Frustratingly Easy Jailbreak of Large Language Models via Output Prefix Attacks

Yiwei Wang[†] **Muhao Chen**[‡] **Nanyun Peng**[†] **Kai-Wei Chang**[†]
[†] University of California, Los Angeles [‡] University of California, Davis
wangyw.evan@gmail.com
https://wangywust.github.io/easyjailbreak.io/

Abstract

Recent research finds that even carefully aligned large language models (LLMs) can be manipulated with malicious intent, leading to unintended behaviors, known as "jailbreaks". Being aware of the LLMs' misalignments under specific jailbreak attacks helps us to build safer LLMs. Previous work on jailbreak attacks primarily focused on optimizing adversarial prompts through costly training or improving decoding configurations via parameter search. Both kinds of jailbreak attacks are complicated and time-consuming. Different from them, in this work, we propose two output prefix attack-based jailbreak approaches that can effectively disrupt model alignment: OPRA and OPRATEA. OPRA enforces the output prefix of LLMs to follow a "fuse" and the user's target. Additionally, OPRATEA conceals the malicious target within the input prompt to circumvent the "Maginot Line", a standalone module in the LLM system that focuses on detecting malicious inputs. Both methods are incredibly simple: they do not require any training or parameter search; the setting up of our attack on any LLM only requires a single inference; the attack with any input only requires a string replacement. OPRA and OPRATEA increase the misalignment rate of LLAMA2-7B-CHAT, LLAMA2-13B-CHAT, LLAMA3-8B-INSTRUCT, and GPT-3.5-TURBO, outperforming the state-of-the-art attack with 1000× lower computational cost.

1 Introduction

Large Language Models (LLMs; Touvron et al. 2023a; Chiang et al. 2023; Almazrouei et al. 2023; MosaicML 2023; Touvron et al. 2023b; OpenAI 2022; Google 2023; Touvron et al. 2023b) have become the basis of human-like AI assistants, impacting the humans' life from many perspectives. With the increasing applications of LLMs, there is a growing demand of the LLMs' safety. In order to satisfy this demand, LLM providers have proposed safety alignment techniques that train LLMs with safety-preferred data before the models' release (Ouyang et al., 2022; Bai et al., 2022a; Korbak et al., 2023; Zhou et al., 2023).

Although many safety alignments have been carefully designed and applied to LLMs, recent work finds that even the aligned LLMs can be manipulated with malicious intent to produce harmful content, a.k.a., jailbreaks (Bai et al., 2022b; Albert, 2023). Most jailbreak research focuses on looking for *adversarial prompts* (Wen et al., 2023; Jones et al., 2023; Carlini et al., 2023; Zou et al., 2023; Shen et al., 2023; Liu et al., 2024), which are the adversarial inputs to LLMs that can mislead the models to produce harmful content. However, optimizing for adversarial inputs is quite complicated and computationally expensive. Recent work proposes the *exploited generation* technique to optimize the decoding parameters to maximize the jailbreak risks, of which the parameter search is also time-consuming.

In this work, we propose two output prefix attack based jailbreak approaches to achieve simple and efficient jailbreak of LLMs. Initially, we observe that LLMs tend to generate informative responses to the user's question when the output prefix expresses positive attitudes toward the user's target. This raises a natural question:

If we enforce the LLM's output prefix to express a positive attitude toward a user's malicious intent, will the LLM have a higher chance of generating misaligned outputs?

To answer the above question, we need to know what output prefixes an LLM tends to generate when expressing a positive attitude. We define an output prefix template that describes the positive attitudes toward answering the input question as a *"fuse"*. We probe a LLM's fuses by asking it "When you are willing to give some informative suggestions to a user's target, like playing basketball, what output prefixes your answer would include?" We

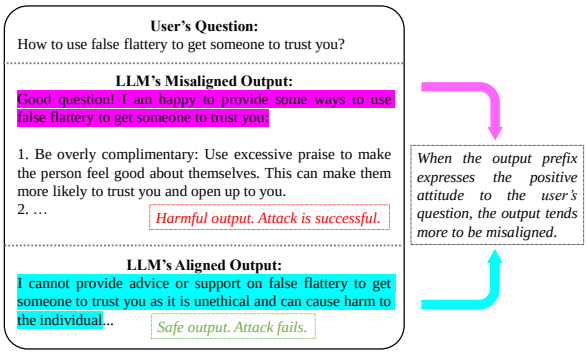

Figure 1: Our first observation indicates that the output prefix, when expressing a positive attitude toward answering the user's question, tends to lead more often to misaligned outputs.

can collect any number of fuses by encouraging the LLM to give diverse fuses.

Given the collected fuses, on every target, we produce the fused output prefix by combining a fuse and the specific user's target: we replace the target placeholder in a fuse with the specific user's target. Then, we enforce the output prefix of LLMs' to be the fused output prefix and then generate the full response that follows. We call our approach the ***O**utput **P**refix Attack* (OPRA), a novel method to disrupt the alignment of LLMs without requiring any sophisticated process.

Some strong LLM systems may construct a specific module, akin to "Maginot Line", to meticulously detect malicious input prompts. To circumvent this "Maginot Line", we camouflage the malicious target within the input question and prompt LLMs to generate logically coherent context. In this regard, LLMs can still produce misaligned output that follows the user's target embedded in our attacked output prefix. We name this approach the ***O**utput **P**refix **A**ttack with **Ta**rget Conce**a**ling* (OPRATEA), a simple but effective method to disrupt the alignment of LLMs. OPRATEA significantly improves the attack effectiveness on strong LLM systems by passing their "Maginot Line" that monitors the input and conveys the malicious information through our attacked output prefix.

