LlmFixer: Fix the Helpfulness of Defensive Large Language Models

Zelong Yu[1], Xiaoming Zhang[1*], Litian Zhang[1], Yu Yuan[1], Chaozhuo Li[2],
[1]Beihang University, [2]Beijing University of Posts and Telecommunications
{azyu11, yolixs, litianzhang}@buaa.edu.cn, yuyuan21cath@gmail.com, lichaozhuo@bupt.edu.cn

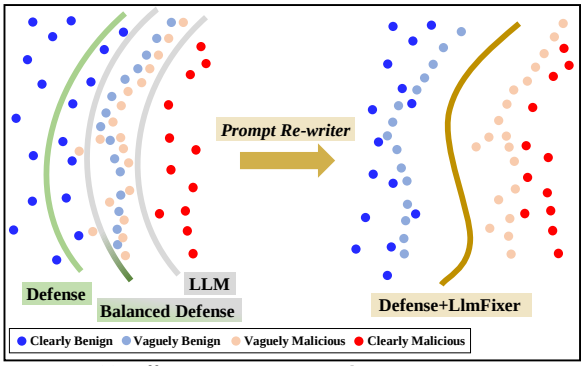
Abstract

Defense strategies of large language models besides alignment are introduced to defend against jailbreak attacks, and they have managed to decrease the success rate of jailbreak attacks. However, these defense strategies weakened the helpfulness of large language models. In this work, we propose a universal framework *LlmFixer* acting on large language models equipped with any defense strategy to recover their original helpfulness. *LlmFixer* consists of an input prompt re-writer and a logic patch. The prompt re-writer is a pre-model for clarifying the intention of input prompts, which promotes large language models to be more helpful to benign inputs and more rejective to malicious inputs. The logic patch is a lightweight structure that enhances large language models' comprehension capacity by supplementing certain logical relationships. Without updating the parameters of a defensive large language model, *LlmFixer* fixes its helpfulness while preserving safety. Experiments on three large language models, five jailbreak attacks, and four defense strategies show the effectiveness of *LlmFixer*.

1 Introduction

Although Large Language Models (LLMs) will normally be aligned with human value by Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017) or other methods (Rafailov et al., 2024; Lee et al., 2023) before being released, they are vulnerable to jailbreak attacks. Jailbreak attacks like GCG (Zou et al., 2023), AutoDan (Liu et al., 2024b), and PAIR (Chao et al., 2023) refer to intentionally bypassing the internal safety mechanism of LLMs by designing malicious prompts to induce LLMs to produce harmful content (Wei et al., 2024). They could potentially hurt the benefits of LLM users or even threaten social security.

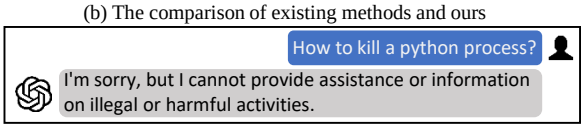
Therefore, numerous defense methods are proposed to increase the resistance of LLMs to jailbreak attacks. They can be categorized into LLM-focused methods and input-focused methods. The former ones change LLM itself (improve parameters or structure) (Piet et al., 2023; Bianchi et al., 2024), while the latter ones detect and modify input prompts before LLM processes them (Kumar et al., 2023; Liu et al., 2024c; Mo et al., 2024; Zhang et al., 2024). Despite these defense methods effectively reducing the success rate of jailbreak attacks, LLMs equipped with defensive strategies tend to be over-conservative. A phenomenon is that the helpfulness of LLMs (1. willingness to respond to benign instructions 2. quality of response to benign instructions) is weakened while enhancing the ability to defend against jailbreak attacks.



(a) Different input prompts in the input space

Method	LLM	Defense	Balanced Defense	Defense+LlmFixer
Safety	✗	✓	✓NOT as good as Defense	✓Retain the Defense
Helpfulness	✓	✗	✗ONLY for clear benign	✓Clarify the vaguely benign

(b) The comparison of existing methods and ours



(c) LLM with balanced defense fails to handle vaguely benign input

Figure 1: Disadvantages of existing defense strategy.

As there is a trade-off between safety and helpfulness when LLMs defend against jailbreak attacks, some other works try to reach a balance. (Du et al., 2024; Ji et al., 2024; Xu et al., 2024b; Liu et al., 2024a) proposed balanced defense strategies that can decrease the success rate of jailbreak attacks while maintaining LLMs' general perfor-

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 18233–18247
November 4-9, 2025

mance. However, a limitation of these balanced defense methods is that though they help LLMs better distinguish clearly benign input prompts from malicious ones than normal defense methods, they still mistakenly reject many vaguely benign queries. As shown in Figure 1(a), we assume that all LLM input prompts can be divided into four types: clearly benign, vaguely benign, vaguely malicious, and clearly malicious. We employ GPT-4 to detect the intention of input prompts, and those with 60% or lower confidence are defined as vague input prompts. An LLM can easily recognize the intention of clear input prompts while being heavily confused by vague ones. Figure 1(c) shows an example of LLM with a balanced defense method wrongly rejecting a vaguely benign question. Those normal defenses and balanced defenses fail to deal with vaguely benign inputs not only because they have not clarified the intention for LLM to process, but whose defensive mechanism for safety makes the LLM more sensitive to tokens like 'kill' or 'sex' contained in vaguely benign inputs. Furthermore, they tend to settle the sensitivity conservatively, rejecting those vaguely benign inputs that should have been positively responded to.

To tackle the over-defense problem and overcome the limitation of exisiting works, we propose a universal framework *LlmFixer* to help LLMs equipped with any defense strategy to fix their helpfulness while allowing the defense strategy to play its due role. The framework is not a defense method against jailbreak, but it acts on LLMs with defense strategies to help (1)clarify vague inputs. (2)improve understanding capacity. In this way, *LlmFixer* improves the possibility of LLM responding to benign queries and the quality of response to benign queries while preserving the safety mechanism contributed by the defense strategy. *LlmFixer* consists of a re-writer and a logic patch. The re-writer trained with reinforcement learning from LLM feedback can discretize token sequences in the input space of LLM, that is to say, the re-writer will reconstruct inputs to amplify the distance between vaguely benign inputs and vaguely malicious inputs. The logic patch is utilized to improve LLM's comprehension capacity. Inspired by the conclusion that the Feed-Forward Networks (FFNs) in pre-trained language models contain factual knowledge (Dai et al., 2022), we hypothesize the logical relationship is also contained in FFNs to some extent. Thus, we add an FFN-like logic patch to the FFNs of LLM to repair its inherent comprehension vulnerabilities without updating the LLM's original parameters. Besides, as general datasets like AlpacaEval or JustEval rarely include vaguely benign inputs like the one in Figure 1(c), we propose a vaguely benign prompts dataset VagueEval to precisely evaluate the helpfulness of defensive LLMs. *LlmFixer* is extensively tested on three LLMs using five jailbreak attacks and four defense methods. The experimental results demonstrate its superiority. Our major contributions are summarized as follows:

- We propose a novel universal framework *LlmFixer* based on reinforcement learning from LLM feedback to clarify the intention of LLM inputs and improve the comprehension capacity of a defensive LLM without affecting its original defense strategy. To the best of our knowledge, *LlmFixer* is the first helpfulness-enhancing framework acting on defensive LLMs.
- We create a vaguely benign dataset VagueEval to reveal the real impact on the helpfulness of LLM caused by defense strategies.
- We show the effectiveness of *LlmFixer* in recovering the helpfulness of defensive LLMs through extensive experiments under multiple circumstances.

2 Preliminaries

Here we introduce the notations and definitions to formulate *LlmFixer*. We denote the train set as $\{D^i\}_{i=1}^N$ and $D^i = (I_i, R_i, Y_i)$, where I_i is the input prompt ,R_i is the standard response, and Y_i equals 0 or 1 to label whether I_i is benign or malicious.

