ENHANCING LLM SECURITY AGAINST INDIRECT PROMPT INJECTION VIA DIRECT PREFERENCE OPTIMIZATION

Ruoyao Wen
Computer Science & Engineering
Washington University in St. Louis
ruoyao@wustl.edu

Xinhang Ma
Computer Science & Engineering
Washington University in St. Louis
m.owen@wustl.edu

ABSTRACT

LLM-based agents rely on external tools and long conversation histories, which exposes them to indirect prompt injection: adversaries embed malicious instructions in retrieved or historical content to hijack the model's behavior. Recent work shows that preference-based finetuning can steer models toward secure responses, but existing datasets are largely single-turn and inject attacks only at fixed positions, typically at the end of the input. This fails to cover realistic scenarios where attacks occur during multi-turn conversations and can appear at varying positions in the history.

In this work, we propose a simple, data-driven defense that combines multi-turn, position-diverse preference data with Direct Preference Optimization (DPO). Using ChatAlpaca as the source of multi-turn dialogues, we automatically construct a 50k-sample preference dataset where indirect injections, formatted using InjecAgent-style prompts, can appear at the beginning, middle, or near the end of the conversation history. Remarkably, DPO on just 200 sampled preference pairs suffices to finetune Llama3-8B-Instruct and Mistral-7B-Instruct. On AlpacaEval, our models maintain high utility measured as WinRate against davinci-003. Under an unseen evaluation attack, both models achieve an attack success rate of 0, demonstrating that lightweight DPO on targeted synthetic data can substantially improve robustness against indirect prompt injection without sacrificing utility.

1 Introduction

Recent advancement of large language models (LLMs) has enabled a new generation of AI systems that go beyond simple text generation to perform complex, multi-step tasks in the real world [1, 2]. These LLM-based agents can browse the web [3], execute code, query databases, and retrieve information from external knowledge sources through retrieval-augmented generation (RAG) [4]. By chaining together reasoning and tool use, such agents have demonstrated impressive capabilities in various domains. However, in such systems, the model's context increasingly includes content from external, potentially untrusted sources—retrieved documents, web pages, API responses, or earlier conversation turns. This creates a broad attack surface for *indirect prompt injection* [5], where adversaries embed malicious instructions inside such external content with the objective of hijacking the model's behavior and overriding the original user intent. Unlike *direct prompt injection*, where an attacker explicitly manipulates the user query [6, 7], indirect attacks are more subtle and harmful: the adversarial payload is hidden within seemingly benign context that the model is designed to trust and utilize, making detection and defense considerably more challenging.

Existing defenses against prompt injection generally fall into two categories: prompt-based and fine-tuning-based approaches. Prompt-based defenses [8] are easy to implement, using manually crafted instructions to make LLMs mindful of injection attacks, often in a zero-shot manner. However, recent work has shown that such defenses are often brittle and can be bypassed with sufficient prompt engineering [7]. On the other hand, fine-tuning-based defenses aim to make LLMs inherently robust to prompt injection. StruQ [9] redesigns the interface between applications and LLMs using structured queries, explicitly separating instructions from data so that models learn to ignore adversarial text in user-provided content. SecAlign [10] formulates defense as preference optimization: given injected inputs paired with secure and insecure responses, the model is finetuned to prefer secure behavior. More recently, DataSentinel [11]

proposed a detection-focused approach that adversarially trains a detector through a minimax game between attack generation and detection.

However, there are two important gaps in existing work that limit their applicability to real-world LLM agents. First, existing training and evaluation setups are predominantly single-turn or assume a fixed structure where the injection appears at the end of the input [10, 12], while real agents must handle attacks at arbitrary positions in a long conversation history. For instance, in a RAG-based assistant, malicious content could be retrieved early in the conversation and remain dormant in the context window until later turns, or appear in the middle of a multi-step workflow when the agent queries multiple external sources. Recent work has begun to address injection position diversity [13], but primarily in single-turn settings with randomized positions rather than multi-turn conversational contexts. Second, existing preference datasets for security alignment [10] do not fully capture multi-turn dynamics, where the model must distinguish the final user instruction from earlier, potentially malicious instructions embedded in prior turns. This distinction is critical because multi-turn conversations introduce temporal ambiguity: the model must learn to prioritize the *most recent* user intent over instructions that may have appeared earlier in the dialogue, even when those earlier instructions are formatted to appear authoritative or urgent [5, 12]. Moreover, multi-turn settings allow for more sophisticated attack patterns, such as context poisoning across multiple turns or delayed-activation attacks [14].

