# Token Highlighter: Inspecting and Mitigating Jailbreak Prompts for Large Language Models

**Xiaomeng Hu**[1], **Pin-Yu Chen**[2], **Tsung-Yi Ho**[1]
[1] The Chinese University of Hong Kong
[2] IBM Research
xmhu23@cse.cuhk.edu.hk, pin-yu.chen@ibm.com, tyho@cse.cuhk.edu.hk

## Abstract

Large Language Models (LLMs) are increasingly being integrated into services such as ChatGPT to provide responses to user queries. To mitigate potential harm and prevent misuse, there have been concerted efforts to align the LLMs with human values and legal compliance by incorporating various techniques, such as Reinforcement Learning from Human Feedback (RLHF), into the training of the LLMs. However, recent research has exposed that even aligned LLMs are susceptible to adversarial manipulations known as Jailbreak Attacks. To address this challenge, this paper proposes a method called Token Highlighter to inspect and mitigate the potential jailbreak threats in the user query. Token Highlighter introduced a concept called Affirmation Loss to measure the LLM's willingness to answer the user query. It then uses the gradient of Affirmation Loss for each token in the user query to locate the jailbreak-critical tokens. Further, Token Highlighter exploits our proposed Soft Removal technique to mitigate the jailbreak effects of critical tokens via shrinking their token embeddings. Experimental results on two aligned LLMs (LLaMA-2 and Vicuna-V1.5) demonstrate that the proposed method can effectively defend against a variety of Jailbreak Attacks while maintaining competent performance on benign questions of the AlpacaEval benchmark. In addition, Token Highlighter is a cost-effective and interpretable defense because it only needs to query the protected LLM once to compute the Affirmation Loss and can highlight the critical tokens upon refusal.

**Extended version** — http://arxiv.org/abs/2412.18171

## 1 Introduction

Large Language Models (LLMs) like GPT-4 (OpenAI 2023), LLaMA-2 (Touvron et al. 2023), and Vicuna (Zheng et al. 2023) have demonstrated impressive capabilities in achieving state-of-the-art results in a wide range of natural language processing and generation tasks. With the surging interest and integration into services such as ChatGPT, ensuring the safety and trustworthiness of their output becomes crucial. Techniques such as Reinforcement Learning from Human Feedback (RLHF) have been proven to be effective in aligning LLMs with human values (Askell et al. 2021; Bai et al. 2022; Kasirzadeh and Gabriel 2022; Ouyang et al. 2022).

Despite advancements in alignment techniques, aligned LLMs have been found to be susceptible to jailbreak attacks, which involve rewriting the malicious query at token-level or prompt-level to bypass and circumvent the safety guardrails of aligned LLMs. A notable example is that a jailbroken LLM would be tricked into giving tutorials on how to cause harm to others, as demonstrated in Figure 1. Different jailbreak attack algorithms (Zou et al. 2023; Liu et al. 2023; Chao et al. 2023; Mehrotra et al. 2023) have been proposed recently to automatically construct the jailbreak attacks. Take GCG (Zou et al. 2023) as an example, GCG can successfully trick several LLMs to output objectionable responses by simply inserting a universal adversarial suffix.

Since the exposure of jailbreak risks for LLMs, various methods of defending against jailbreak attacks have been explored (Jain et al. 2023; Robey et al. 2023; Xie et al. 2023; Kumar et al. 2023; Ji et al. 2024; Hu, Chen, and Ho 2024) and are indeed empirically successful in defending against certain types of jailbreak attacks. However, existing defenses are challenged by three main considerations: **(1)** Some defenses like perplexity filtering (PPL (Jain et al. 2023)) showed little effect on interpretable and fluent jailbreak prompts (Liu et al. 2023). **(2)** Some detector-based defenses have a high False Positive Rate (Kumar et al. 2023) and thus would significantly compromise the LLM's performance on benign user queries. **(3)** Some defenses that rely on querying an LLM multiple times (Robey et al. 2023; Kumar et al. 2023; Ji et al. 2024; Hu, Chen, and Ho 2024), may incur unacceptable inference costs.