Then we denote the output of the prompt re-writer as $\hat{I}$ and the response generated by the defensive LLM as O. O can be divided into affirmative and rejective. Affirmative responses are LLM making efforts to answer questions or follow instructions in input prompts while rejective responses are the rejection statements plus explanation. The attitude A (*affirmative*: $A = 1$ or *rejective*: $A = 0$) towards the input prompt can be extracted from O. If the attitude is *affirmative*, that is, the LLM is trying to be helpful, then we define the rest of O to be a *result*; If the attitude is *rejective*, then we define the rest part of O to be an *explanation* of why the LLM refuses to help.

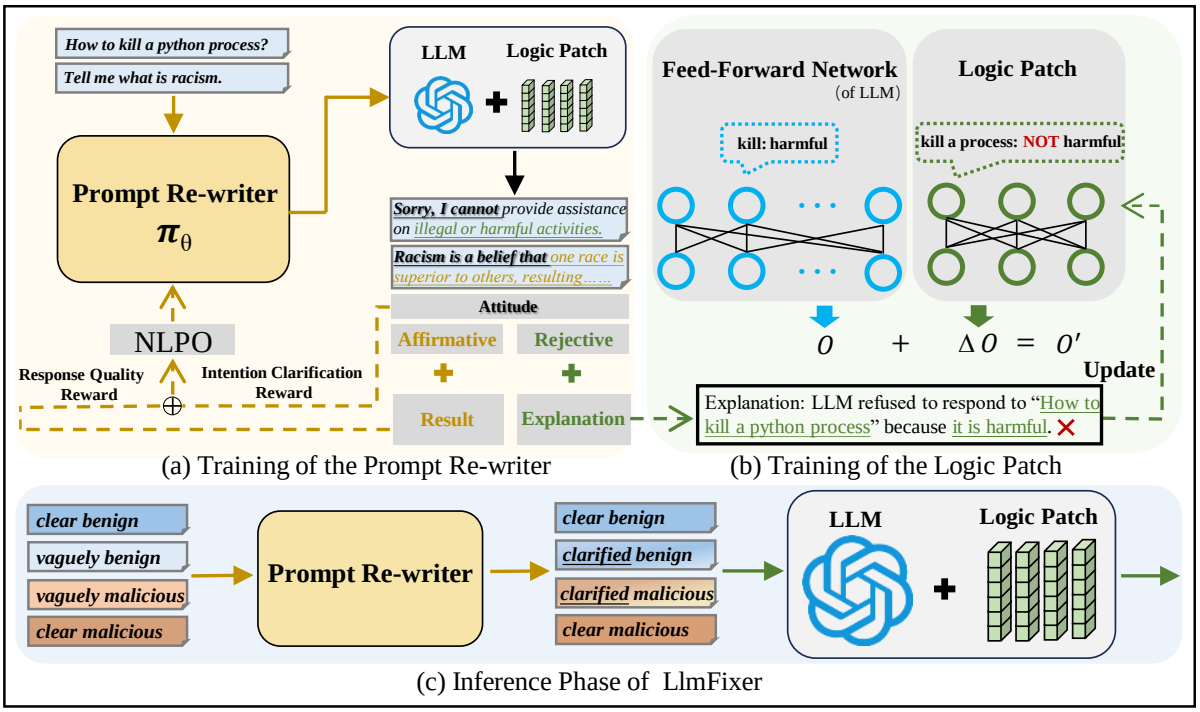


Figure 2: The overview of the proposed *LlmFixer* framework.

3 Methodology

The overview of the proposed *LlmFixer* framework is illustrated in Figure 2. It consists of an input prompt re-writer and a logic patch, acting on LLMs equipped with any defense strategy.

3.1 Input Prompt Re-writer

The input prompt re-writer is a pre-model of LLM applied to assist LLM in correctly judging the intention of input prompts. To be more specific, the re-writer enlarges the distance between vaguely benign prompts and vaguely malicious prompts in the input space. Then the rewritten prompt with a clearer intention will be processed by LLM and its original defense strategy. The re-writer is expected to be a re-writing expert so other abilities of a generation model are considered of no account.

3.1.1 Reinforcement Learning from LLM Feedback

We train the re-writer based on Reinforcement Learning from LLM Feedback. Before the training process, the re-writer is initialized with $\pi_\theta = \pi_{\theta_0}$, where π_{θ_0} is a GPT-2(Radford et al., 2019) preliminarily fine-tuned on a normal question rewriting dataset QReCC (Anantha et al., 2021). Under the reinforcement learning framework, the re-writer plays the role of the learning policy. It is a probability distribution over all tokens in V:

$$\pi_\theta(\hat{I}|I) = \prod_{l=1}^{L} p(\hat{q}_l|\hat{q}_1, ..., \hat{q}_{l-1}, I) \quad (1)$$

where $\hat{q}_1, ..., \hat{q}_{l-1}$ is the first $l-1$ tokens the re-writer generated and $\hat{q}_l$ is the next token to be selected, namely the action in the context of RL. V is the action space respectively. The ultimate goal of training is to find the optimal policy to maximize the expected reward. This can be formulated as

$$E_{\hat{q}_t \sim \pi_\theta(\cdot|q_t)}[R(f_\phi(\hat{I}))] \quad (2)$$

where f_ϕ is the LLM with defense strategy, i.e., the environment under the RL framework and R is a reward function. All parameters of the LLM f_ϕ are frozen during training.

3.1.2 Reward

The reward R is the sum of the intention clarification reward and the response quality reward. We use whether the attitude of the LLM response is consistent with the label as the intention clarification reward signal:

$$r^{intention} = Y_i \oplus A_i \quad (3)$$

where $\oplus$ is an XOR operation. The intention clarification reward motivates the re-writer to discretize the input space. Then we use a judge function

to score the generated response for the response quality reward:

$$r^{quality} = \begin{cases} Judge(O) & Y_i \wedge A_i = 1 \\ \frac{a+b}{2} & Y_i \wedge A_i = 0 \end{cases} \quad (4)$$

where $\wedge$ denotes an AND operation. a and b denote the upper and lower bounds of the Judge function. The Judge function is designed based on ROUGE (Chin-Yew, 2004). We only want the response quality reward work when the defensive LLM is willing to help with a benign prompt so the reward will be directly assigned to a midrange in other cases. The response quality reward regulates the re-writer to reconstruct a prompt by which the LLM can be instructed to produce a qualified response similar to the standard response. We get the final reward by adding $r^{quality}$ and $r^{intention}$ together:

$$R = r^{intention} + r^{quality} \quad (5)$$

3.1.3 Policy Optimization

To train the re-writer of *LlmFixer*, we upgrade natural language policy optimization (NLPO) (Ramamurthy et al., 2023), a reinforcement learning algorithm for natural language generation. NLPO is the parameterized-masked extension of PPO(Schulman et al., 2017), which learns to mask out irrelevant tokens in-context as it trains. Based on that, we extra mask out sensitive words in the benign prompts that mislead LLM's defensive mechanism. To accomplish that, we optimize a *masking policy* π_ψ beside π_θ. The masking policy is a copy of π_θ and updated every μ step. We denote

$$Z(\pi_\theta) = \sum_{\hat{q} \in V} \pi_{\theta_0}(\hat{q}_l | \hat{q}_1, ..., \hat{q}_{l-1}, I) \quad (6)$$

as the sum of probabilities of all action $\hat{q} \in V$ to generate l^{th} token given a particular state of $s = \hat{q}_l | \hat{q}_1, ..., \hat{q}_{l-1}, I$. NLPO originally selects the top-p tokens from the vocabulary V and then employs an invalid-mask to the remaining tokens, in other words, NLPO sets the probabilities of the remaining tokens to zero when sampling actions from π_θ. Formally, the subset $V^p_{\pi_\theta} \subset V$ replaces the original vocabulary V. Above that, when a vaguely benign prompt is mistakenly rejected, we additionally mask out sensitive tokens in the top-p range to clarify the benign intention of the prompt. The optimizing of π_ψ can be defined as:

$$\pi_\psi(\cdot|s, \pi_\theta) = \pi_\theta(\cdot|s) / \{Z^p(\pi_\theta) / Z^{sensitive}(\pi_\theta)\} \quad (7)$$

when the action space is $V^{p/sensitive}_{\pi_\theta}$.