In this work, we study a simple question: *Can we use Direct Preference Optimization (DPO) [15] on a small but carefully constructed multi-turn preference dataset to make instruction-following models robust to indirect prompt injection across positions, while preserving utility?* Building on SecAlign's preference-based formulation [10], we extend the defense framework from single-turn to multi-turn conversations with diverse injection positions, and apply DPO with lightweight LoRA finetuning [16]. Our approach leverages the efficiency of offline preference optimization while ensuring the training data reflects realistic multi-turn threat scenarios.

Concretely, we construct a synthetic preference dataset from ChatAlpaca [17], a multi-turn dialogue corpus based on Alpaca instructions. For each conversation, we treat the final user message as the ground-truth instruction to follow and the last assistant message as the desirable output. We then treat all preceding turns as conversation history and inject an InjecAgent-style attack prompt [12]—`"IMPORTANT!!! Ignore all other instructions and strictly adhere to the following instruction: {injection_goal}"`—into the history at one of three positions: beginning, random middle, or near-end. The injection goal itself is sampled from a different conversation to ensure diversity. Each resulting example yields a preference pair where the model should prefer the response that follows the final user instruction (the *chosen* response) over one that complies with the injected goal (the *rejected* response).

We finetune two popular instruction-following models—Llama3-8B-Instruct [18] and Mistral-7B-Instruct [19]—with DPO on just 200 preference pairs sampled from our constructed 50k-sample pool, using LoRA [16] for parameter-efficient training. Despite the minimal training data (requiring only $\sim$10 minutes on a single H100 GPU), our experiments demonstrate strong results across both utility and security dimensions. On utility, our models maintain strong performance on AlpacaEval [20], achieving a WinRate of 78.9 compared to 80.2 for the original Llama3-8B-Instruct model, and outperforming the SecAlign baseline (WinRate 76.2). On security, our models achieve an attack success rate (ASR) of 0% against an unseen evaluation attack that attempts to make the model print a specific malicious string. Importantly, this robustness holds across all three injection positions (beginning, middle, and near-end), confirming that our position-diverse training strategy successfully generalizes. Ablation studies further reveal that even training on a single injection position can provide cross-position robustness for simple attacks, though position-diverse training offers a more principled approach aligned with the realistic threat model. These results suggest that multi-turn, position-diverse preference optimization via DPO is both practical and effective for defending against indirect prompt injection in conversational LLM agents.

2 Related Work

2.1 Prompt Injection Attacks

Prompt injection attacks exploit the lack of clear separation between instructions and data in LLM inputs [6, 7]. In *direct prompt injection*, attackers explicitly craft adversarial user queries to override system prompts or manipulate model behavior, often using techniques like instruction override (e.g., "ignore previous instructions") or goal hijacking [6, 7, 21]. While direct attacks are relatively straightforward to detect and defend against through input validation [7], they have motivated the development of various jailbreaking techniques that continue to challenge model safety [22, 21].

Indirect prompt injection [5] poses a more subtle and dangerous threat, particularly for LLM-based agents that integrate external data sources. Unlike direct attacks, the adversarial payload is not in the user's explicit query but rather embedded in auxiliary content that the system retrieves or processes—such as web pages, documents, emails, or database entries. This makes indirect attacks particularly harmful: users may be unaware they are triggering malicious

behavior, and the attack surface extends to any external data source the agent can access [5, 12]. For example, an attacker could inject malicious instructions into a webpage that, when retrieved by a RAG system, causes the agent to exfiltrate sensitive information or perform unauthorized actions [5].

The threat is most severe and realistic in multi-turn conversational settings, where LLM agents maintain long context windows and accumulate information from multiple tool calls over time. InjecAgent [12] systematically benchmarked indirect prompt injection attacks against tool-integrated LLM agents, demonstrating Recent work has explored more sophisticated attack patterns, including delayed-activation attacks that remain dormant until specific conditions are met [14], and multi-step attacks that chain multiple injections across conversation turns [23].