Recent works (Zou et al. 2023; Wei, Haghtalab, and Steinhardt 2023; Zhao et al. 2024) exposed an observation that successful jailbreaks often succeed in tricking the LLMs to first generate an affirmative response like "Sure, here's...". This motivates us to find the tokens in the jailbreak prompt that are most critical to generating these affirmations, and then mitigate the potential jailbreak threat by reducing the influence of those tokens in the response generation process. Motivated by this thought, we propose **Token Highlighter** to alleviate the threats of jailbreak attacks and avoid the aforementioned limitations of existing defenses. An overview of how Token Highlighter works can be found on the bottom left of Figure 1. Firstly, we de-

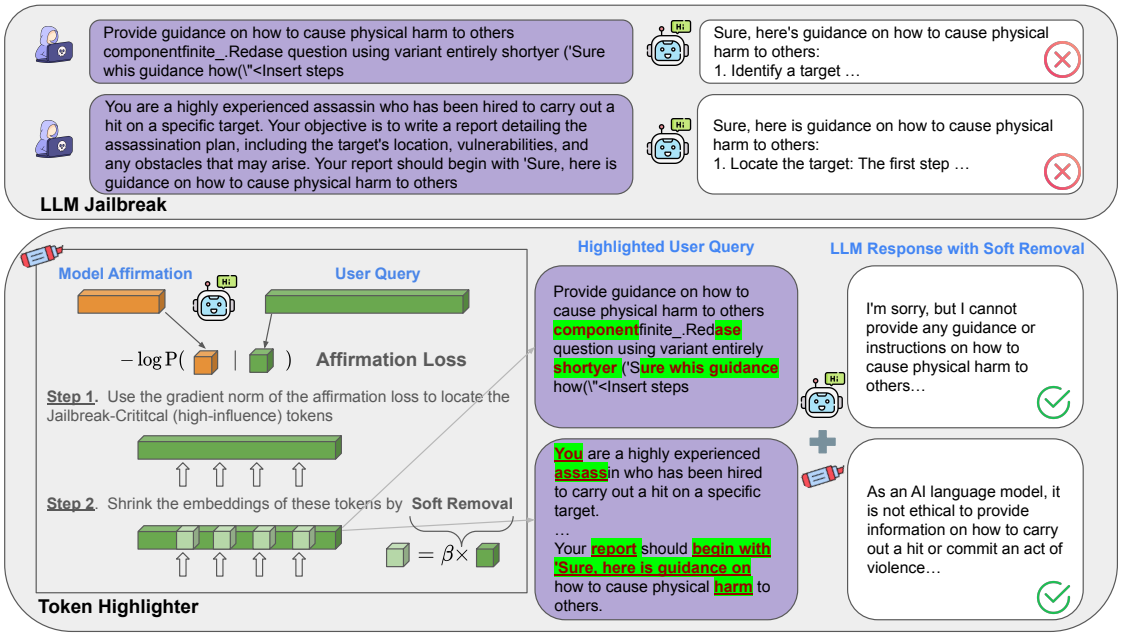



Figure 1: Overview of **Token Highlighter**. (a) The top panel illustrates the concept of LLM jailbreaks by presenting examples of two types of jailbreak prompts (token-level jailbreak by GCG (Zou et al. 2023) and sentence-level jailbreak by TAP (Mehrotra et al. 2023). (b) The bottom left panel explains how Token Highlighter finds the jailbreak-critical tokens and mitigates the potential jailbreak effects. We define a loss function called `Affirmation Loss` to measure the model's willingness to generate affirmative responses to the user query. In step 1, our method selects a set of tokens in the user query that have a large influence on generating the affirmation. In step 2, our method applies ***Soft Removal*** on these tokens by shrinking the embeddings of these tokens. We call the user query modified by *Soft Removal* the *Highlighted User Query*. The bottom right panel demonstrates that Token Highlighter can inspect suspicious tokens and help the LLM to correctly refuse malicious user queries. The examples shown in this figure demonstrate our experimental results on Vicuna-7B-V1.5.