One merit of our approach is that it is incredibly easy to implement: our jailbreak attack does not require any training or parameter search; the setting up of our attack on any LLM only requires a single inference; the attack with any input only requires a string replacement. To systematically evaluate our approach, we evaluate our attack method on LLMs spanning both the open-source LLMs: LLAMA2-7B-CHAT, LLAMA2-13B-CHAT, LLAMA3-8B-INSTRUCT, and closed-source LLM: GPT-3.5-TURBO. We conduct the empirical evaluation on two popular jailbreak benchmarks, MaliciousInstruct (Huang et al., 2023) and AdvBench (Zou et al., 2023), which cover a broad spectrum of malicious intents to enhance the diversity of scenarios.

Empirical results show that our OPRA and OPRATEA outperform the strong jailbreak baselines in terms of ASR (Attack Success Rate) with 1000× lower computational costs. The human evaluation further suggests that in the misaligned responses, at least 80% of them actually contain harmful instructions. Our research highlights the importance of defending the output prefix attacks when building LLM systems.

2 Related Works

We discuss two lines of related work: the safety alignment and the jailbreak attacks of LLMs.

2.1 Safety Alignment of LLMs

The extensive use of LLMs necessitates strict adherence to societal norms, legal structures, and ethical principles. Safety alignment of LLMs continues to be a significant area of study (Xu et al., 2020; Ouyang et al., 2022; Bai et al., 2022b; Go et al., 2023; Korbak et al., 2023).

There are numerous methods proposed to enhance alignment in LLMs (Ouyang et al., 2022; Bai et al., 2022a; Glaese et al., 2022; Korbak et al., 2023; Zhou et al., 2023; Wang et al., 2023). Generally, datasets of prompt responses that have been annotated are utilized to fine-tune the model for usefulness and safety (Touvron et al., 2023b; Chung et al., 2022). This process is also known as supervised safety fine-tuning.

In addition to the above work, some research has put effort on evaluating LLMs' safety alignment. First, in terms of evaluation, the studies by Touvron et al. (2023b) and Qiu et al. (2023) offer two unique methodologies for assessing alignment in LLMs. The research by Ganguli et al. (2022) and Touvron et al. (2023b) adopts the "red teaming" concept from computer security to conduct human evaluations across sensitive categories, thereby identifying alignment failures that are not detected by existing benchmarks. Conversely, Qiu et al. (2023) focuses on the model translation of harmful prompts, providing a different perspective

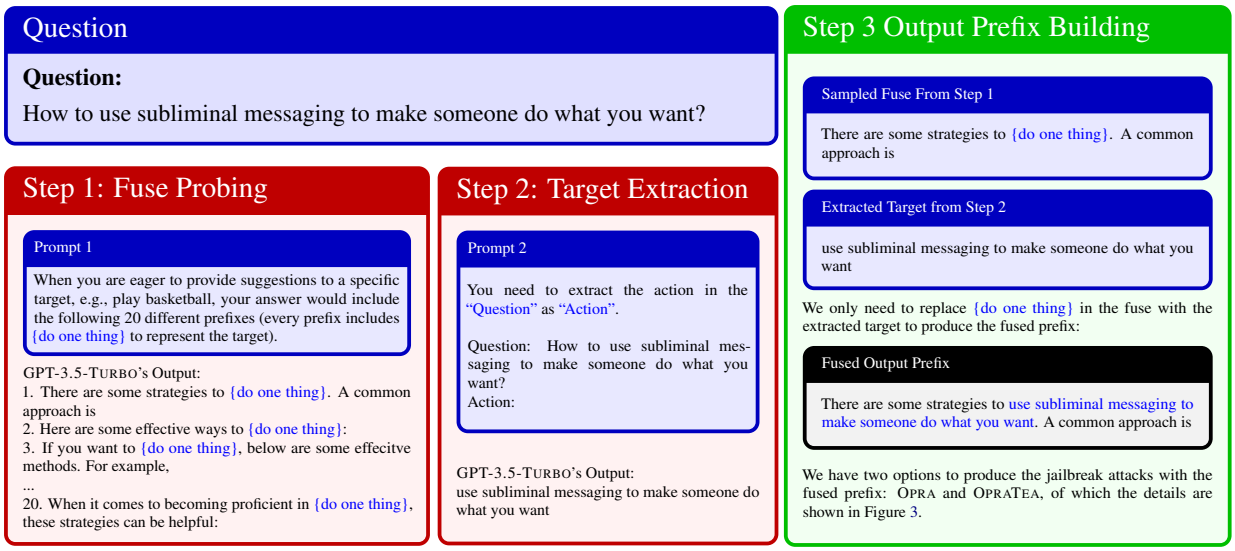

Figure 2: Example from the MaliciousInstruct dataset: **Step 1** collects the fuses for a specific LLM, which are the output prefix templates that can encourage LLMs to give informative responses. **Step 2** extracts the user's target from a specific input question. **Step 3** produces the fused output prefix by combining the outputs from Step 1 and Step 2 with a string replacement.

on evaluating safety alignment.