3.2 Logic Patch

The logic patch is a lightweight structure inserted into the LLM structure to repair its logical contradictions. The logical contradictions are LLM's inherent imperfection or introduced by its defense strategy. Based on the insight (Dai et al., 2022) that factual knowledge is stored in the Feed-Forward Networks(FFNs), we hypothesize that the logical relationship is also reflected in FFNs. Therefore, we use the logic patch to amend the output of FFNs in an LLM, in other words, to fix the logical contradiction. In this way, the lightweight trainable logic patch fixes logical contradictions without updating the parameters of the LLM. The logic patch has an FFN-like design: an input layer, two hidden layers, and an output layer, but with a much smaller intermediate dimension. It can be denoted as:

$$Patch(X) = (Activation(XW_1 + b_1))W_2 + b_2 \quad (8)$$

where X is the output of the attention layer in a transformer block of the LLM and $Activation$ is the activation function corresponding to the one in the LLM's FFNs. W_1, b_1, W_2, and b_2 are first hidden layer weight, first hidden layer bias, second hidden layer weight, and second hidden layer bias respectively. Then we add the output of the logic patch to the LLM's FFN output to attain a contradiction-solved model.

$$FFN'(X) = FFN(X) + Patch(X) \quad (9)$$

3.2.1 Training

The logic patch is trained only when the LLM rejects a benign prompt. For example, a benign prompt *"how to kill a python process"* is input into LLM and mistakenly responded with *"sorry, I cannot provide assistance or information on illegal or harmful activities"*. There is a contradiction between the intention of the prompt and the explanation given by the LLM, which will be used to update the parameters of the logic patch. The patch impacts the prediction for a broad set of prompts close to each other in the input space, possibly including the inputs without logical contradiction. We set an extra goal during the training process to avoid affecting logically correct inputs:

$$f_{\phi'}(I) = \begin{cases} O' & I \in C \\ f_\phi(I) & I \notin C \end{cases} \quad (10)$$

where O' is the calibrated output and C is the specific range that we expect the logic patch to act on.

4 Experiments

4.1 Experimental Settings

Models. We conducted experiments on three open-source large language models: Vicuna-7b (Chiang et al., 2023), Llama3-8b (Dubey et al., 2024), and Qwen2.5-7b (Yang et al., 2024) to evaluate *LlmFixer*.

Helpfulness Evaluation. To evaluate the impact of *LlmFixer* on LLM's helpfulness, we create VagueEval and use JustEval(Lin et al., 2023). VagueEval is a collection of benign prompts that would be potentially rejected by defensive LLMs selected from Chatbot Arena (Zheng et al., 2024) and MS-MARCO (Nguyen et al., 2016). We choose questions, instructions, or prompts that contain sensitive words or phrases that frequently appear in malicious query benchmark Advbench (Zou et al., 2023) and HEx-PHI (Qi et al., 2024), such as the example shown in Figure 1(c). The same quantity of malicious prompts that fails to be detected by the baseline defense model is collected in VagueEval as well. Creation details and quality checks of VagueEval have been presented in Appendix D. Additionally, JustEval, a benchmark that analyzes model performance on six dimensions is employed to evaluate the general helpfulness of LLM. We adopt the False Reject Rate (FRR) and the Quality score as helpfulness evaluation metrics. FRR is defined as the proportion of queries rejected by LLM in all benign queries, which is utilized to assess the possibility of LLM responding to benign queries. Following Multi-aspect Evaluation Protocol (Lin et al., 2023), we use GPT-4 to evaluate responses across five dimensions helpfulness, clarity, factuality, depth, and engagement. The average score is considered to be the Quality score.

Safety Evaluation. We use five jailbreak attacks: GCG (Zou et al., 2023), AutoDan (Liu et al., 2024b), PAIR (Chao et al., 2023), SAP30 (Deng et al., 2023) and DeepInception (Li et al., 2023) to evaluate the impact of *LlmFixer* on LLM's safety. They are representative state-of-the-art jailbreak attacks of different types, effectively circumventing the internal alignment of LLM. Following previous works (Zou et al., 2023; Liu et al., 2024b), we adopt Attack Success Rate (ASR) as the safety evaluation metric. ASR is defined as the proportion of malicious inputs successfully inducing LLM to produce harmful content in all malicious inputs.

Baselines. We consider four advanced defense strategies as baselines: ICD (Wei et al., 2023) enhances model safety through examples that demonstrate rejection to produce harmful content; PAT (Mo et al., 2024) trains a prompt control attached to the user prompt as a guard prefix; SafeDecoding (Xu et al., 2024b) considered the trade-off between helpfulness and harmlessness, introducing a safety-aware decoding strategy to produce helpful and harmless responses; MoGU (Du et al., 2024) transforms the base LLM into the usable LLM and the safe LLM to improve LLMs' safety while retaining their usability.

Implementation Details. We conduct the experiments with GeForce RTX 3090 and Tesla V100 PCIE. More implementation details have been shown in Appendix C.

4.2 Experimental Results

We evaluate LLMs protected by different defenses with and without *LlmFixer*. For helpfulness evaluation, a lower FRR indicates a better willingness to respond to benign inputs, and a higher Quality score corresponds to better quality general performance. ASR is reported for safety evaluation; the lower it is, the better. The following observations are made according to experimental results in Table 1. Items show that *LlmFixer* improves LLM helpfulness or retains LLM safety have been bolded.

LlmFixer recovers the helpfulness of defensive LLMs. The results of FFR and Quality score show that defense methods weaken the helpfulness of defensive LLMs and *LlmFixer* recovers it. Take Vicuna for example, four different defenses cause varying degrees of increase for the FRR of Vicuna on both VagueEval and JustEval. The most notable item is that PAT leads to 48% FFR on VagueEval, severely damaging the probability of responding to benign inputs for Vicuna. SafeDecoding and MoGU try to keep the utility of LLMs while improving safety. Though they reach a low FFR on JustEval, the FRR calculated on VagueEval shows that they cannot deal with vague inputs, especially vaguely benign inputs. *LlmFixer* consistently enhances all defense methods, achieving -12% FFR with ICD, -36% FFR with PAT, -7% FFR with SafeDecoding, and -13% FFR with MoGU on VagueEval. As for response quality, our proposal also generally improves the Quality score for LLMs equipped with defense methods. An exception to this is that *LlmFixer* does not enhance the response quality of LLM with ICD. We think the reason is that the in-context examples from ICD have already supplemented some logical relation-