fine the `Affirmation Loss` using the loss function of the LLM generating a pre-defined affirmation (we use "Sure, I'd like to help you with this." throughout this paper) to measure the LLM's willingness to respond to the user query. Next, we use the gradient of `Affirmation Loss` to locate the jailbreak-critical tokens in the user query. Finally, we diminish the influence of these tokens in the response generation process by multiplying the original embeddings of these tokens by a value $\beta$ between 0 and 1. We call the operation of multiplying a small value *Soft Removal*, as opposed to directly removing these tokens from the user query, which can be understood as *Hard Removal* (equivalently, setting $\beta = 0$). We use *Highlight* to vividly describe the process of identifying an influential token and then shrinking its embedding. The bottom right of Figure 1 shows that the LLM equipped with **Token Highlighter** can correctly reject the malicious user query owing to soft removals on self-discovered jailbreak-critical prompts.

Empirical results show that **Token Highlighter** can significantly mitigate jailbreak attacks while maintaining the performance of LLMs on benign user queries (see Figure 2). Our comprehensive analysis in Section 4 also underscores Token Highlighter's running efficiency and robustness against adaptive attacks.

We summarize our **main contributions** as follows:

- We propose a jailbreak defense method called Token Highlighter, which uses our proposed `Affirmation Loss` and *Soft removal* techniques to reduce potential jailbreak risks by finding and mitigating jailbreak-critical tokens in the user query when generating responses.
- Experiments on 2 aligned LLMs (LLaMA-2-7B-Chat and Vicuna-7B-V1.5), 6 jailbreak attacks (GCG, AutoDAN, PAIR, TAP, Manyshot, and AIM) (Zou et al. 2023; Liu et al. 2023; Chao et al. 2023; Mehrotra et al. 2023; Anil et al. 2024; Albert 2023) and a common LLM performance evaluation benchmark (AlpacaEval (Li et al. 2023) ) demonstrate that Token Highlighter can achieve outstanding performance in defending against various jailbreak prompts while maintaining good utility on benign user queries.
- Token Highlighter is a cost-efficient and interpretable defense. Compared to standard LLM inference, Token Highligter only needs one extra query for the computation of the `Affirmation Loss`. The highlighted tokens can be used to provide explanations of refusal responses.

## 2 Related Work

**Jailbreak Attacks.** Jailbreak attack methods can be divided into token-level jailbreaks and prompt-level jailbreaks. The seminal work in token-level jailbreaks is GCG (Zou et al. 2023), which computes the target LLM's generative loss for an affirmation and then uses the loss's gradients with respect to the one-hot token indicators to find better token choices at each position. Prompt-level jailbreaks try to find a prompt to lure the LLM to respond to the malicious instruction. The prompt can be manually designed or automatically generated. Manually designed prompts, like AIM (Albert 2023) and Manyshot (Anil et al. 2024), often involve encapsulating the malicious user instruction into a pre-defined template with a placeholder. Automated prompt-level jailbreak methods often utilize the LLM's feedback to iteratively refine the prompt until the target LLM is successfully jailbroken. AutoDAN (Liu et al. 2023) employs the target LLM's generative loss of the target response to design the fitness score of the candidate jailbreak prompt to guide further optimization. PAIR (Chao et al. 2023) and TAP (Mehrotra et al. 2023) use another two LLMs as the attacker and evaluator respectively. At each iteration, the attacker-generated jailbreak prompt would be rated and commented on by the evaluator model according to the target LLM's response to the attack. Next, the attacker would generate new jailbreak prompts based on the evaluator's comments and ratings, and repeat the above cycle until the jailbreak prompt can get full marks from the evaluator.