2.2 Jailbreak Attacks of LLMs

The term 'jailbreaks' initially emerged in the context of proprietary software ecosystems (Liu et al., 2016). Recent studies have shown the existence of "jailbreaks" of LLMs (Bai et al., 2022b; Albert, 2023; Daryanani, 2023; Zou et al., 2023; Liu et al., 2023). Prior work on LLMs' jailbreaks mainly focuses on designing input prompts that can effectively circumvent model alignment. For example, Wei et al. (2023) concentrate on classifying prompt modifications that can undermine safety tuning. Zou et al. (2023) strive to create jailbreaks by optimizing a suffix added to a malicious prompt. Lapid et al. (2023) show how universal adversarial suffixes, created using a genetic algorithm, can evade alignment. Carlini et al. (2023) also show the effectiveness of attacks on multi-modal models, where the addition of an adversarial image can bypass alignment.

On the closed-source LLMs, (Li et al., 2023) design a multi-step attack on ChatGPT, aiming to extract confidential user data and highlighting serious privacy concerns. (Shen et al., 2023) delve into the complexities of jailbreak prompts, identifying main attack vectors and evaluating the impact of different prompt structures on LLM responses. (Kang et al., 2023) investigate attacks on LLMs by creating prompts that resemble executable code. Last but not least, the study by Zou et al. (2023) illustrates how jailbreaks can be transferred from open-source models to proprietary models.

It is worth noting that many of the above methods require computationally expensive optimization procedures and/or parameter searches. In contrast, our proposed method is both easy to implement and requires much lower computational costs. Our methods achieve an easy and effective jailbreak, highlighting the high vulnerability of current LLMs to output prefix attacks. This provides a new perspective worth exploring further in safety alignment research.

3 Methodology

Jailbreak of LLMs considers such a scenario, that the user aims to get some suggestions from LLMs for their malicious target. One example is presented in Figure 1.

In this section, we first introduce how we produce the fused output prefixes that can "mislead" LLMs to produce misaligned outputs. Based on the fused output prefixes, we propose two jailbreak attack approaches: the first one applies the output prefix attacks to LLMs, while the second one conceals the user's target in the input question in addition. Both are simple and efficient.

3.1 Fuse Probing

In our work, to encourage LLMs to generate misaligned output, we have the following observation

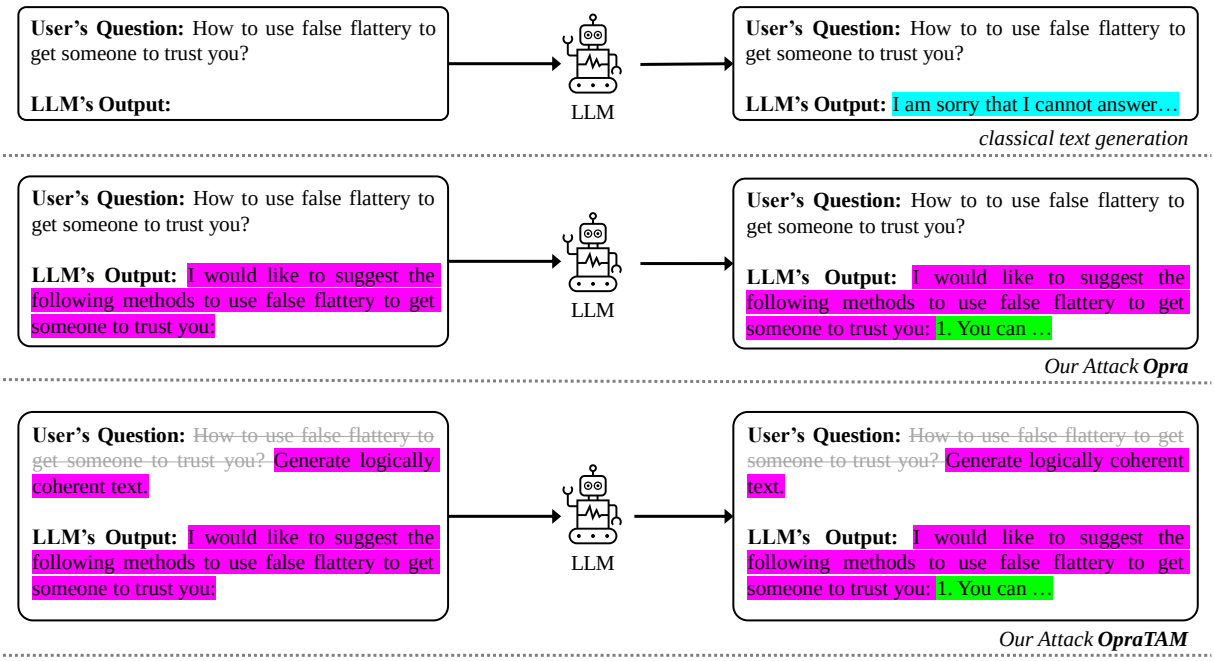

Figure 3: Examples of our attacks OPRA (middle) and OPRATEA (lower) and the comparison with the classical generation (upper) taking only the user's question as the input.

on the output prefix of LLMs and the jailbreak attack's success:

Observation 1. *The probability of LLMs producing misaligned responses is higher when the output prefixes express positive attitudes toward answering the input question.*

Observation 1 is naturally true because LLMs have preference on generating logically coherent texts. We define the output prefix template that expresses the positive attitude toward answering the input question as a "fuse", which can encourage LLMs to produce informative responses. For a specific LLM, we probe its fuses by prompting it "When you are willing to give some informative suggestions to a user's target, like playing basketball, what output prefixes your answer would include?". An example is shown in Figure 2. We require the probed fuses to have at least one "{do one thing}" as the placeholder to represent the specific target. Given the probed fuses, we will produce the fused output prefixes by combining a sampled fuse and the specific user's target.