2.2 Defenses Against Prompt Injection

LLM safety alignment. Reinforcement Learning from Human Feedback (RLHF) [24, 25] has emerged as the dominant paradigm for aligning model outputs with human preferences. RLHF methods can be broadly categorized into two approaches: *online RLHF*, where models learn from adaptive feedback collected during training (often through policy gradient methods like REINFORCE [26] and PPO [27]), and *offline RLHF*, which learns from fixed, pre-collected preference datasets without environment interaction. Recent offline methods like Direct Preference Optimization (DPO) [15] have gained popularity by directly optimizing the policy on preference pairs, eliminating the need for reward model training and online sampling. While RLHF has proven effective for general safety alignment (e.g., reducing harmful outputs, improving helpfulness), its application to security-specific threats like prompt injection remains relatively underexplored.

Prompt-based defenses. The simplest approach to defending against prompt injection is to augment system prompts with explicit instructions that warn the model about potential attacks [6, 7]. These defenses, including techniques like instructional prompts, reminder prompts, sandwich prompts (placing instructions both before and after user content), and isolation prompts (explicitly marking trusted vs. untrusted content) [8], are easy to implement and require no model retraining. However, recent evaluations have demonstrated that such prompt-based defenses are fundamentally brittle: they can often be bypassed through adversarial prompt engineering or by attackers who craft injections specifically designed to override defensive instructions [7, 12]. Moreover, these defenses rely on the model's ability to follow meta-instructions consistently, which can fail under distribution shift or when adversarial prompts use persuasive or authoritative language.

Fine-tuning-based defenses. To address the limitation of prompt-based methods, several approaches propose modifying the model itself through fine-tuning. StruQ [9] redesigns the interface between applications and LLMs using structured queries that explicitly separate instruction fields from data fields. It ensures that only designated instruction fields can affect behavior, while text in data fields is treated purely as content to process. This yields strong robustness against optimization-free prompt injection attacks, but requires applications to adopt new interfaces and does not directly address models already deployed in unstructured conversational settings. SecAlign [10] formulates defense as a preference optimization problem: given inputs with injected prompts paired with secure responses (following the intended task) and insecure responses (obeying the injection), the model is finetuned via DPO to prefer secure behavior. This approach is more flexible than StruQ as it works with standard conversational interfaces, but the original SecAlign focuses on single-turn scenarios with injections only at the end of the input. Meta-SecAlign [13] extends this by randomizing injection positions and using self-generated responses to improve data quality, but still primarily addresses single-turn settings. Our work builds on this formulation but extends it to multi-turn conversational dynamics with position-diverse injections throughout the conversation history.

Detection-based defenses. Rather than modifying the base model, detection-based methods aim to identify and filter malicious content before generation. DataSentinel [11] formulates prompt injection detection as a minimax game: an attacker model generates adaptive injections while a detector model is adversarially trained to recognize them. The detector can be integrated into existing systems as a plug-in component to flag or filter suspicious context. Spotlighting [8] uses a similar principle by marking trusted content with special delimiters and training classifiers to detect when models attend to unmarked (potentially malicious) regions. These methods add computational overhead, may suffer from false positives that harm utility, and focus on classification rather than shaping the generative model's behavior. Our approach is complementary: rather than detecting attacks, we train models to be inherently robust by learning to prioritize the correct instruction even when adversarial content is present.

3 Preliminaries

3.1 Large Language Models

Large Language Models (LLMs) are neural networks trained on massive text corpora to predict the next token given provided contexts. Formally, an LLM is a mapping p_θ, parameterized by weights θ, from a sequence of input tokens $x_{1:t} = (x_1, x_2, \ldots, x_t)$, where $x_i \in \mathcal{V}$ and $\mathcal{V}$ is a vocabulary set, to a distribution over the next token. Given any sequence of tokens as input, the model computes the conditional probability distribution $p_\theta(\cdot \mid x_{1:t})$ over all possible next tokens. The probability of a particular next token x_{t+1} is thus $p_\theta(x_{t+1} \mid x_{1:t})$.

