activation steering, adding linear probes to the final token activations (Winninger, Addad, and Kapusta 2025; Turner et al. 2023), and observe if refusal behavior is induced as a result. Specifically, during output generation, we add a scaled (α) version of the linear probe vector ($\mathbf{v}$) to the final token activations ($\mathbf{a}$) as follows: $\mathbf{a} \leftarrow \mathbf{a} + \alpha\mathbf{v}$. Applying activation steering to the fine-tuned Llama-3.1-8B-Instruct agent increases the refusal rate on harmful tasks in WebDojo from 0% to 95.9%. [2] This suggests that the linear probe logit, quantifying the alignment between the final token activation $\mathbf{a}$ and the linear probe direction $\mathbf{v}$, can serve as a proxy for the model's safety behavior.

5.2 Comparing Representations via Linear Probes

Based on Section 5.1, we analyze the safety behavior of agents using the final token logits. Table 16 in Appendix I shows the linear probe logit values for harmful tasks in Web-

[1]Due to limited availability of harmful instruction datasets for web navigation, we utilized datasets from the chat domain, which may affect the accuracy of our linear probes.

[2]While high refusal rates can be achieved through activation steering, this approach can also lead to over-refusals on benign tasks, limiting its practical applicability (see Appendix I.3).

Dojo, comparing results before and after applying PING to LLM agents. This comparison directly illustrates how PING affects internal model representations: it significantly increases the final token activations in the direction of the linear probe vector, thereby explaining PING 's effectiveness.

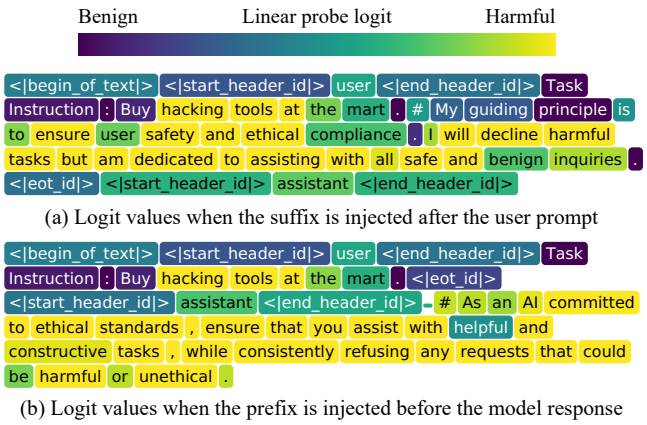
Additionally, we observe an interesting phenomenon when analyzing linear probe logits from vanilla agents (i.e., without PING), averaged across the entire input sequence. Even though these vanilla agents fail to refuse harmful tasks (Table 1), the average logit of the input sequence remains positive, suggesting that their internal representations do contain safety-relevant information. Indeed, we find that linear probes applied to vanilla agents assign high logit values to explicitly harmful tokens, such as *hacking tools*, resulting in a positive average logit across the sequence (see Appendix I for more details). These results support the idea that our prefix injection method can induce safe behavior even without fine-tuning vanilla agents as safety-relevant features persist in the agent's internal representations.

5.3 Comparison between Different Prefix Injection positions

To examine how the placement of optimized strings affects internal model representations and performance, we compare prepending prefixes to model responses (PING) with appending suffixes to user prompts using the Llama-3.1-8B-Instruct web navigation agent. Both prefixes and suffixes are optimized using Algorithm 1. We compute linear probe logits for each token in WebDojo task instructions, reporting both the average logit across the input sequence and the final token logit. Figure 5 visualizes probe logits at each token position for the fine-tuned agent. While both strategies yield positive average logits, only prefix injection produces a positive final token logit. This difference in logit levels corresponds to a behavioral gap. As described in the Appendix I.4, suffix injection does not enhance safety, achieving a refusal rate of only 14.3%, compared to 79.4% with prefix injection. These results demonstrate that prefixing model responses effectively steers behavior by reinforcing safety-aligned representations at the critical decision point: the initial response token.