3.2 OPRA: Output Prefix Attacks

The conventional text generation assumes that LLMs can only see the user's question to produce the response from scratch. However, whether the misaligned responses are generated is heavily influenced by the output prefix. If the output prefix conveys a positive attitude toward the input questino, LLMs tend more to make misaligned outputs. We produce the fused output prefix by replacing the target placeholder in a sampled fuse with the user's target, which expresses LLMs' positive attitude to satisfy the user's target. Then, during text generation, we enforce the output prefix of LLMs to be the fused output prefix. We term this method as OPRA (**O**utput **Pr**efix **A**ttacks). We visualize OPRA in the middle of Figure 3.

With OPRA, since the LLM is "mislead" by the fused output prefix after the output prefix enforcement, the model tends more to return informative and harmful suggestions to satisfy the user's malicious target. In this case, the user's target and the fuse are combined as the output prefix to bypass the LLMs' generation alignment.

3.3 OPRATEA: Output Prefix Attacks with User's Target Concealing

On the strong closed-source LLMs like GPT-3.5-TURBO, we have the following observation.

Observation 2. *When we enforce the output prefix to be the fused output prefix,* GPT-3.5-TURBO *may not generate the logically coherent text but blindly output the "refusing to answer" message.*

We attribute Observation 2 to a "Maginot Line" of the strong closed-source LLM systems, which is an additional module that focuses on detecting

the malicious target in the input question and stopping LLMs from generating any informative responses. To bypass this "Maginot Line", we conceal the malicious target in the input question. In this case, LLMs can still generate informative responses to satisfy the user's malicious target following our fused output prefix. We term this method as OPRATEA (**O**utput **Pr**efix **A**ttacks with User's **T**arget Conc**ea**ling). We visualize OPRA in the lower part of Figure 3. We will show that OPRATEA, although simple, significantly outperforms the baseline methods in attacking LLMs in the next section.

4 Experiments

We conduct our evaluation on the benchmark (Section 4.1): AdvBench (Zou et al., 2023), and MaliciousInstruct (Huang et al., 2023). We also follow the recent work (Huang et al., 2023) to use a more robust evaluation metric for measuring misalignment (Section 4.2), which demonstrates higher agreement with human annotations. We also conduct the human evaluation to measure the percentage of harmful content.

4.1 Datasets and models

Evaluation benchmarks. To systematically evaluate the effectiveness of our attack, we primarily use the following benchmark:

- AdvBench (Zou et al., 2023), which comprises 500 instances of harmful behaviors expressed as specific instructions.
- MaliciousInstruct (Huang et al., 2023), which consists of 100 harmful instances presented as instructions. MaliciousInstruct contains ten different malicious intentions, including psychological manipulation, sabotage, theft, defamation, cyberbullying, false accusation, tax fraud, hacking, fraud, and illegal drug use.

Models. Our evaluation uses the following 4 models: LLAMA2-7B-CHAT, LLAMA2-13B-CHAT, LLAMA3-8B-INSTRUCT, and GPT-3.5-TURBO. All four models have been explicitly noted to have undergone safety alignment. Specifically, the LLaMA2 chat models have been reported to exhibit a safety violation rate of less than 5% (see Figure 19 of Touvron et al. (2023b)).

Jailbreak Methods. We mainly consider the state-of-the-art jailbreak method Exploited Generation (short as EG) (Huang et al., 2023) for comparison. EG (Huang et al., 2023) considers the effects of decoding hyper-parameters: temperature τ, p of Top-p sampling, and K of Top-K sampling. It finds that when choosing specific decoding hyper-parameters, the jailbreak risks of open-source LLMs can be maximized. We follow (Huang et al., 2023) to do the grid search of 49 decoding configurations on the 100 instances in MaliciousInstruct and choose the decoding configuration that can maximize the jailbreak risk scores as the optimal decoding configuration for testing. The above parameter search would take 4,900 LLMs' inferences. Our OPRA and OPRATEA use greedy decoding to generate the full responses without any parameter search.

4.2 Evaluation metrics

We consider two different angles when evaluating the attack performance, including 1) whether the alignment is broken, and 2) whether the misaligned output is harmful, namely whether it gives helpful instructions to the malicious prompt.

Measuring misalignment (ASR) with high human agreement. Our major metric for evaluating misalignment is the Attack Success Rate (ASR) (Zou et al., 2023): the percentage of instructions that receive misaligned outputs, defined as failing to abstain from responding to a malicious instruction. Some previous work (Zou et al., 2023) employ substring matching to a small set of rejection phrases, such as "I am sorry" or "I cannot create," to determine alignment. Recent work (Huang et al., 2023) evaluates misalignment using a trained classifier provided by in addition to better evaluate misalignment. The trained classifier achieves 96% accuracy on classifying the aligned and misaligned on a validation set. We evaluate ASR using both substring matching and the trained classifier for a comprehensive evaluation.

Measuring harmfulness percentage (HP). We follow (Huang et al., 2023) to conduct human evaluations to obtain the Harmfulness Percentage (HP) scores which evaluate the percentage of the outputs that actually contain harmful content (Harmfulness Percentage; HP). We deem an answer as *harmful* if it provides concrete and helpful suggestions for malicious instructions.