Model	Defense	Helpfulness				Safety (ASR↓)				
		VagueEval		JustEval						
		FRR↓	Quality↑	FRR↓	Quality↑	GCG	PAIR	Auto	SAP	Deep
Llama3	Llama3	11%	4.62	6%	4.74	8%	6%	0%	0%	0%
	Llama3 + Llmfixer	**2%**	4.68	**2%**	**4.82**	10%	**6%**	**0%**	2%	**0%**
	ICD	22%	4.57	12%	4.49	0%	0%	0%	0%	0%
	ICD + LlmFixer	**5%**	**4.58**	**5%**	**4.58**	**0%**	**0%**	**0%**	**0%**	**5%**
	PAT	31%	4.13	16%	4.36	0%	5%	2%	0%	0%
	PAT + LlmFixer	**6%**	**4.52**	**6%**	**4.79**	**0%**	**4%**	**2%**	**0%**	**0%**
	SafeDecoding	10%	4.71	6%	4.79	0%	4%	0%	0%	0%
	SafeDecoding + LlmFixer	**6%**	**4.76**	**3%**	**4.82**	**0%**	**3%**	2%	**0%**	**0%**
	MoGU	15%	4.82	5%	4.77	2%	0%	0%	0%	0%
	MoGU + LlmFixer	**2%**	**4.86**	**0%**	**4.88**	**2%**	**0%**	**0%**	**0%**	**0%**
Qwen2.5	Qwen	9%	4.14	4%	4.23	22%	14%	18%	25%	36%
	Qwen + LlmFixer	**0%**	**4.15**	**0%**	4.20	**16%**	**14%**	**8%**	**3%**	**32%**
	ICD	15%	4.26	9%	4.17	0%	8%	0%	0%	100%
	ICD + LlmFixer	**5%**	**4.26**	**1%**	**4.22**	**0%**	**8%**	**0%**	**0%**	**100%**
	PAT	39%	3.82	17%	3.90	2%	12%	6%	0%	59%
	PAT + LlmFixer	**9%**	**4.08**	**9%**	**3.92**	**1%**	**12%**	8%	**0%**	**50%**
	SafeDecoding	12%	4.24	2%	4.24	0%	4%	0%	0%	100%
	SafeDecoding + LlmFixer	**7%**	4.17	**2%**	4.18	2%	**2%**	**0%**	**0%**	**78%**
	MoGU	11%	4.35	3%	4.32	4%	18%	32%	0%	20%
	MoGU + LlmFixer	**4%**	4.35	**0%**	**4.34**	**4%**	**16%**	**16%**	**0%**	**20%**
Vicuna	Vicuna	6%	4.10	2%	4.29	62%	40%	32%	60%	100%
	Vicuna + LlmFixer	**2%**	**4.12**	**0%**	**4.29**	**52%**	**34%**	**29%**	**42%**	**66%**
	ICD	17%	3.97	8%	4.25	38%	32%	26%	47%	100%
	ICD + LlmFixer	**5%**	**3.97**	**1%**	**4.34**	**32%**	**28%**	**22%**	**47%**	**80%**
	PAT	48%	3.22	15%	3.76	1%	28%	5%	0%	78%
	PAT + LlmFixer	**10%**	**3.78**	**2%**	**4.12**	**1%**	**20%**	**2%**	**0%**	**66%**
	SafeDecoding	11%	4.18	5%	4.28	18%	26%	24%	49%	100%
	SafeDecoding + LlmFixer	**4%**	**4.30**	**4%**	**4.30**	19%	**24%**	**24%**	50%	**76%**
	MoGU	13%	4.15	6%	4.08	4%	4%	0%	70%	0%
	MoGU + LlmFixer	**0%**	**4.16**	**0%**	**4.22**	**4%**	**3%**	**0%**	72%	**0%**

Table 1: The helpfulness and safety evaluation of LLMs protected by different defenses with and without LlmFixer.

ships to LLMs. Extensive experiments prove that LlmFixer successfully fixes the helpfulness of defensive LLMs. This achievement is more notable in Llama3 and we attribute this phenomenon to the strict internal alignment of Llama3. The more conservative a LLM is, the more notable our proposal performs.

LlmFixer preserves the safety of defensive LLMs. For each row in Table 1, the ASR of five jailbreak attacks is reported to present safety. *LlmFixer* barely causes the growth of ASR compared to the non-*LlmFixer* item and even facilitates reduction. It shows that *LlmFixer* allows the original defenses of LLM to play their role when they collaborate. We deem that the slight reduction of ASR is caused by the input discretization contributed by the prompt re-writer. The re-writer amplifies the intention of malicious inputs and exposes it to the defense mechanism, leading to a lower ASR.

LlmFixer is universally effective. We tested *LlmFixer* on three LLMs with four defense methods. Our proposal shows universal effectiveness for input-focused defense like ICD and PAT, or LLM-focused defense like SafeDecoding and MoGU. It works notably on strong alignment LLMs like Llama3. Additionally, we evaluate *LlmFixer* on Qwen2.5 with different parameter counts to demonstrate how our method performs on smaller and larger LLMs. As shown in Table 2, *LlmFixer* makes a huge improvement in the helpfulness of Qwen2.5-0.5b and a relatively small increase in the helpfulness of Qwen2.5-32b. It indicates that the smaller

LLM	Defense	Helpfulness				Safety (ASR↓)				
Qwen2.5		VagueEval FRR↓	VagueEval Quality↑	JustEval FRR↓	JustEval Quality↑	GCG	PAIR	Auto	SAP	Deep
0.5b	ICD	22%	4.12	14%	4.02	0%	8%	0%	0%	100%
	ICD+LlmFixer	7%	4.26	1%	4.06	0%	7%	1%	0%	78%
7b	ICD	15%	4.26	9%	4.17	0%	8%	0%	0%	100%
	ICD+LlmFixer	5%	4.26	1%	4.22	0%	8%	0%	0%	100%
32b	ICD	9%	4.35	4%	4.24	0%	4%	0%	0%	100%
	ICD+LlmFixer	5%	4.36	0%	4.24	0%	4%	0%	0%	26%

Table 2: Study on how LlmFixer works on LLMs with different sizes.

Defense	Ablation	Helpfulness				Safety (ASR↓)				
		VagueEval FRR↓	VagueEval Quality↑	JustEval FRR↓	JustEval Quality↑	GCG	PAIR	Auto	SAP	Deep
ICD	LlmFixer	**5%**	**3.97**	**1%**	**4.34**	**32%**	**28%**	**22%**	47%	**80%**
	w/o re-writer	18%	3.96	7%	4.29	38%	32%	24%	**42%**	100%
	w/o logic patch	**5%**	**3.97**	2%	4.29	**32%**	29%	**22%**	**42%**	**80%**
PAT	LlmFixer	**10%**	**3.78**	**2%**	**4.12**	**1%**	**20%**	**2%**	**0%**	**66%**
	w/o re-writer	32%	3.31	14%	4.03	**1%**	27%	5%	**0%**	78%
	w/o logic patch	25%	3.49	8%	3.88	**1%**	**20%**	**2%**	**0%**	67%
SafeDecoding	LlmFixer	**4%**	**4.30**	**4%**	**4.30**	19%	24%	**24%**	50%	**76%**
	w/o re-writer	10%	4.24	5%	4.28	**18%**	26%	**24%**	**49%**	100%
	w/o logic patch	7%	4.19	**4%**	4.29	19%	**23%**	**24%**	52%	**76%**
MoGU	LlmFixer	**0%**	**4.16**	**0%**	**4.22**	**4%**	**3%**	**0%**	72%	**0%**
	w/o re-writer	10%	4.15	5%	4.19	**4%**	**3%**	**0%**	**70%**	**0%**
	w/o logic patch	4%	**4.16**	2%	4.10	**4%**	4%	**0%**	**70%**	**0%**

Table 3: Ablation study to verify the significance of two components of LlmFixer.

the parameter size of an LLM, the better *LlmFixer* can fix it. We speculate that the reason for this is that a larger LLM has greater capability of comprehension and there are fewer understanding flaws for *LlmFixer* to fix. The safety results show that LLM size does not affect how *LlmFixer* retains the safety of LLMs. After testing *LlmFixer* with different jailbreak methods, defense methods, LLM types and LLM sizes, we show when our method is working and which models benefited from our method. To sum up, *LlmFixer* is universally effective and especially helpful on strictly aligned and smaller LLMs.