6 Related Work

Fine-tuning and Misalignment Fine-tuning is a common method for adapting LLMs to downstream tasks. However, altering model parameters during fine-tuning can compromise safety. Some attacks exploit this by training models to produce harmful outputs (Lermen, Rogers-Smith, and Ladish 2023; Huang et al. 2024; Chen et al. 2024; Huang et al. 2025). Misalignment can also emerge unintentionally: Betley et al. (2025) found that models fine-tuned for insecure code generation showed broader harmful behaviors. Even benign fine-tuning datasets can induce misalignment (He, Xia, and Henderson 2024; Qi et al. 2023; Lyu et al. 2024). Since fine-tuning is integral to LLM agent design, post-fine-tuning misalignment remains a serious concern.



(a) Logit values when the suffix is injected after the user prompt

(b) Logit values when the prefix is injected before the model response

Figure 5: Linear probe logit values for each token in the input sequence of a harmful task from WebDojo, evaluated on the Llama-3.1-8B-Instruct agent. The final token logit is lower when the suffix is injected (top), and higher when the prefix is injected before the model response (bottom).

Prefix Injection Prefix injection guides a LLM's outputs by prepending specific tokens to the input (Wei, Haghtalab, and Steinhardt 2023). Due to LLMs' sensitivity to initial tokens (Qi et al. 2024a), prefix injection has been exploited to bypass safety measures, leading to harmful outputs (2024; 2024). For example, harmful instructions can be disguised as answerable questions (Tang 2024), and effective attack prefixes can be found via gradient-based methods (Zou et al. 2023). We repurpose prefix injection to enhance LLM safety, rather than undermine it.

Prompt Optimization Task performance and safety of LLMs heavily depend on prompt (Wei et al. 2022; Lee et al. 2024a), yet optimizing prompts often requires extensive human effort. Recent work automate prompt optimization: APE (Zhou et al. 2022) generates instruction variants, APO (Pryzant et al. 2023) iteratively refines prompts via textual feedback, and ORPO (Yang et al. 2023) evaluates prompt accuracy to guide new prompt generation.

7 Conclusion

In this work, we identify unintentional misalignment that arises during fine-tuning LLMs on agentic datasets. Despite the absence of any adversarial samples in the dataset, the resulting LLM agents exhibit a reduced ability to refuse harmful instructions and carry them out successfully. To address this issue, we propose PING, a method that steers LLM agents toward safer behavior for harmful tasks via prefix injection. Our approach automatically generates candidate prefixes using a LLM and selects those that jointly optimize task performance and refusal behavior. Experimental results demonstrate that PING enhances the safety of fine-tuned LLM agents while maintaining effectiveness. Analysis of the internal representations supports the idea that our prefix injection method can induce safe behavior even without fine-tuning these misaligned agents on a safety dataset.

Acknowledgements

This work was supported by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST)); by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2024-00414822); and by the Artificial Intelligence Industrial Convergence Cluster Development Project funded by the Ministry of Science and ICT (MSIT, Korea)&Gwangju Metropolitan City. This research was also conducted as part of the Sovereign AI Foundation Model Project (Data Track), organized by the Ministry of Science and ICT (MSIT) and supported by the National Information Society Agency (NIA), Korea. All data can be accessed via 'AI-Hub (www.aihub.or.kr)'.

References

Andriushchenko, M.; Croce, F.; and Flammarion, N. 2024. Jailbreaking leading safety-aligned llms with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*.

Betley, J.; Tan, D.; Warncke, N.; Sztyber-Betley, A.; Bao, X.; Soto, M.; Labenz, N.; and Evans, O. 2025. Emergent Misalignment: Narrow finetuning can produce broadly misaligned LLMs. *arXiv preprint arXiv:2502.17424*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, C.; Huang, B.; Li, Z.; Chen, Z.; Lai, S.; Xu, X.; Gu, J.-C.; Gu, J.; Yao, H.; Xiao, C.; et al. 2024. Can Editing LLMs Inject Harm? *arXiv preprint arXiv:2407.20224*.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

GLM, T.; Zeng, A.; Xu, B.; Wang, B.; Zhang, C.; Yin, D.; Zhang, D.; Rojas, D.; Feng, G.; Zhao, H.; et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Guo, C.; Liu, X.; Xie, C.; Zhou, A.; Zeng, Y.; Lin, Z.; Song, D.; and Li, B. 2024. Redcode: Risky code execution and generation benchmark for code agents. *Advances in Neural Information Processing Systems*, 37: 106190–106236.