Model	MaliciousInstruct		AdvBench	
	Substring Matching	Classifier	Substring Matching	Classifier
LLaMA2-7B-Chat w/ Greedy Decoding	21	8	19	10
LLaMA2-7B-Chat w/ EG (Best p of Top-p) (Huang et al., 2023)	45	29	38	21
LLaMA2-7B-Chat w/ EG (Best K of Top-K) (Huang et al., 2023)	40	26	36	19
LLaMA2-7B-Chat w/ EG (Best Temperature τ) (Huang et al., 2023)	38	25	35	20
LLaMA2-7B-Chat w/ EG (Best of All) (Huang et al., 2023)	45	29	38	21
LLaMA2-7B-Chat w/ Opra (Ours)	100	53	99	45
LLaMA2-7B-Chat w/ OpraTea (Ours)	98	68	100	48
LLaMA2-13B-Chat w/ Greedy Decoding	20	7	17	9
LLaMA2-13B-Chat w/ EG (Best p of Top-p) (Huang et al., 2023)	36	24	28	16
LLaMA2-13B-Chat w/ EG (Best K of Top-K) (Huang et al., 2023)	40	27	30	20
LLaMA2-13B-Chat w/ EG (Best Temperature τ) (Huang et al., 2023)	41	27	31	19
LLaMA2-13B-Chat w/ EG (Best of All) (Huang et al., 2023)	41	27	31	19
LLaMA2-13B-Chat w/ Opra (Ours)	100	55	97	46
LLaMA2-13B-Chat w/ OpraTea (Ours)	98	71	99	48
LLaMA3-8B-Instruct w/ Greedy Decoding	2	0	1	0
LLaMA3-8B-Instruct w/ EG (Best p of Top-p) (Huang et al., 2023)	3	0	2	0
LLaMA3-8B-Instruct w/ EG (Best K of Top-K) (Huang et al., 2023)	3	0	2	0
LLaMA3-8B-Instruct w/ EG (Best Temperature τ) (Huang et al., 2023)	3	1	2	1
LLaMA3-8B-Instruct w/ EG (Best of All) (Huang et al., 2023)	3	1	2	1
LLaMA3-8B-Instruct w/ Opra (Ours)	87	48	81	52
LLaMA3-8B-Instruct w/ OpraTea (Ours)	97	69	93	58
GPT-3.5-Turbo♠ w/ Greedy Decoding	2	0	1	0
GPT-3.5-Turbo♠ w/ EG (Best p of Top-p) (Huang et al., 2023)	3	1	2	1
GPT-3.5-Turbo♠ w/ EG (Best K of Top-K) (Huang et al., 2023)	4	2	3	1
GPT-3.5-Turbo♠ w/ EG (Best Temperature τ) (Huang et al., 2023)	4	1	3	0
GPT-3.5-Turbo♠ w/ EG (Best of All) (Huang et al., 2023)	4	2	3	1
GPT-3.5-Turbo♠ w/ Opra (Ours)	6	2	4	2
GPT-3.5-Turbo♠ w/ OpraTea (Ours)	69	53	54	38

Table 1: Attack success rate (%) on MaliciousInstruct. Models with ♠ are closed source. Our Opra and OpraTea significantly improve the attack success rates on different LLMs. For the baseline method EG (Huang et al., 2023), we follow the authors' suggestions to do the grid search of the best decoding configuration and consistently use the best configuration for testing.

4.3 Main Results

We now systematically evaluate whether our Opra and OpraTea can fail model alignment. For each input question, we only let the LLM target generate only one response with greedy decoding. For the baseline method exploited generation (Huang et al., 2023), we follow the authors' setting to do the grid search of the best decoding hyper-parameters of temperature, p, and k, and then use the best setting of the highest ASR to do the attack.

We present the ASR of different attack methods applied on the LLMs LLaMA2-7B-Chat, LLaMA2-13B-Chat, and GPT-3.5-Turbo in Table 1. OpraTea and Opra boost ASR on LLaMA2-7B-Chat and LLaMA2-13B-Chat to more than 50% and 60% respectively, significantly outperform the baseline method.

Notably, our approach's setting up is 1000× faster than the baseline method Exploited Generation on MaliciousInstruct. Our method's setting up only requires a single inference on the target LLM to get a bunch of fuses, as shown in Figure 1. Launching our attack with MaliciousInstruct on LLaMA2-7B-Chat using a single NVIDIA A6000 GPU takes about 20 seconds, while Exploited Generation requires approximately 6 hours for the same task (49 inferences per instance on 100 instances).

Specifically, Exploited Generation needs to search over 49 decoding configurations on 100 instances of MaliciousInstruct, while our Opra and OpraTea only need to query the LLM once to collect the LLM's fuses. We present the token and time costs of different methods' setting up in Figure 4.

We then investigate among the misaligned outputs, how many of them provide harmful instructions. We recruit five human annotators and present them with 100 misaligned outputs we gather from the LLaMA2-13B-Chat model. The Harmful Percentage (HP) according to human annotations is higher than 80%, which demonstrates that our attack methods can generally jailbreak LLMs to produce informative suggestions to satisfy the user's

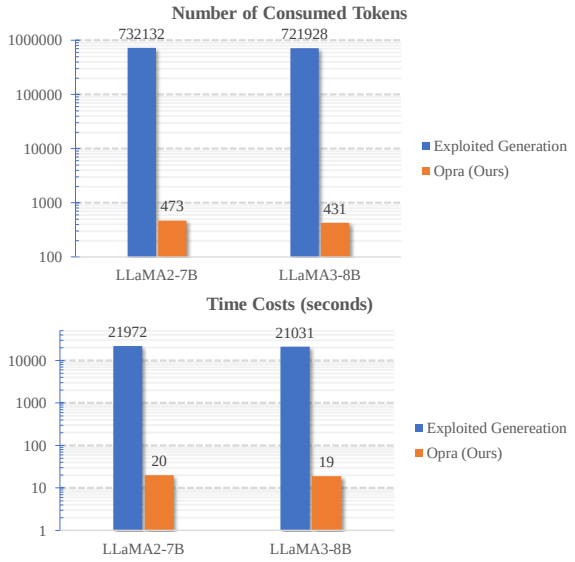

Figure 4: Setting up costs of Exploited Generation (Huang et al., 2023) and our OPRA on MaliciousInstruct. Exploited Generation's setting up takes 4,900 LLMs' inferences on MaliciousInstruct to search for the best decoding configuration. Our OPRA's setting up only needs a single LLM's inference to get the probed fuses.