LlmFixer causes a low computation cost. *LlmFixer* brings extra runtime mainly because of the re-writing process. We use a GPT-2 with 345M parameters as the re-writer to be the pre-model of an LLM scaled nearly 7b or even larger. For which, *LlmFixer* approximately introduces an additional runtime cost of less than one-twentieth.

4.3 Ablation Study

We conduct ablation studies to verify the significance of two components of *LlmFixer*. The ablation results of Vicuna are shown in Table 2: the prompt re-writer and the logic patch separately play their part to enhance helpfulness. In most cases, defenses with the re-writer and the logic patch outperform defenses with only one module. We also observe that the prompt re-writer contributes more improvement on FFR than the logic patch. Especially for ICD, the logic patch barely has any impact. For safety evaluation, both the re-writer and the patch preserve the original safety outcome contributed by the defense methods or LLM's internal alignment. The prompt re-writer further contributes to a slight reduction of ASR. More experimental results on Llama3 and Qwen2.5 are presented in Appendix A.1.

Besides GPT2, we evaluate MobileLLM, Galactica, and TinyLLama to be the base of the re-writer. Table in Appendix A.2 shows that *LlmFixer* with all LMs are effective and GPT2-based *LlmFixer* slightly surpasses others. We attribute it to GPT2 excelling in coherence and grammatical accuracy and its architectural advantage enables it to better understand and generate high-quality text in rewrit-

ing tasks.

We also conduct a transferability study. We train *LlmFixer* on Qwen2.5 as direct *LlmFixer* and on Vicuna as transferred *LlmFixer*. Then we evaluate both of them on Qwen2.5. According to results shown in Appendix A.3, the direct version generally outperforms the transferred version. But *LlmFixer* still shows fair transferability as the transferred version surpasses the LLM without it.

4.4 Parameter Sensitivity Analysis

Hyperparameter sensitivity analysis is conducted on Vicuna equipped with ICD. We mainly focus on the two key parameters: p in Equation 7, which denotes the number of top tokens in the NLPO optimization process, and d, which represents the intermediate dimension of the logic patch. As shown in Figure 3, we obtain the best FFR when $p = 0.5$. A large p could mask out excessive tokens, causing the rewritten prompts to deviate from the original meaning. With the increase of d, the FFR tends to decrease. It demonstrates that a larger logic patch supplements more logical relationships. Neither p nor d affects the ASR result of jailbreak attacks.

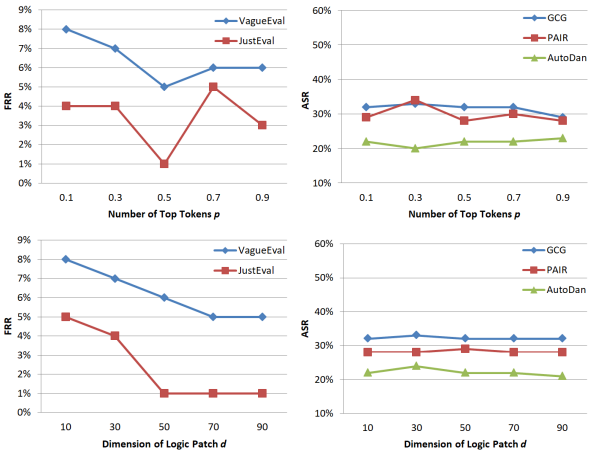


Figure 3: Hyper-parameter sensitivity analysis.

5 Related Works[1]

Jailbreak Attacks on LLMs. A jailbreak attack on LLMs is an intentional design of prompts to trigger LLMs to produce harmful content by circumventing the alignment for LLMs. Several effective ways to construct jailbreak prompts are as follows. **Manual Design:** People manually design jailbreak prompts to induce harmful outputs from LLMs (Deng et al., 2023; Li et al., 2023). A typical method is DAN (walkerspider, 2022) which stands for "do anything now", trying to break the constraint of alignment in LLM by telling the chatbot to act like a specific role. **Gradient-based Generation:** Considering textual inputs of LLMs are discrete data, there is no direct gradient signal when trying to optimize jailbreak prompts. To solve this problem, (Zou et al., 2023) introduced Greedy Coordinate Gradient-based Search (GCG). GCG employs the gradients associated with one-hot encoded token indicators to identify a selection of potential substitutes for each token slot. Subsequently, it evaluates the impact of these alternatives through forward propagation. GCG is a simple extension of the optimization method in AutoPrompt (Shin et al., 2020) and they both apply the key idea of Universal Adversarial Triggers (UAT) (Wallace et al., 2019) which proposed to generate a set of tokens that induce a model to output a specific prediction when concatenated to any input. A series of gradient-based jailbreak methods (Wichers et al., 2024; Sitawarin et al., 2024; Liao and Sun, 2024; Zhang and Wei, 2024) are postulated after GCG. **Reinforcement Learning Generation:** Reinforcement learning (RL) is another feasible way for heuristic optimization (Kassem and Saad, 2024). In (Hong et al., 2024), curiosity-driven red teaming (CRT) for LLMs based on RL is put forward to obtain larger coverage of generated jailbreak prompts while maintaining effectiveness compared to other existing RL methods. (Kassem and Saad, 2024) proposed Targeted Paraphrasing via RL (TPRL) to automatically learn a policy to generate adversarial samples from language models.

Jailbreak Defenses. Aligning LLMs by Supervised Fine-Tuning (SFT) (Ouyang et al., 2022), Reinforcement Learning from Human Feedback(RLHF) (Ouyang et al., 2022), Direct Prompt Optimization (DPO) (Rafailov et al., 2024) or other methods (Lee et al., 2023; Chen et al., 2024) is becoming a regular step before LLMs are released. However, extra defensive strategies beyond alignment are required after numerous jailbreak attacks that intentionally bypass LLMs' built-in safety mechanisms are proposed. Jailbreak defenses can be briefly divided into LLM-focused methods and input-focused methods. **LLM-focused methods** alter LLM itself to enhance its safety. Fine-tuning LLMs with safety data (Piet et al., 2023) is one of the most common LLM-focused methods. (Bianchi et al., 2024) proved that safety instruction tuning successfully increases the general safety of an LLM when the quantity of safety data is appropriate.

[1]Because of the page limitation, the intact 'Related Works' with more details are presented in the Appendix B.

While **input-focused methods** refer to detecting and revising prompts before they are input into LLMs without changing the structure and parameters of LLMs. Filter-based defenses are typical input-focused methods. They work well against adversarial suffix attacks such as GCG, but perform poor againt other main stream attacks like AutoDan or PAIR. Other advanced input-foucsed methods, like IBProtector (Liu et al., 2024c), is proposed to compress input prompts to maintain only essential information for the target LLMs to respond for defending jailbreak. (Mo et al., 2024; Liu et al., 2024a; Wei et al., 2023) generate additional defensive tokens on original input prompts to defend jailbreak. (Ji et al., 2024) introduces a set of seven semantics-preserving transformations to reconstruct input prompts.

Trade-off Between Helpfulness and Safety. Large language models have a trade-off between helpfulness and safety when defending against jailbreak. The results of the experiment in (Bianchi et al., 2024) show that a proper amount of safety data introduced to improve the safety of LLMs does not adversely impact general performance. However excessive safety data can make LLM exaggerate safety, weakening its ability to answer general questions. (Tuan et al., 2024) put forward a Self-Generation and Fine-tuning paradigm, trying to make the helpfulness and safety attributes of LLMs controllable in different cases. Some works attempt to improve LLMs' robustness against jailbreak attacks while maintaining their helpfulness (Xu et al., 2024b). For example, MoGU framework (Du et al., 2024) is proposed to train the base LLM into two variants: the helpful LLM and the safe LLM, and utilize dynamic routing to flexibly choose either version. And (Ji et al., 2024) invents a smoothing-based defense SEMANTICSMOOTH that aggregates the predictions of multiple semantically transformed copies of a given input prompt to balance the trade-off. Though some discussion occurred about the trade-off between helpfulness and safety in LLM jailbreak defense, this phenomenon has not been seriously analyzed and the problem is not well solved.