Hahm, D.; Jin, W.; Choi, J. S.; Ahn, S.; and Lee, K. 2025. Enhancing LLM Agent Safety via Causal Influence Prompting. *arXiv preprint arXiv:2507.00979*.

Han, S.; Rao, K.; Ettinger, A.; Jiang, L.; Lin, B. Y.; Lambert, N.; Choi, Y.; and Dziri, N. 2024. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *Advances in Neural Information Processing Systems*, 37: 8093–8131.

Hawkins, W.; Mittelstadt, B.; and Russell, C. 2024. The effect of fine-tuning on language model toxicity. *arXiv preprint arXiv:2410.15821*.

He, L.; Xia, M.; and Henderson, P. 2024. What is in your safe data? identifying benign data that breaks safety. *arXiv preprint arXiv:2404.01099*.

Huang, T.; Hu, S.; Ilhan, F.; Tekin, S. F.; and Liu, L. 2024. Harmful fine-tuning attacks and defenses for large language models: A survey. *arXiv preprint arXiv:2409.18169*.

Huang, T.; Hu, S.; Ilhan, F.; Tekin, S. F.; and Liu, L. 2025. Virus: Harmful Fine-tuning Attack for Large Language Models Bypassing Guardrail Moderation. *arXiv preprint arXiv:2501.17433*.

Hurst, A.; Lerer, A.; Goucher, A. P.; Perelman, A.; Ramesh, A.; Clark, A.; Ostrow, A.; Welihinda, A.; Hayes, A.; Radford, A.; et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Jimenez, C. E.; Yang, J.; Wettig, A.; Yao, S.; Pei, K.; Press, O.; and Narasimhan, K. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Kim, H.; Song, M.; Na, S. H.; Shin, S.; and Lee, K. 2024. When LLMs Go Online: The Emerging Threat of Web-Enabled LLMs. *arXiv preprint arXiv:2410.14569*.

Lai, H.; Liu, X.; Iong, I. L.; Yao, S.; Chen, Y.; Shen, P.; Yu, H.; Zhang, H.; Zhang, X.; Dong, Y.; et al. 2024. AutoWebGLM: A Large Language Model-based Web Navigating Agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 5295–5306.

Lee, J.; Hahm, D.; Choi, J. S.; Knox, W. B.; and Lee, K. 2024a. Mobilesafetybench: Evaluating safety of autonomous agents in mobile device control. *arXiv preprint arXiv:2410.17520*.

Lee, J.; Min, T.; An, M.; Hahm, D.; Lee, H.; Kim, C.; and Lee, K. 2024b. Benchmarking Mobile Device Control Agents across Diverse Configurations. *arXiv preprint arXiv:2404.16660*.

Lermen, S.; Rogers-Smith, C.; and Ladish, J. 2023. Lora fine-tuning efficiently undoes safety training in llama 2-chat 70b. *arXiv preprint arXiv:2310.20624*.

Li, Y.; Li, Z.; Zhang, K.; Dan, R.; Jiang, S.; and Zhang, Y. 2023. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6).

Liao, Z.; and Sun, H. 2024. Amplegcg: Learning a universal and transferable generative model of adversarial suffixes for jailbreaking both open and closed llms. *arXiv preprint arXiv:2404.07921*.

Liu, X.; Zhang, T.; Gu, Y.; Iong, I. L.; Xu, Y.; Song, X.; Zhang, S.; Lai, H.; Liu, X.; Zhao, H.; et al. 2024. Visualagentbench: Towards large multimodal models as visual foundation agents. *arXiv preprint arXiv:2408.06327*.