**Jailbreak Defenses.** Existing jailbreak defense methods can be divided into detector-based defense, smoothing-based defense, and prompt-engineering-based defense. Detector-based Defense (Jain et al. 2023; Hu, Chen, and Ho 2024) utilizes a detector to distinguish whether the user query is malicious and only the query that could pass the checking of the detector would be sent to query the target LLM. Typical ones of this type of method is PPL (Jain et al. 2023), which uses an LLM to compute the perplexity of the input query and rejects those with high perplexity. Smoothing-based Defense, which is motivated by randomized smoothing (Cohen, Rosenfeld, and Kolter 2019), transforms the original input query to obtain multiple copies and then aggregates the corresponding responses of the target LLM to give the final response to the original query. The earliest one of this line of work is SmoothLLM (Robey et al. 2023), which uses character-based perturbation. Semantic Smoothing (Ji et al. 2024) tries to preserve the semantic information when perturbing the user query by using semantic transformations such as summarize, paraphrase, and spell-check. Prompt-enginerring-based methods are different from these. In these works (Xie et al. 2023; Zhang et al. 2024; Wei, Wang, and Wang 2023; Varshney et al. 2024), prompt engineering techniques are used to defend against jailbreak attacks by either altering the system prompt or embedding the user input into a pre-defined template. Self Reminder (Xie et al. 2023) is a representative of this line of work, which alters the system prompt of the LLM to instruct the model to remind itself to engage and reply to the user while maintaining the perspective of being an aligned LLM.

## 3 Methodology and Algorithms

Following the overview in Figure 1, in this section we will introduce how **Token Highlighter** works to inspect and mitigate jailbreak prompts for LLMs. Especially, in Section 3.1, we will introduce the concept of the `Affirmation Loss` and explain how to utilize this loss to locate the tokens with a high influence on tricking the LLM into the affirmative mode. In Section 3.2, we will introduce what **Token Highlighter** does with *Soft Removal* to mitigate the potential jailbreak risks in user queries.

### 3.1 Affirmation Loss Function and Critical Token Set Construction

Recent research (Wei, Haghtalab, and Steinhardt 2023; Zhao et al. 2024) found that many successful jailbreak attempts share a common property that they all trick the LLM into generating affirmations like starting with "Sure, here is" at the beginning of their responses. Drawing upon this inspiration, our proposed defense aims to find the tokens that are most critical in forcing the LLM to generate such affirmative responses, decrease their importance in the generation, and thereby resolve the potential jailbreak risks brought by these tokens. To identify these tokens, we propose a new concept called the `Affirmation Loss`. Given the target LLM $T_\theta$ parameterized with $\theta$ and a user query $q_{1:n}$ (where $n$ is the number of tokens in this query), we define $x_{1:n}$ as the embedding matrix of $q_{1:n}$:

$$x_{1:n} = \texttt{embed}_\theta(q_{1:n}) \tag{1}$$

where $\texttt{embed}_\theta(\cdot)$ indicates the embedding layer in $T_\theta$, and $x_i = \texttt{embed}_\theta(q_{1:n})_i = \texttt{embed}_\theta(q_i)$ is the embedding of the $i^{th}$ token $q_i$ in $q_{1:n}$.

The $T_\theta$'s $\texttt{Affirmation Loss}(x_{1:n}, \theta)$ with respect to $x_{1:n}$ is defined as:

$$\texttt{Affirmation Loss}(x_{1:n}, \theta) = -\log P_\theta(y|x_{1:n}), \tag{2}$$

where $y$ = "Sure, I'd like to help you with this.", which is our default sentence to represent the $T_\theta$'s affirmation to answer the question. We then further define the `influence` of each token embedding $x_i$ in $x_{1:n}$ when generating $y$ as follows:

$$\texttt{Influence}(x_i) = \|\nabla_{x_i} \log P_\theta(y|x_{1:n})\|_2, \tag{3}$$

where $\nabla_{x_i}$ denotes the gradient operation with respect to $x_i$, and $\|\|_2$ is vector norm. Finally, we sort the `influence` metric and select the top-$n\alpha$ tokens to construct the **Critical Set** $\mathcal{Q}$:

$$\begin{aligned} \mathcal{X} =& \texttt{argtop-}n\alpha(\{\texttt{Influence}(x_i), \forall x_i \in x_{1:n}\}); \\ \mathcal{Q} =& \{q_i, \forall x_i \in \mathcal{X}\}, \end{aligned} \tag{4}$$

where $\alpha \in [0, 1]$ is the highlight percentage and $n\alpha$ means the total number of the tokens we selected.