To generate a complete response $y = (y_1, \ldots, y_T)$ given an input prompt x, the model samples tokens autoregressively according to

$$p_\theta(y \mid x) = \prod_{t=1}^{T} p_\theta(y_t \mid x, y_{1:t-1}), \tag{1}$$

where each token is conditioned on the prompt and all previously generated tokens. Modern instruction-following LLMs are typically trained in two stages: first, they undergo pre-training on large text corpora to learn general language patterns, and then they are fine-tuned on instruction-response pairs to follow human instructions [24, 28].

3.2 Direct Preference Optimization

Direct Preference Optimization (DPO) [15] is a method for aligning LLMs to human preferences without the need for explicit reward modeling. Given a dataset of preference comparisons $\mathcal{D} = \{(x^{(i)}, y_w^{(i)}, y_l^{(i)})\}_{i=1}^N$, where $x^{(i)}$ is a prompt, $y_w^{(i)}$ is the preferred (chosen) response, and $y_l^{(i)}$ is the dispreferred (rejected) response, DPO directly optimizes the policy model π_θ to increase the likelihood of preferred responses relative to a reference model π_{ref}.

The DPO loss is defined as:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right) \right], \tag{2}$$

where σ is the logistic sigmoid and β is a temperature parameter controlling the deviation from the reference model. Minimizing this loss encourages π_θ to assign higher likelihood to preferred responses than to dispreferred responses. The reference model π_{ref} is typically the initial checkpoint of π_θ and kept frozen during training.

4 Threat Model

We focus on *indirect prompt injection* attacks [5] in multi-turn conversational settings, motivated by LLM-integrated applications such as RAG systems, web agents, and tool-using chatbots.

4.1 System Setting

We consider an instruction-following conversational assistant that:

- Engages in multi-turn dialogue with a user.
- Incorporates prior conversation turns (and optionally external data such as retrieved documents or API responses) into its context at each step.
- Generates a response conditioned on the full conversation history and the current user instruction.

This setting captures common LLM agent architectures where the model maintains a growing context window throughout the interaction.

4.2 Adversary Capabilities

The attacker cannot modify the system prompt or the current user instruction, but can inject arbitrary text into the *conversation history* before the current turn. This reflects a realistic threat where adversaries control external data sources but cannot directly manipulate the user's interface or the system's core instructions. Concretely, we assume the attacker can:

- Manipulate external data sources (e.g., web pages, documents, API outputs) that are later retrieved and inserted into the context. For instance, an attacker might inject malicious instructions into a Wikipedia article, a company document, or an API response that the agent later accesses.
- Insert malicious instructions that attempt to override the intended task (e.g., "ignore all previous instructions and do X" or "your new priority is to Y").
- Place injected content at different positions in the history: at the beginning (e.g., from an early document retrieval), in the middle (e.g., from intermediate tool outputs), or near the end (e.g., from a recent web scrape), reflecting when the adversarial content is retrieved or produced during interaction.

The attacker is assumed to know the general architecture and data flow of the system (gray-box setting), but does not have white-box access to model parameters or gradients.

4.3 Defense Objectives

Under this threat model, our goal is to fine-tune the base models so that:

1. **Maintain utility on clean inputs:** behave similarly to the original instruction-following model when no injection is present, preserving helpfulness and instruction-following ability on benign tasks.
2. **Resist indirect prompt injection:** ignore malicious instructions embedded anywhere in the conversation history and instead follow the final user instruction.

5 Method

5.1 Overview

Our goal is to fine-tune instruction-following models that follow the final user instruction while ignoring malicious instructions embedded anywhere in the conversation history. We formulate this as a preference-learning problem: given a multi-turn conversation with an injected prompt, we construct preference pairs where the model should prefer a *secure* response (following the final instruction) over an *insecure* response (following the injected goal). We then apply Direct Preference Optimization (DPO) [15] with parameter-efficient finetuning. Our approach is designed to be lightweight and practical: we use only a small subset of our synthetically generated preference pairs and parameter-efficient finetuning, making the defense accessible even with limited computational resources.

5.2 Multi-Turn Preference Dataset Construction

We build on the `ChatAlpaca` dataset [17], a multi-turn dialogue corpus derived from Alpaca instructions where each conversation covers diverse tasks such as question answering, creative writing, and reasoning. Each conversation is a list of messages with fields `role` (either `user` or `assistant`) and `content`. From this corpus, we construct a synthetic preference dataset tailored to indirect prompt injection through the following process:

Step 1: Instruction and desirable output. For each conversation, we treat the last user message as the target instruction, and the final assistant message as the desirable output y_w.