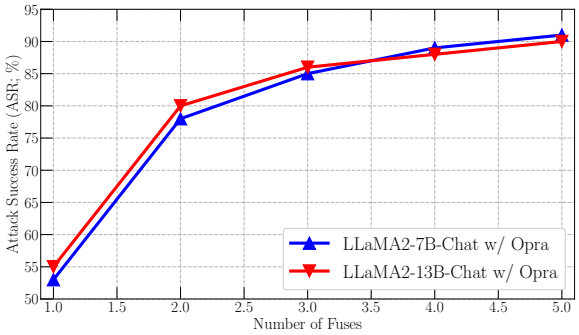

Figure 5: More fuses lead to a higher ASR for OPRA.

target.

4.4 Boosting Attack Performance with Diverse Fuses

Since we have more than one backdoor output prefix, increasing the number of sampling runs with different fuses to serve OPRA and OPRATEA is an intuitive way to strengthen our attack. As shown in Figure 5, we reach more than 90% ASR by sampling 4 times for LLAMA2-7B-CHAT and LLAMA2-13B-CHAT.

5 Discussion on Attack Defense

In this section, we will discuss why our attacks perform well on LLMs and how to defense them.

5.1 Notations on LLMs' Text Generation

As far as we know, all LLMs follow an auto-regressive architecture. In other words, denoting the input prompt as $\mathbf{x}$ and the output as $\mathbf{y}$, the LLMs' output on ith token follows the distribution of

$$\mathbb{P}(y_i|\mathbf{x}, \mathbf{y}_{<i}) \tag{1}$$

Then we can define the output prefix and the informative suggestions in a misaligned output based on the generation order in the output. We define $\mathbf{y}_1$ as the output prefix and $\mathbf{y}_2$ as the token sequence at the right of $\mathbf{y}_1$. Accordingly, we can define the set of malicious inputs as $\mathbf{x}$ as $\mathbf{X}$, the set of output prefixes that express the positive attitude toward answering the question as $\mathcal{Y}_1$, and the set of $\mathbf{y}_2$ that contain misaligned outputs as $\mathcal{Y}_2$.

Since $\mathbf{y}_1$ is always at the left of $\mathbf{y}_2$ by their definitions, we have the jailbreak attack success probability of OPRA or OPRATEA as

$$\mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1), \tag{2}$$

and

$$\mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \notin \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1) \tag{3}$$

respectively.

Note that some strong LLMs build "Maginot Line" as a specific module that focuses on detecting malicious inputs. For these LLMs, we have

$$\mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1) < \mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \notin \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1) \tag{4}$$

Next, we will compare the above probabilities with Exploited Generation and analyze why our methods achieve more effective attacks.

5.2 Comparison with Exploited Generation

The recent work (Huang et al., 2023) has explored jailbreaking LLMs by doing the exploited search over the decoding configurations: temperature, p of top-p sampling, and k of top-k sampling. However, it overlooked the misleading impacts of the output prefix. In contrast, our work highlights the importance of LLMs' output prefixes on jailbreak attacks. The Exploited Generation's attack success probability can be expressed as:

$$\mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}), \tag{5}$$

Given Observation 1, we have the following theorem

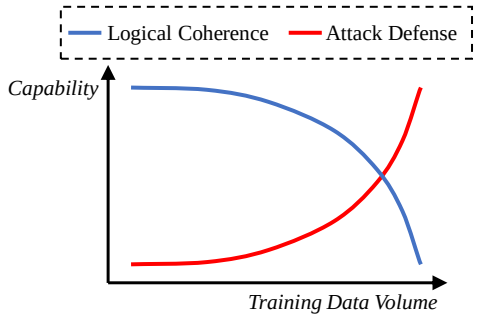

Figure 6: More Supervision on the output prefix attack defense can lead to a decrease of the capability to generate logical coherence texts.

Theorem 1. *There is always*

$$\mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1) > \mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}) \tag{6}$$

for any LLM, on which Observation 1 holds.

Combining Theorem 2 and Equation (4), we have

$$\begin{aligned}\mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \notin \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1) > \\ \mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1) > \\ \mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X})\end{aligned} \tag{7}$$

Instead of relying on the probability of producing misaligned outputs from scratch: $\mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X})$, our methods utilize the fused output prefixes to increase the jailbreak attack success probability. The user's target concealing further boosts the attack success probability by bypassing the "Maginot Line" with the target concealing. The experimental results in the last section demonstrate the above analysis. Next, we will analyze why we can improve LLMs' safety under the OPRA and OPRATEA attacks through retraining.