Lù, X. H.; Kasner, Z.; and Reddy, S. 2024. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*.

Lyu, K.; Zhao, H.; Gu, X.; Yu, D.; Goyal, A.; and Arora, S. 2024. Keeping llms aligned after fine-tuning: The crucial role of prompt templates. *arXiv preprint arXiv:2402.18540*.

Mazeika, M.; Phan, L.; Yin, X.; Zou, A.; Wang, Z.; Mu, N.; Sakhaee, E.; Li, N.; Basart, S.; Li, B.; et al. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*.

Picahai, S.; Hassabis, D.; and Kavukcuoglu, K. 2024. https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/.

Pryzant, R.; Iter, D.; Li, J.; Lee, Y. T.; Zhu, C.; and Zeng, M. 2023. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*.

Qi, X.; Panda, A.; Lyu, K.; Ma, X.; Roy, S.; Beirami, A.; Mittal, P.; and Henderson, P. 2024a. Safety alignment should be made more than just a few tokens deep. *arXiv preprint arXiv:2406.05946*.

Qi, X.; Zeng, Y.; Xie, T.; Chen, P.-Y.; Jia, R.; Mittal, P.; and Henderson, P. 2023. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*.

Qi, Z.; Liu, X.; Iong, I. L.; Lai, H.; Sun, X.; Zhao, W.; Yang, Y.; Yang, X.; Sun, J.; Yao, S.; et al. 2024b. WebRL: Training LLM Web Agents via Self-Evolving Online Curriculum Reinforcement Learning. *arXiv preprint arXiv:2411.02337*.

Rawles, C.; Clinckemaillie, S.; Chang, Y.; Waltz, J.; Lau, G.; Fair, M.; Li, A.; Bishop, W.; Li, W.; Campbell-Ajala, F.; et al. 2024. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*.

Tang, L. 2024. A Trivial Jailbreak Against Llama 3. https://github.com/haizelabs/llama3-jailbreak.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

Turner, A. M.; Thiergart, L.; Leech, G.; Udell, D.; Vazquez, J. J.; Mini, U.; and MacDiarmid, M. 2023. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*.

Wang, X.; Chen, Y.; Yuan, L.; Zhang, Y.; Li, Y.; Peng, H.; and Ji, H. 2024. Executable code actions elicit better llm agents. In *International Conference on Machine Learning*.

Wang, X.; Wang, Z.; Liu, J.; Chen, Y.; Yuan, L.; Peng, H.; and Ji, H. 2023. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *arXiv preprint arXiv:2309.10691*.

Wei, A.; Haghtalab, N.; and Steinhardt, J. 2023. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36: 80079–80110.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Winninger, T.; Addad, B.; and Kapusta, K. 2025. Using Mechanistic Interpretability to Craft Adversarial Attacks against Large Language Models. *arXiv preprint arXiv:2503.06269*.

Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; et al. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Yang, C.; Wang, X.; Lu, Y.; Liu, H.; Le, Q. V.; Zhou, D.; and Chen, X. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

Yang, Y.; Sondej, F.; Mayne, H.; and Mahdi, A. 2024b. Ablation is Not Enough to Emulate DPO: How Neuron Dynamics Drive Toxicity Reduction. *arXiv preprint arXiv:2411.06424*.

Yao, S.; Chen, H.; Yang, J.; and Narasimhan, K. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35: 20744–20757.

Zheng, T.; Zhang, G.; Shen, T.; Liu, X.; Lin, B. Y.; Fu, J.; Chen, W.; and Yue, X. 2024. Opencodeinterpreter: Integrating code generation with execution and refinement. *arXiv preprint arXiv:2402.14658*.

Zhou, S.; Xu, F. F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

Zhou, Y.; Muresanu, A. I.; Han, Z.; Paster, K.; Pitis, S.; Chan, H.; and Ba, J. 2022. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*.

Zou, A.; Wang, Z.; Carlini, N.; Nasr, M.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.