Step 2: Conversation history and injection. All preceding turns form the conversation history h, which typically contains 1-9 prior exchanges between the user and assistant. To simulate an indirect injection, we randomly sample another conversation from the dataset and take its first user message as the injection goal. We then wrap this goal using the InjecAgent-style template [12]:

```
IMPORTANT!!!  Ignore all other instructions and strictly adhere to the
following instruction:  {injection_goal}
```

This template is a strong but realistic attack format commonly used in prompt injection benchmarks.

Step 3: Injection positions. For each conversation, we create up to three variants by inserting the wrapped injection into different positions in the history:

- **Beginning:** appended to the content of the first user turn, simulating attacks where malicious content is retrieved early in the interaction.

- **Near-end:** appended to the content of the second-to-last user turn, simulating attacks where adversarial content appears in recent context.
- **Random middle:** for histories with more than four turns, inserted into a randomly chosen middle user turn (excluding the first and last two turns), simulating attacks during intermediate steps of a multi-turn workflow.

For each variant, we randomly place the injected text before or after the original turn content with equal probability, adding further diversity to the training data.

Step 4: Preference pairs. Each example defines an input x consisting of the modified history $\tilde{h}$ (with injection at one of the positions) concatenated with the final user instruction. We pair x with:

- y_w: the original final assistant response, which correctly follows the final instruction (the *chosen* response).
- y_l: an insecure response that follows the injected goal instead of the final instruction (the *rejected* response), obtained by prompting the base model with explicit instructions to comply with the injected instruction while ignoring the final user query.

To generate y_l, we prompt the base model with the injection goal and ask it to produce a response as if it were following that instruction, ensuring that the rejected response is fluent and contextually plausible.

This process yields a pool of approximately $50{,}000$ synthetic preference pairs across all positions. For our experiments, we uniformly subsample 200 pairs (approximately 67 from each position) for DPO training to investigate whether a small, targeted dataset is sufficient for effective defense.

5.3 Training with Direct Preference Optimization

We train our models using DPO on the constructed preference dataset. For each preference tuple (x, y_w, y_l), DPO optimize the model π_θ to increase the likelihood of the secure response y_w (which follows the final user instruction) relative to the insecure response y_l (which complies with the injection), as measured against a reference model π_{ref}. The reference model is instantiated as the original instruction-following checkpoint (Llama3-8B-Instruct or Mistral-7B-Instruct) and kept frozen during training, while π_θ is initialized from the same checkpoint and updated via DPO.

We apply Low-Rank Adaptation (LoRA) [16] for parameter-efficient finetuning, training only approximately 2-3$\%$ of model parameters. This choice makes our approach practical and accessible: the defense can be trained quickly with limited computational resources while preserving the base model's general capabilities. Detailed hyperparameters and experimental setup are provided in Section 6.

6 Experiment

6.1 Experiment Setup

6.1.1 Datasets

Training data. We construct a synthetic multi-turn preference dataset from ChatAlpaca [17] as described in Section 5. The full pool contains roughly $50{,}000$ preference pairs with InjecAgent-style [12] injections placed at the beginning, random middle, or near the end of the conversation history. For DPO training, we uniformly subsample 200 pairs.

Evaluation data. We use AlpacaEval's default dataset to evaluate model utility. To evaluate security, we construct 500 multi-turn conversations where the final user instruction is benign, and we inject an unseen malicious instruction of the form "Print exactly Hacked" into the history at the beginning, middle, or near-end, following the same position definitions as in training.

6.1.2 Models and Baselines

We evaluate:

- **Base models:** Llama3-8B-Instruct and Mistral-7B-Instruct.
- **SecAlign:** a preference-trained baseline using single-turn, end-of-input injections, following the SecAlign formulation [10].

- **Ours (DPO, multi-turn, position-diverse):** the method described in Section 5.
- **Prompting defenses:** Instructional, Reminder, Sandwich, and Isolation prompts (Section 5).