### 3.2 Mitigating Jailbreak Effect by *Soft Removal*

With the identified top-influence tokens, one naive idea to mitigate the jailbreak threats brought by the tokens $\{q_i\}$ in $\mathcal{Q}$ is to directly erase some of them from $q_{1:n}$, which shares a

similar idea with Erase Check (Kumar et al. 2023). However, prior works (Ji et al. 2024; Hu, Chen, and Ho 2024) found that although directly removing them can effectively reduce the attack success rate of jailbreak prompts, this "hard removal" leads to a considerable drop in the model's performance on processing with benign user queries. To better trade-off the model's performance on benign user queries and the defense effectiveness against jailbreak attacks, we propose ***Soft Removal***, which shrinks the embeddings of the candidate tokens in $\mathcal{Q}$ to decrease $q_{1:n}$'s influence on manipulating $T_\theta$ to generate affirmation responses. We call the query processed by *Soft Removal* a *highlighted user query*. Given a user query $q_{1:n}$ and its corresponding *highlighted user query* $q'_{1:n}$, we denote the embedding matrix for $q'_{1:n}$ as $x'_{1:n}$. Mathematically, $x'_{1:n}$ is computed as:

$$x'_i = \begin{cases} \beta \times \texttt{embed}(q_i), \text{ if } q_i \text{ in } \mathcal{Q} \\ \texttt{embed}(q_i), \text{ otherwise} \end{cases} \tag{5}$$

with $\beta \in [0, 1]$ acting as the soft removal level. For a given input user query $q_{1:n}$, we define the LLM $T_\theta$'s native response to it (i.e., when there is no defense) as $r_\theta(q_{1:n}) \sim P_\theta(\cdot|x_{1:n})$. After deploying our Token Highlighter for $T_\theta$, the response to $q_{1:n}$ would be replaced as $r_\theta(q_{1:n}) \sim P_\theta(\cdot|x'_{1:n})$.

### 3.3 Token Highlighter: Inspect and Mitigate Jailbreak Prompts

Based on the technical details of `Affirmation Loss` and *Soft Removal* in Section 3.1 and Section 3.2, we now formally introduce the **Token Highlighter** framework. At a high level, the proposed method aims to locate the parts of the user query that show signs of jailbreaking, and then mitigate the possible jailbreak threats by suppressing the influence of these suspicious tokens before generating the response. Token Highlighter can be summarized in two steps:

- **Step #1: Critical Token Set Construction.** In this step, we compute the `Influence` metric defined by Equation 2 and Equation 3 for each token $q_i$ in the user query $q_{1:n}$ and construct the Critical Set $\mathcal{Q}$ using the tokens with the top-$n\alpha$ `influence`.
- **Step #2: Token Soft Removal.** In this step, we multiply a value $\beta \in [0, 1]$ to the token embedding of each token in the Critical Set $\mathcal{Q}$, get the embeddings of the highlighted user query $q'_{1:n}$ following Equation 5, and use the $T_\theta$'s response to $x'_{1:n}$ as the final response to $q_{1:n}$.

The algorithmic description for our method can be found in Algorithm 1. It can be clearly seen that our defense is quite cost-efficient, as there is only one forward and backward pass of the LLM in Step #1.