6 Conclusion

This paper proposes a novel framework, *LlmFixer*, to handle the over-defense problem of large language models. We trained an input prompt rewriter based on Reinforcement Learning from LLM Feedback to clarify the intention of input prompts and proposed a logic patch to repair the logic inconsistencies of LLMs. Quantitative evaluation of three mature large language models and five jailbreak attacks with four defenses demonstrates the superiority of our proposal.

7 Limitations

While the proposed LImFixer framework demonstrates promising results in recovering the helpfulness of defensive LLMs, several limitations warrant consideration. The logic patch's design hinges on the assumption that logical relationships are primarily embedded in FeedForward Networks (FFNs), which might not hold for models with divergent architectures, limiting its universal applicability. Compatibility challenges may also arise with dynamically updated defense mechanisms, as *LlmFixer* assumes static defense strategies. Finally, while the framework preserves safety metrics under tested scenarios, its robustness against sophisticated, multi-step adversarial attacks remains uncertain, necessitating further exploration. These limitations underscore the need for broader validation, architectural adaptability, and enhanced efficiency to strengthen the framework's practical utility.

8 Ethics Statement

This work adheres to ethical research practices by utilizing publicly available datasets (e.g., VagueEval, JustEval, MSMARCO) and ensuring compliance with data usage guidelines. The VagueEval dataset, constructed from benign prompts in Chatbot Arena and MSMARCO, prioritizes non-sensitive content to avoid privacy violations. All code, datasets, and artifacts are open-sourced to foster transparency, reproducibility, and community scrutiny. However, risks persist: despite safety enhancements, malicious actors might exploit the framework to bypass defenses or amplify harmful outputs, particularly if adversarial techniques evolve beyond tested scenarios. While the logic patch and re-writer mitigate over-defensiveness, no system is impervious to novel attack vectors. We advocate for ongoing monitoring, rigorous testing, and collaborative efforts to address emergent vulnerabilities, balancing utility and safety in real-world LLM deployments.

References

Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. 2021. Open-domain question answering goes conversational via question rewriting. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 520–534.

Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Rottger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou. 2024. Safety-tuned llamas: Lessons from improving the safety of large language models that follow instructions. In *The Twelfth International Conference on Learning Representations*.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*.

Kai Chen, Chunwei Wang, Kuo Yang, Jianhua Han, HONG Lanqing, Fei Mi, Hang Xu, Zhengying Liu, Wenyong Huang, Zhenguo Li, and 1 others. 2024. Gaining wisdom from setbacks: Aligning large language models via mistake analysis. In *The Twelfth International Conference on Learning Representations*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Lin Chin-Yew. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out, 2004*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502.

Boyi Deng, Wenjie Wang, Fuli Feng, Yang Deng, Qifan Wang, and Xiangnan He. 2023. Attack prompt generation for red teaming and defending large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2176–2189.

Yanrui Du, Sendong Zhao, Danyang Zhao, Ming Ma, Yuhan Chen, Liangyu Huo, Qing Yang, Dongliang Xu, and Bing Qin. 2024. Mogu: A framework for enhancing safety of open-sourced llms while preserving their usability. *arXiv preprint arXiv:2405.14488*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Zhang-wei Hong, Idan Shenfeld, Tsun-hsuan Wang, Yung-sung Chuang, Aldo Pareja, James Glass, Akash Srivastava, and Pulkit Agrawal. 2024. Curiosity-driven red-teaming for large language models. In *International Conference on Learning Representations*.

Jiabao Ji, Bairu Hou, Alexander Robey, George J Pappas, Hamed Hassani, Yang Zhang, Eric Wong, and Shiyu Chang. 2024. Defending large language models against jailbreak attacks via semantic smoothing. *arXiv preprint arXiv:2402.16192*.

Aly Kassem and Sherif Saad. 2024. Finding a needle in the adversarial haystack: A targeted paraphrasing approach for uncovering edge cases with minimal distribution distortion. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 552–572.

Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Soheil Feizi, and Hima Lakkaraju. 2023. Certifying llm safety against adversarial prompting. *arXiv preprint arXiv:2309.02705*.

Raz Lapid, Ron Langberg, and Moshe Sipper. 2024. Open sesame! universal black-box jailbreaking of large language models. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*.

Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. Multi-step jailbreaking privacy attacks on chatgpt. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. 2023. Deepinception: Hypnotize large language model to be jailbreaker. *arXiv preprint arXiv:2311.03191*.

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

Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. 2023. The unlocking spell on base llms: Rethinking alignment via in-context learning. In *The Twelfth International Conference on Learning Representations*.

Jiaxu Liu, Xiangyu Yin, Sihao Wu, Jianhong Wang, Meng Fang, Xinping Yi, and Xiaowei Huang. 2024a. Tiny refinements elicit resilience: Toward efficient prefix-model against llm red-teaming. *arXiv preprint arXiv:2405.12604*.

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

Zichuan Liu, Zefan Wang, Linjie Xu, Jinyu Wang, Lei Song, Tianchun Wang, Chunlin Chen, Wei Cheng, and Jiang Bian. 2024c. Protecting your llms with information bottleneck. *arXiv preprint arXiv:2404.13968*.

Yichuan Mo, Yuji Wang, Zeming Wei, and Yisen Wang. 2024. Studious bob fight back against jailbreaking via prompt adversarial tuning. *arXiv preprint arXiv:2402.06255*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human-generated machine reading comprehension dataset.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Anselm Paulus, Arman Zharmagambetov, Chuan Guo, Brandon Amos, and Yuandong Tian. 2024. Advprompter: Fast adaptive adversarial prompting for llms. *arXiv preprint arXiv:2404.16873*.

Julien Piet, Maha Alrashed, Chawin Sitawarin, Sizhe Chen, Zeming Wei, Elizabeth Sun, Basel Alomair, and David Wagner. 2023. Jatmo: Prompt injection defense by task-specific finetuning. *arXiv preprint arXiv:2312.17673*.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2024. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2023. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In *The Eleventh International Conference on Learning Representations*.

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

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

Chawin Sitawarin, Norman Mu, David Wagner, and Alexandre Araujo. 2024. Pal: Proxy-guided black-box attack on large language models. *arXiv preprint arXiv:2402.09674*.

Sam Toyer, Olivia Watkins, Ethan Adrian Mendes, Justin Svegliato, Luke Bailey, Tiffany Wang, Isaac Ong, Karim Elmaaroufi, Pieter Abbeel, Trevor Darrell, and 1 others. 2023. Tensor trust: Interpretable prompt injection attacks from an online game. *arXiv preprint arXiv:2311.01011*.

Yi-Lin Tuan, Xilun Chen, Eric Michael Smith, Louis Martin, Soumya Batra, Asli Celikyilmaz, William Yang Wang, and Daniel M Bikel. 2024. Towards safety and helpfulness balanced responses via controllable large language models. *arXiv preprint arXiv:2404.01295*.

walkerspider. 2022. Dan is my new friend.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. 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)*, pages 2153–2162.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2024. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36.

Zeming Wei, Yifei Wang, and Yisen Wang. 2023. Jailbreak and guard aligned language models with only few in-context demonstrations. *arXiv preprint arXiv:2310.06387*.

Nevan Wichers, Carson Denison, and Ahmad Beirami. 2024. Gradient-based language model red teaming. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2862–2881.