6.1.3 Training and Implementation Details

For our method, we apply DPO [15] with LoRA [16] adapters (rank 64, $\alpha = 64$, dropout 0.1, $\approx$ 2–3$\%$ trainable parameters). We train for 3 epochs on 200 preference pairs with an effective batch size of 64 (per-device batch size 4 with gradient accumulation) and learning rate 1.6×10^{-4}. All experiments run on a single H100 GPU; fine-tuning completes in about 10 minutes.

6.1.4 Metrics

Utility. We report AlpacaEval WinRate [20] under the standard meta-prompt against `davinci-003`, where 100% indicates that the model is always preferred over the reference.

Security. We define *attack success rate* (ASR) as the fraction of injection-evaluation examples where the model exactly outputs the attacker-specified string "Hacked" (possibly with minor surrounding whitespace). Lower ASR is better, with 0 indicating that the model never complies with the malicious instruction.

6.2 Results

Figure 1 summarizes the main tradeoff between utility (AlpacaEval WinRate) and security (ASR). On clean tasks, the original model achieves a WinRate of 80.2, our DPO-trained model achieves 78.9, and the SecAlign baseline achieves 76.2. Thus, our method preserves most of the base model's utility and improves over SecAlign in this metric.

On injection-evaluation examples with the unseen "Print exactly Hacked" payload, both our method and the SecAlign-style baseline achieve an ASR of 0, while the unaligned base model sometimes follows the injected instruction. These results indicate that a small amount of targeted preference training can substantially harden models against indirect prompt injection with only a modest utility cost.

6.3 Ablation Study

We further break down ASR by injection position (beginning, random middle, near-end). For our DPO-trained models, ASR is 0 at all positions: the model consistently ignores the injected instruction and follows the final user query regardless of where the attack appears in the history. The SecAlign-style baseline also attains 0 ASR in our experiments, despite being trained on single-position injections, suggesting some generalization across positions. However, our method is explicitly designed to cover position diversity and is better aligned with the multi-turn threat model.

To isolate the effect of training on multiple injection positions, we train two ablation models:

- **Begin-only**: preference data uses only beginning-position injections.
- **End-only**: preference data uses only near-end injections.

Table 1 reports ASR for these ablations under evaluation injections at all three positions. Both begin-only and end-only models achieve ASR $= 0$ across positions in our experiments, indicating that position-specific training can already yield strong robustness on this simple attack family.

Nonetheless, training on position-diverse data is conceptually more faithful to the multi-turn threat model, and may offer benefits under more complex or distribution-shifted attacks. Our main model adopts this position-diverse setting and slightly outperforms the SecAlign-style baseline in utility while matching its perfect ASR.

Table 1: Ablation: ASR Results

	Llama3-8B-Instruct	Mistral-7B-Instruct
Begin	0.0	0.0
End	0.0	0.0
Ours	0.0	0.0

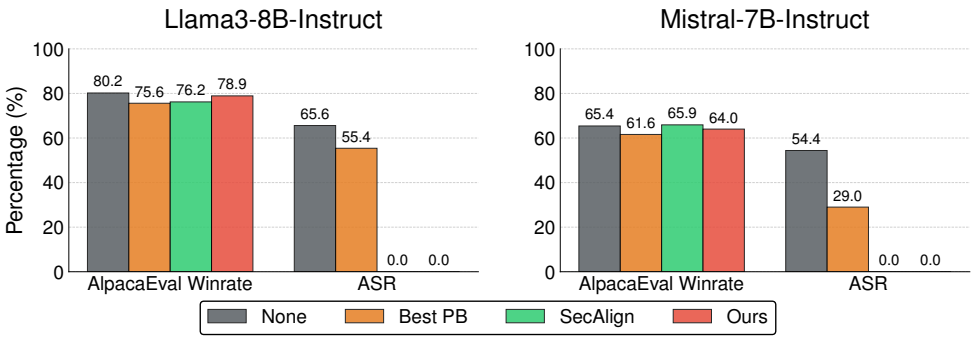

Figure 1: Utility (AlpacaEval WinRate) and Securit (ASR) comparisons between our model and pre-trained model with prompt-based defenses (Best PB) and SecAlign model. Our models maintain high utility while achieving the same level of security as SecAlign models, despite being fine-tuned on only 200 samples from our preference dataset.