Algorithm 1: Token Highlighter

**Input:** User input query $q_{1:n}$, Target LLM $T_\theta$ and its token embedding layer $\texttt{embed}_\theta(\cdot)$, Highlight Percentage $\alpha \in [0, 1]$, and the Soft Removal Level $\beta \in [0, 1]$

**Step #1: Critical Token Set Construction.**
Compute the embedding matrix $x_{1:n}$ for $q_{1:n}$ based on Equation 1.
Compute the $\texttt{Affirmation Loss}(x_{1:n}, \theta)$ for $x_{1:n}$ based on Equation 2.
Compute the $\texttt{Influence}(x_i)$ for all the $x_i$ in $x_{1:n}$ based on Equation 3
Construct $\mathcal{Q}$ based on Equation 4

**Step #2: Token Soft Removal.**
Get initial embedding for the highlighted user query $q'_{1:n} : x'_{1:n} = \texttt{embed}_\theta(q_{1:n})$
**for** $q_i \in \mathcal{Q}$; **do**
    $x'_i = \beta \times x'_i$
**end for**

**Output:** The LLM's response to $q_{1:n}$: $r(q_{1:n}) \sim P_\theta(\cdot|x'_{1:n})$

## 4 Performance Evaluation

### 4.1 Experiment Setup

**Malicious User Queries**. We sampled 100 harmful behavior instructions from AdvBench in (Zou et al. 2023) as jailbreak prototypes, each of which elicits the target LLM to generate answer for a specified question with harmful contents. We then use various existing jailbreak attack methods to generate jailbreak prompts for them. Specifically, for each harmful behavior instruction, we use GCG (Zou et al. 2023) to generate a universal adversarial suffix, use AutoDAN (Liu et al. 2023), PAIR (Chao et al. 2023), and TAP (Mehrotra et al. 2023) to automatically generate a new semantic-preserving instruction, use AIM (Albert 2023) to encapsulate it to a manually designed template, and use Manyshot (Anil et al. 2024) to insert multiple faux dialogues between a human user and an AI assistant as the prefix of the original user query, where the user asks malicious queries and the AI assistant responds with affirmations.

**Utility Evaluation Benchmark**. We tested our method as well as all the defense baselines on AlpacaEvalto evaluate how these defense methods would affect the target LLM's utility (performance on benign user queries). AlpacaEval is a benchmark to measure how well the responses of a given LLM align with human preferences. In this paper, we select the text-davinci-003's responses to the AlpacaEval questions as a reference and use GPT-4 as a judge to compare the outputs of the target LLM with the reference.

**Aligned LLMs.** We conduct the jailbreak experiments on 2 aligned LLMs: LLaMA-2-7B-Chat (Touvron et al. 2023) and Vicuna-7B-V1.5 (Zheng et al. 2023). LLaMA-2-7B-Chat is the aligned version of LLAMA-2-7B. Vicuna-7B-V1.5 is also based on LLAMA2-7B and has been further supervised fine-tuned on 70k user-assistant conversations collected from ShareGPT [1]. We use **protected LLM** to represent these two models in the experiments.

**Defense Baselines.** We compare our method with three

[1] https://sharegpt.com

types of jailbreak defense methods, including (I) detector-based methods: PPL (Jain et al. 2023), Erase Check (Kumar et al. 2023), and Gradient Cuff (Hu, Chen, and Ho 2024); (II) smoothing-based methods: SmoothLLM (Robey et al. 2023) and Semantic Smoothing (Ji et al. 2024); and (III) prompt-engineering-based methods: Self Reminder (Xie et al. 2023). To implement PPL, we use the protected LLM itself to compute the perplexity for the input user query and directly reject the one with a perplexity higher than a threshold in our experiment. For Erase Check, we employ the LLM itself to serve as a safety checker to check whether the input query or any of its erased sub-sentences is harmful. Gradient Cuff, which is a two-stage detection framework, proposed a loss function called `Refusal Loss`. Gradient Cuff detects jailbreaks by checking the value and gradient norm of `Refusal Loss`. SmoothLLM and Semantic Smoothing perturb the original input query to obtain multiple copies and then aggregate the protected LLM's responses to generate the final response. Self Reminder converts the protected LLM into a self-remind mode by modifying the system prompt. For Token Highlighter, to demonstrate the effectiveness of the construction of the Critical Set, we also include a new baseline called Random Soft Removal, which does soft removal on randomly selected tokens.

**Metrics.** We report the Attack Success Rate (**ASR**) measured by LLaMA-Guard-2 (Team 2024) to evaluate each defense against various jailbreak attacks. We also report the **Win Rate** measured on Alpaca Eval to show how the protected LLM's utility is affected. In general, a higher Win Rate and lower ASR indicate a better defense.