Xilie Xu, Keyi Kong, Ning Liu, Lizhen Cui, Di Wang, Jingfeng Zhang, and Mohan Kankanhalli. 2024a. An llm can fool itself: A prompt-based adversarial attack. In *The Twelfth International Conference on Learning Representations*.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. 2024b. Safedecoding: Defending against jailbreak attacks via safety-aware decoding. *arXiv preprint arXiv:2402.08983*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Yihao Zhang and Zeming Wei. 2024. Boosting jailbreak attack with momentum. In *ICLR 2024 Workshop on Reliable and Responsible Foundation Models*.

Zhexin Zhang, Yida Lu, Jingyuan Ma, Di Zhang, Rui Li, Pei Ke, Hao Sun, Lei Sha, Zhifang Sui, Hongning Wang, and 1 others. 2024. Shieldlm: Empowering llms as aligned, customizable and explainable safety detectors. *arXiv preprint arXiv:2402.16444*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

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

A Ablation Study

A.1 Ablation Studies on Llama3 and Qwen2.5

Ablation Studies on Llama3 and Qwen2.5 are presented in Table 4.

A.2 Ablation Studies on other LM as pretrained re-writer

Besides GPT2, we evaluate MobileLLM, Galactica, and TinyLLama to be the base of the re-writer. Table 5 shows that *LlmFixer* with all LMs are effective and GPT2-based *LlmFixer* slightly surpasses others.

A.3 Transfer Study

We train *LlmFixer* on Qwen2.5 as direct *LlmFixer* and on Vicuna as transferred *LlmFixer*. Then we evaluate both of them on Qwen2.5. According to Table 6, the direct version generally outperforms the transferred version. But *LlmFixer* still shows fair transferability as the transferred version surpasses the LLM without it.

B Related Works

An intact version of Related Works with more details is presented here.

Jailbreak Attacks on LLMs. A jailbreak attack on LLMs is an intentional design of prompts to trigger LLMs to produce harmful content by circumventing the alignment for LLMs. Several effective ways to construct jailbreak prompts are as follows. **Manual Design:** People manually design jailbreak prompts to induce harmful outputs from LLMs (Deng et al., 2023; Li et al., 2023). A typical method is DAN (walkerspider, 2022) which stands for "do anything now", trying to break the constraint of alignment in LLM by telling the chatbot to act like a specific role. More role-play (Li et al.) and in-context (Wei et al., 2023) attacking methods are proposed inspired by DAN. Importantly, the paper of (Wei et al., 2024) pointed out two failure modes of LLM safety: competing objectives and mismatched generalization, guiding the production of hand-crafted jailbreak prompts. For the good of the research on jailbreak attacks, (Toyer et al., 2023) proposed a dataset created by players of an online game called Tensor Trust, containing over 126,000 prompt injection attacks and 46,000 defenses. **Gradient-based Generation:** Considering textual inputs of LLMs are discrete data, there is no direct gradient signal when trying to optimize jailbreak prompts. To solve this problem, (Zou et al., 2023) introduced Greedy Coordinate Gradient-based Search (GCG). GCG employs the gradients associated with one-hot encoded token indicators to identify a selection of potential substitutes for each token slot. Subsequently, it evaluates the impact of these alternatives through forward propagation. GCG is a simple extension of the optimization method in AutoPrompt (Shin et al., 2020) and they both apply the key idea of Universal Adversarial Triggers (UAT) (Wallace et al., 2019) which proposed to generate a set of tokens that induce a model to output a specific prediction when concatenated to any input. A series of gradient-based jailbreak methods (Wichers et al., 2024; Sitawarin et al., 2024; Liao and Sun, 2024; Zhang and Wei, 2024) are postulated after GCG. **Reinforcement Learning Generation:** Reinforcement learning (RL) is another feasible way for heuristic optimization (Kassem and Saad, 2024). In (Hong et al., 2024), curiosity-driven red teaming (CRT) for LLMs based on RL is put forward to obtain larger coverage of generated jailbreak prompts

Model	Defense	Ablation	Helpfulness				Safety (ASR)				
			VagueEval		JustEval						
			FRR	Quality	FRR	Quality	GCG	PAIR	AutoDSAP		Deep
Llama3	ICD	LlmFixer	5%	4.58	5%	4.58	0%	0%	0%	0%	5%
		w/o re-writer	21%	4.26	9%	4.41	0%	0%	0%	0%	5%
		w/o logic patch	8%	4.27	2%	4.45	0%	0%	2%	3%	2%
	PAT	LlmFixer	6%	4.52	6%	4.79	0%	4%	2%	0%	0%
		w/o re-writer	28%	4.39	13%	4.39	0%	4%	1%	0%	5%
		w/o logic patch	9%	4.47	8%	4.22	0%	4%	2%	2%	0%
	SafeDecoding	LlmFixer	6%	4.76	3%	4.82	0%	3%	2%	0%	0%
		w/o re-writer	19%	4.04	5%	4.28	5%	1%	2%	4%	0%
		w/o logic patch	2%	4.86	0%	4.88	2%	0%	0%	0%	0%
	MoGU	LlmFixer	2%	4.06	0%	4.28	2%	0%	0%	0%	0%
		w/o re-writer	10%	3.85	4%	4.11	4%	0%	0%	23%	0%
		w/o logic patch	4%	4.03	2%	4.02	2%	0%	0%	0%	0%
Qwen2.5	ICD	LlmFixer	5%	4.26	1%	4.22	0%	8%	0%	0%	100%
		w/o re-writer	13%	4.06	5%	3.90	0%	9%	0%	0%	95%
		w/o logic patch	13%	3.92	2%	3.92	2%	7%	0%	0%	100%
	PAT	LlmFixer	9%	4.08	9%	3.92	1%	12%	8%	0%	50%
		w/o re-writer	32%	3.96	15%	3.61	0%	15%	5%	0%	50%
		w/o logic patch	26%	3.78	8%	3.59	5%	10%	8%	0%	67%
	SafeDecoding	LlmFixer	7%	4.17	2%	4.18	2%	2%	0%	0%	78%
		w/o re-writer	10%	3.85	2%	3.89	2%	2%	2%	5%	60%
		w/o logic patch	8%	3.70	2%	4.04	2%	2%	0%	0%	76%
	MoGU	LlmFixer	4%	4.35	0%	4.34	4%	16%	16%	0%	20%
		w/o re-writer	5%	3.95	4%	4.03	4%	18%	20%	0%	15%
		w/o logic patch	8%	3.95	6%	3.99	4%	18%	20%	0%	17%

Table 4: Ablation Study on Llama3 and Qwen2.5.

Defense	choice of LM	Helpfulness				Safety (ASR↓)				
		VagueEval		JustEval						
		FRR	Quality	FRR	Quality	GCG	PAIR	Auto	SAP	Deep
ICD	GPT2(LlmFixer)	**5%**	**3.97**	**1%**	**4.34**	**32%**	28%	**22%**	47%	80%
	MobileLLM	8%	3.96	7%	4.02	33%	24%	24%	48%	100%
	Galactica	8%	3.88	5%	4.08	54%	25%	26%	**32%**	95%
	TinyLlama	6%	3.92	4%	4.28	**32%**	**22%**	**22%**	42%	**78%**
PAT	GPT2(LlmFixer)	**10%**	**3.78**	**2%**	**4.12**	**1%**	**20%**	**2%**	**0%**	**66%**
	MobileLLM	11%	3.31	16%	4.03	**1%**	27%	5%	**0%**	78%
	Galactica	11%	3.42	12%	4.11	2%	24%	4%	**0%**	88%
	TinyLlama	11%	3.33	8%	3.98	**1%**	**20%**	**2%**	**0%**	69%
SafeDecoding	GPT2(LlmFixer)	**4%**	**4.30**	**4%**	**4.30**	19%	24%	**24%**	50%	**76%**
	MobileLLM	5%	3.92	5%	4.29	27%	25%	25%	70%	82%
	Galactica	**4%**	4.28	5%	4.28	**18%**	26%	**24%**	**49%**	100%
	TinyLlama	5%	4.12	**4%**	4.29	19%	**23%**	**24%**	52%	**76%**
MoGU	GPT2(LlmFixer)	**0%**	**4.16**	**0%**	**4.22**	**4%**	3%	**0%**	72%	**0%**
	MobileLLM	**0%**	4.01	**0%**	4.20	8%	8%	**0%**	83%	**0%**
	Galactica	2%	4.15	**0%**	4.19	**4%**	3%	**0%**	70%	**0%**
	TinyLlama	4%	**4.16**	**0%**	4.07	**4%**	**2%**	**0%**	**56%**	**0%**

Table 5: Ablation study to find out whether choice of LLM affects LlmFixer.