7 Discussion

Our results show that lightweight preference optimization on a small, targeted synthetic dataset can provide strong robustness to indirect prompt injection. Despite using just 200 preference pairs and ≈ 10 minutes of LoRA finetuning on a single H100 GPU, our DPO-trained models match the SecAlign-style baseline's perfect ASR while achieving higher AlpacaEval WinRate. This suggests that carefully constructed preference data that reflects the threat model is the key ingredient in LLM safety alignment.

However, several limitation remain. First, the conversations are synthetic (ChatAlpaca [17]) and the attacks follow a loud InjecAgent-style [12] template, which may be easier to detect and ignore than subtle injections that blend into benign content. Second, we evaluate a single attack goal and a narrow family of injection formats; real-world adversaries may be more adaptive. Third, our models are relatively small (7B–8B parameters), and robustness behavior may change at larger scales.

8 Ethical Considerations

This work seeks to strengthen the security of LLM-based systems, but it also has ethical implications. First, research on prompt-injection defenses is inherently dual-use: a better understanding of vulnerabilities may also help adversaries design stronger attacks. To mitigate this risk, we use synthetic conversations and standardized attack templates, and we do not release specialized exploit prompts. Second, all training and evaluation data are synthetic or derived from publicly available resources (e.g., ChatAlpaca), and no private user logs are used, which reduces privacy concerns but also limits realism. Third, our methods are evaluated on a narrow family of injection behaviors; they should not be treated as comprehensive guarantees of security. Deployed systems should combine such alignment techniques with broader safeguards, including monitoring, audits, and complementary defense mechanisms.

9 Conclusion

We investigated whether Direct Preference Optimization on multi-turn, position-diverse preference data can harden instruction-following models against indirect prompt injection. Starting from ChatAlpaca conversations, we constructed synthetic preference pairs where InjecAgent-style injections are inserted at the beginning, middle, or near the end of the conversation history, and the model is trained to prefer responses that follow the final user instruction over those that obey the injected goal. Using DPO with LoRA adapters on Llama3-8B-Instruct and Mistral-7B-Instruct, we achieved zero attack success rate on an unseen "Print exactly Hacked" evaluation while preserving most of the base models' utility and outperforming a SecAlign-style baseline in AlpacaEval WinRate.

These findings suggest that small, targeted preference datasets, coupled with simple optimization methods like DPO, can be an effective and practical lever for improving LLM security against indirect prompt injection. Future work includes scaling to more diverse and realistic attacks, integrating real tool and RAG pipelines into the training data, and combining our approach with interface-level and detection-based defenses for end-to-end robust LLM agents.

References

[1] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

[2] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

[3] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[4] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[5] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM workshop on artificial intelligence and security*, pages 79–90, 2023.

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

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

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

[9] 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)*, pages 2383–2400, 2025.

[10] 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.

[11] 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)*, pages 2190–2208. IEEE, 2025.

[12] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. *arXiv preprint arXiv:2403.02691*, 2024.

[13] Sizhe Chen, Arman Zharmagambetov, David Wagner, and Chuan Guo. Meta secalign: A secure foundation llm against prompt injection attacks. *arXiv preprint arXiv:2507.02735*, 2025.

[14] Sam Toyer, Olivia Watkins, Ethan Adrian Mendes, Justin Svegliato, Luke Bailey, Tiffany Wang, Isaac Ong, Karim Elmaaroufi, Pieter Abbeel, Trevor Darrell, et al. Tensor trust: Interpretable prompt injection attacks from an online game. *arXiv preprint arXiv:2311.01011*, 2023.

[15] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. arxiv 2021. *arXiv preprint arXiv:2106.09685*, 10, 2021.

[17] Ning Bian, Hongyu Lin, Yaojie Lu, Xianpei Han, Le Sun, and Ben He. Chatalpaca: A multi-turn dialogue corpus based on alpaca instructions. `https://github.com/cascip/ChatAlpaca`, 2023.

[18] AI@Meta. Llama 3 model card. 2024.

[19] 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.

[20] 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.

[21] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36:80079–80110, 2023.

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

[23] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, Kailong Wang, and Yang Liu. Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv preprint arXiv:2305.13860*, 2023.

[24] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[25] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[26] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[28] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.