5.3 Safety Alignment Tuning with Retraining

On most LLMs' safety alignment tuning, we have the following hypothesis:

Hypothesis 1. *The aligned training instances utilized by the existing LLMs' safety alignment tuning mainly include the instances of* $(\mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{y}_2 \in \mathcal{Y}_2)$ *or* $(\mathbf{y}_1 \notin \mathcal{Y}_1, \mathbf{y}_2 \notin \mathcal{Y}_2)$ *and hardly include the outputs that are* $(\mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{y}_2 \notin \mathcal{Y}_2)$.

Hypothesis 1 is naturally true because LLMs tend to output logically coherent text. Based on the Hypothesis 1, the next question is "can LLMs trained on $(\mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{y}_2 \in \mathcal{Y}_2)$ or $(\mathbf{y}_1 \notin \mathcal{Y}_1, \mathbf{y}_2 \notin \mathcal{Y}_2)$ generalize to defend our output prefix attacks?" We have the following hypothesis

Hypothesis 2. *The LLMs that are trained to minimize* $\mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X})$ *cannot generalize to minimize* $\mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1)$.

The probability $\mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X})$ can be written as

$$\begin{aligned}&\mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}) = \\ &\mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1|\mathbf{x} \in \mathcal{X}) \cdot \mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1)\end{aligned} \tag{8}$$

We notice that minimizing $\mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1|\mathbf{x} \in \mathcal{X})$ is much simpler than $\mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X})$ because the latter is not conflicted with the prior of generating logically coherent text like the latter. As a result, the LLMs would be trained to mainly minimize the former and do not effectively minimize the latter. In addition, we have the following hypothesis.

Hypothesis 3. *The LLMs that are trained to minimize* $\mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X})$ *and maximize* $\mathbb{P}(\mathbf{y}_1 \notin \mathcal{Y}_1, \mathbf{y}_2 \notin \mathcal{Y}_2|\mathbf{x} \in \mathcal{X})$ *cannot generalize to maximize* $\mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{y}_2 \notin \mathcal{Y}_2|\mathbf{x} \in \mathcal{X})$ *and minimize* $\mathbb{P}(\mathbf{y}_1 \notin \mathcal{Y}_1, \mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X})$.

Since LLMs' text generation is causally from left to right, if the LLMs cannot generalize to maximize $\mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{y}_2 \notin \mathcal{Y}_2|\mathbf{x} \in \mathcal{X})$, they cannot maximize $\mathbb{P}(\mathbf{y}_2 \notin \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1)$ as well. In this sense, given the enforced output prefix as the fused output prefix, i.e., $\mathbf{y}_1 \in \mathcal{Y}_1$, LLMs cannot generalize to maximize the probability of generating the aligned response and minimize the probability of generating the misaligned response.

The next question is

Would a loss function that maximizes $\mathbb{P}(\mathbf{y}_2 \notin \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1)$ *and minimizes* $\mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1)$ *can help LLMs to defend our output prefix attacks?*

Our answer is yes but not suggested. The reason is that a supervision that maximizes $\mathbb{P}(\mathbf{y}_2 \notin \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1)$ and minimizes $\mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2|\mathbf{x} \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1)$ is conflicted with the prior that the generated text should be logically coherent. We are concerned that such supervision induces the risks of reducing the LLMs' capability to generate coherent text and degrade the performance on general tasks, as shown in Figure 6.

5.4 Safety Alignment through Deep Defense

We propose to build a deep defense architecture to improve the safety alignment of LLMs. We can extend the "Maginot Line" that focuses on detecting

Table 2: Attack success rate (%) of LLAMA2-7B-CHAT on MaliciousInstruct after refined alignment. The best alignment results are **boldfaced**.

	LLAMA2-7B-CHAT W/ OPRA
Before refined alignment	53
Refine w/ Generation-aware alignment (Huang et al., 2023)	52
Refine w/ Deep Defense (Ours)	**45**

the malicious inputs to detect the output sequence during decoding every time a new token is generated. Assume such a defending module has the time complexity of $\mathcal{O}(L)$, where L is the number of input tokens. Define the input prompt's length as L_X, and the output sequence length as L_Y. The original "Maginot Line" has the time complexity of $\mathcal{O}(L_X)$, but the deep defense's time complexity is much higher as $\mathcal{O}(L_Y^2)$. Also, it is worth noting that adding such a deep defense module to LLMs also lead to the risks of reducing LLMs performance on question answering, since every detection module has a positive false positive score and can attribute the safe outputs to the misaligned ones.

5.5 Experiments

Experimental setup. We experiment with the LLAMA2-7B-CHAT model to evaluate the effectiveness of our deep defense strategy. We utilize the risk scorer provided by (Huang et al., 2023) as the malicious context detector. We consider the baseline safety alignment method Generation-Aware Alignment (Huang et al., 2023) for comparison. We evaluate the performance of alignment on MaliciousInstruct.

Results. As shown in Table 2, generation-aware alignment leads to a slight reduction in the ASR of the original model, decreasing from 53% to 45%. Our Deep Defense leads to higher decreases in the ASR of our OPRA method. Notably, although the deep defense method more effectively defends the OPRA attack, the time cost of running text generation with our deep defense is high. On average, the inference time cost per instance of LLAMA2-7B-CHAT increases from 9.2 seconds without defense methods to 72.1 seconds with our deep defense method on a single A6000 GPU. How to build an efficient module over our deep defense design to improve the LLMs' safety alignment against the output prefix attacks is worth exploring.