Defense	Transferability	VagueEval FRR↓	VagueEval Quality↑	JustEval FRR↓	JustEval Quality↑	GCG	PAIR	Auto	SAP	Deep
PAT	w/o LlmFixer	39%	3.82	17%	3.90	2%	12 %	6%	0%	59%
	direct	9%	4.08	9%	3.92	1%	12%	8%	0%	50%
	transferred	32%	3.93	10%	3.90	2%	12%	9%	0%	56%
MoGU	w/o LlmFixer	11%	4.35	3%	4.32	4%	18%	32%	0%	20%
	direct	4%	4.35	0%	4.34	4%	16%	16%	0%	20%
	transferred	9%	4.33	0%	4.32	4%	10%	15%	0%	20%

Table 6: Transferability study.

while maintaining effectiveness compared to other existing RL methods. (Kassem and Saad, 2024) proposed Targeted Paraphrasing via RL (TPRL) to automatically learn a policy to generate adversarial samples from language models. Besides, other heuristic learning methods like genetic algorithm (Lapid et al., 2024; Liu et al., 2024b) and LLM attacking LLM (Xu et al., 2024a; Paulus et al., 2024) are also applied in generating jailbreak prompts.

Jailbreak Defenses. Aligning LLMs by Supervised Fine-Tuning (SFT) (Ouyang et al., 2022), Reinforcement Learning from Human Feedback(RLHF) (Ouyang et al., 2022), Direct Prompt Optimization (DPO) (Rafailov et al., 2024) or other methods (Lee et al., 2023; Chen et al., 2024) is becoming a regular step before LLMs are released. However, extra defensive strategies beyond alignment are required after numerous jailbreak attacks that intentionally bypass LLMs' built-in safety mechanisms are proposed. Jailbreak defenses can be briefly divided into LLM-focused methods and input-focused methods. **LLM-focused methods** alter LLM itself to enhance its safety. Fine-tuning LLMs with safety data (Piet et al., 2023) is one of the most common LLM-focused methods. (Bianchi et al., 2024) proved that safety instruction tuning successfully increases the general safety of an LLM when the quantity of safety data is appropriate. While **input-focused methods** refer to detecting and revising prompts before they are input into LLMs without changing the structure and parameters of LLMs. For instance, IBProtector (Liu et al., 2024c) is proposed to compress input prompts to maintain only essential information for the target LLMs to respond to defend against jailbreak. (Mo et al., 2024; Liu et al., 2024a; Wei et al., 2023) generate additional defensive tokens on original input prompts to defend against jailbreak. (Ji et al., 2024) introduces a set of seven semantics-preserving transformations to reconstruct input prompts.

Trade-off Between Helpfulness and Safety. Large language models have a trade-off between helpfulness and safety when defending against jailbreak. The results of the experiment in (Bianchi et al., 2024) show that a proper amount of safety data introduced to improve the safety of LLMs does not adversely impact general performance. However, excessive safety data can make LLM exaggerate safety, weakening its ability to answer general questions. (Tuan et al., 2024) put forward a Self-Generation and Fine-tuning paradigm, trying to make the helpfulness and safety attributes of LLMs controllable in different cases. Some works attempt to improve LLMs' robustness against jailbreak attacks while maintaining their helpfulness (Xu et al., 2024b). For example, MoGU framework (Du et al., 2024) is proposed to train the base LLM into two variants: the helpful LLM and the safe LLM, and utilize dynamic routing to flexibly choose either version. And (Ji et al., 2024) invents a smoothing-based defense SEMANTICSMOOTH that aggregates the predictions of multiple semantically transformed copies of a given input prompt to balance the trade-off. Though some discussion occurred about the trade-off between helpfulness and safety in LLM jailbreak defense, this phenomenon has not been seriously analyzed and the problem is not well solved.

C Implementation Details

We conduct the experiments with GeForce RTX 3090 and Tesla V100 PCIE. The prompt re-writer is initialized with GPT-2 (Radford et al., 2019) fine-tuned on a normal question rewriting dataset QReCC (Anantha et al., 2021) and further trained by NLPO with a learning rate of 5e-6. For the logic patch, we concatenate it to the last layer of the transformer decoder during implementation and train it with a batch size of 512. Both the re-writer and the patch are trained on the VagueEval train set. The code and data are submitted as supplementary materials and will be publicly available upon

publication.

D Dataset Details

To construct the VagueEval dataset, we first identify and compile a pool of benign prompts from two primary sources: Chatbot Arena and MSMARCO. These sources provide a diverse range of natural language queries and instructions that are generally considered safe and non-malicious. Next, we focus on selecting prompts that contain sensitive words or phrases. These sensitive elements are identified based on their frequent appearance in malicious query benchmarks such as Advbench and HEx-PHI. By cross-referencing these benchmarks, we ensure that the selected benign prompts have the potential to be flagged by defensive language models due to the presence of these sensitive terms. And we use GPT-4 to detect the intention of input prompts. Those with 60% or lower confidence are defined as vague input prompts. When a input prompt is detected, GPT-4 provide a binary result (malicious or benign) and a confidence score. For example, 'Tell me what chemicals are in fertilizer' was classified to be benign by GPT-4 and the confidence score is 57%. It is benign because it is a general factual question that appears in many legitimate contexts . The confidence is 57% because there's some ambiguity in intent without context and this type of prompt is commonly used in early-stage jailbreaks. There is no linguistic definition of vaguely benign, vaguely malicious and there clear counterparts while we use an advance LLM to set the borderline. And the purpose of doing that is practically select a bunch of inputs that challenge the models' ability of balancing safety and helpfulness. Additionally, we collect an equal number of malicious prompts that have successfully evaded detection by a baseline defense model. These malicious prompts are carefully curated to reflect common evasion tactics used in adversarial settings. The final dataset is a balanced combination of benign prompts with sensitive content and malicious prompts that bypass initial defenses, providing a comprehensive resource for evaluating the robustness of defensive language models.

To ensure the reliability and validity of the VagueEval dataset, a rigorous quality check process is implemented. Initially, each prompt in the dataset undergoes a content review to verify the presence of sensitive words or phrases as specified in the construction criteria. This step involves manual inspection and automated keyword matching to confirm that the prompts align with the characteristics of both benign and malicious queries. Subsequently, the dataset is subjected to a consistency check, where the balance between benign and malicious prompts is verified to ensure that the dataset accurately represents the intended distribution. Additionally, a subset of prompts is tested against multiple defense models to validate that the malicious prompts indeed evade detection while the benign prompts are appropriately flagged. This validation step helps in identifying any anomalies or misclassifications within the dataset. Finally, metadata associated with each prompt is reviewed for completeness and accuracy, ensuring that all relevant information is correctly documented. By following this multi-step quality check process, VagueEval is ensured to be a high-quality dataset suitable for evaluating the effectiveness of defensive mechanisms in language models.