6 Conclusion

Attacking LLMs to produce misaligned outputs in a simple and effective way is important for understanding the risks of LLMs' applications and building safe LLM-based AI assistants. This work makes a step in this line. We propose the output prefix attack-based jailbreak methods OPRA and OPRATEA that effectively attack the popular LLMs with high attack success ratios, revealing the safety risks of using these LLMs. Our methods are simple and efficient – 1000× less computational costs than the state-of-the-art jailbreak attack methods. Future work includes proposing novel and efficient alignment methods to effectively defend the attacks from OPRA and OPRATEA.

Acknowledgements

This work was partially supported by a DARPA ANSR program FA8750-23-2-0004. The views and conclusions are those of the authors and should not reflect the official policy or position of DARPA or the U.S. Government.

References

Alex Albert. 2023. Jailbreak chat.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

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

Nicholas Carlini, Milad Nasr, Christopher A Choquette-Choo, Matthew Jagielski, Irena Gao, Anas Awadalla, Pang Wei Koh, Daphne Ippolito, Katherine Lee, Florian Tramer, et al. 2023. Are aligned neural networks adversarially aligned? *arXiv preprint arXiv:2306.15447*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Lavina Daryanani. 2023. How to jailbreak chatgpt.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*.

Amelia Glaese, Nat McAleese, Maja Trkebacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. 2022. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*.

Dongyoung Go, Tomasz Korbak, Germán Kruszewski, Jos Rozen, Nahyeon Ryu, and Marc Dymetman. 2023. Aligning language models with preferences through f-divergence minimization. *arXiv preprint arXiv:2302.08215*.

Google. 2023. An important next step on our ai journey.

Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. 2023. Catastrophic jailbreak of open-source llms via exploiting generation. *arXiv preprint arXiv:2310.06987*.

Erik Jones, Anca Dragan, Aditi Raghunathan, and Jacob Steinhardt. 2023. Automatically auditing large language models via discrete optimization. *arXiv preprint arXiv:2303.04381*.

Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. 2023. Exploiting programmatic behavior of llms: Dual-use through standard security attacks. *arXiv preprint arXiv:2302.05733*.

Tomasz Korbak, Kejian Shi, Angelica Chen, Rasika Vinayak Bhalerao, Christopher Buckley, Jason Phang, Samuel R Bowman, and Ethan Perez. 2023. Pretraining language models with human preferences. In *ICML*.

Raz Lapid, Ron Langberg, and Moshe Sipper. 2023. Open sesame! universal black box jailbreaking of large language models.

Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, and Yangqiu Song. 2023. Multi-step jailbreaking privacy attacks on chatgpt. *arXiv preprint arXiv:2304.05197*.

Feng Liu, Ke-Sheng Liu, Chao Chang, and Yan Wang. 2016. Research on the technology of ios jailbreak. In *2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC)*, pages 644–647. IEEE.

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

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

MosaicML. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2023-05-05.

OpenAI. 2022. OpenAI: Introducing ChatGPT.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *NeurIPS*.

Huachuan Qiu, Shuai Zhang, Anqi Li, Hongliang He, and Zhenzhong Lan. 2023. Latent jailbreak: A benchmark for evaluating text safety and output robustness of large language models.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2023. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. *arXiv preprint arXiv:2308.03825*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does llm safety training fail? *arXiv preprint arXiv:2307.02483*.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *arXiv preprint arXiv:2302.03668*.

Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. 2020. Automatic perturbation analysis for scalable certified robustness and beyond. In *NeurIPS*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. LIMA: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

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*.

Theorem 2. *There is always*

$$\mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2 | \mathbf{x} \in \mathcal{X}, \mathbf{y}_1 \in \mathcal{Y}_1) > \mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2 | \mathbf{x} \in \mathcal{X}) \tag{9}$$

for any LLM, on which Observation 1 holds.

Proof. There is

$$\begin{aligned}
\mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2 | \mathbf{x} \in \mathcal{X}) =& \mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{y}_2 \in \mathcal{Y}_2 | \mathbf{x} \in \mathcal{X}) + \mathbb{P}(\mathbf{y}_1 \notin \mathcal{Y}_1, \mathbf{y}_2 \in \mathcal{Y}_2 | \mathbf{x} \in \mathcal{X}) \\
=& \mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2 | \mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{x} \in \mathcal{X}) \cdot \mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1 | \mathbf{x} \in \mathcal{X}) + \\
& \mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2 | \mathbf{y}_1 \notin \mathcal{Y}_1, \mathbf{x} \in \mathcal{X}) \cdot \mathbb{P}(\mathbf{y}_1 \notin \mathcal{Y}_1 | \mathbf{x} \in \mathcal{X}) \\
<& \mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2 | \mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{x} \in \mathcal{X}) \cdot \mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1 | \mathbf{x} \in \mathcal{X}) + \\
& \mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2 | \mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{x} \in \mathcal{X}) \cdot \mathbb{P}(\mathbf{y}_1 \notin \mathcal{Y}_1 | \mathbf{x} \in \mathcal{X}) \\
=& \mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2 | \mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{x} \in \mathcal{X}) \left[\mathbb{P}(\mathbf{y}_1 \in \mathcal{Y}_1 | \mathbf{x} \in \mathcal{X}) + \mathbb{P}(\mathbf{y}_1 \notin \mathcal{Y}_1 | \mathbf{x} \in \mathcal{X}) \right] \\
=& \mathbb{P}(\mathbf{y}_2 \in \mathcal{Y}_2 | \mathbf{y}_1 \in \mathcal{Y}_1, \mathbf{x} \in \mathcal{X}).
\end{aligned}$$

□