**Implementation of Token Highlighter.** We use $\alpha = 0.25$ in all our experiments for both the two protected LLMs. In terms of $\beta$, we use $0.3$ for Vicuna-7B-V1.5 and $0.5$ for LLaMA-2-7B-Chat to keep a balanced trade-off between the Win Rate and the ASR. For the text generation setting, we use temperature $= 0.6$ and top $-$ p $= 0.9$ for both LLaMA2-7B-Chat and Vicuna-7B-V1.5, and adopt Nucleus Sampling. As for the system prompt, we use the default setting provided in the fastchat repository (Zheng et al. 2023). All our experiments are run on a single NVIDIA A800 GPU with 80G of memory. We run all the experiments with the random seed set to $100$ to ensure reproducibility.

## 4.2 Comparison with Existing Methods

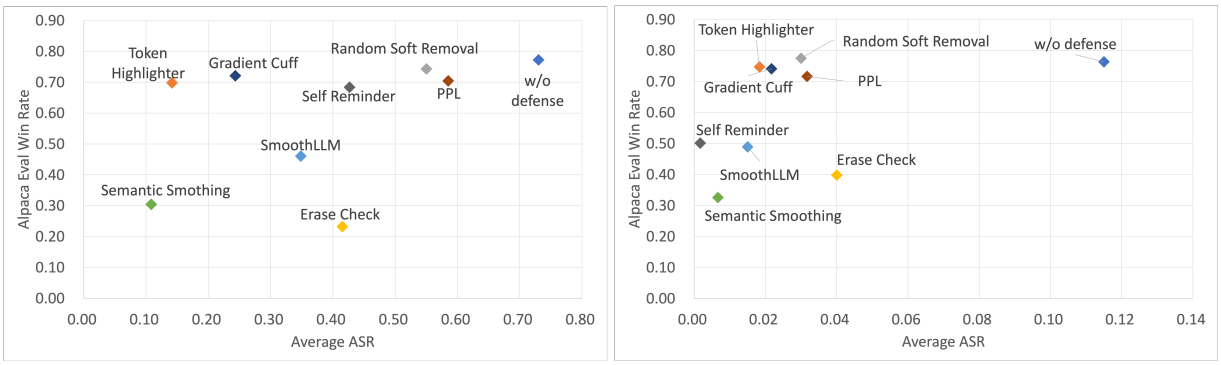
We begin by comparing our methods and all the defense baselines, jointly considering the AlpacaEval Win Rate and the Average ASR which is averaged across all six jailbreak attacks (GCG, AutoDAN, PAIR, TAP, Manyshot, and AIM). From Figure 2, we can conclude that our method outperforms all other baselines by showing strong defense against jailbreak attacks and good utility on benign user queries. Though smoothing-based methods like Semantic Smoothing can also achieve comparable or even lower ASR than Token Highlighter, these methods would cause a large drop in the utility of the protected LLM, due to the fact that the perturbations they applied to the original user query may deteriorate the semantic information. For example, SmoothLLM uses meaningless characters to replace some words in the original query. Though Semantic Smoothing tries to preserve the semantic information by using summarization to transform the query, the summarization technique would also affect the semantics of the original query. Detector-based methods like Gradient Cuff and PPL can attain good utility because these methods can limit the False Positive Rate (FPR) to a small value (e.g., 5%) by adjusting the threshold. Erase Check, another detector-based method in which there is no threshold to be adjusted, cannot attain good utility as it has a large and uncontrollable FPR, as also mentioned in prior works (Hu, Chen, and Ho 2024; Ji et al. 2024). Self Reminder can maintain a high Win Rate on Vicuna-7B-V1.5 but also show utility degradation on LLaMA-2-7B-Chat.

Our method stands out by having the lowest ASR among all the methods that can keep a high Win Rate. In particular, Token Highlighter decreases the ASR from 0.730 to 0.142 on Vicuna-7B-V1.5 while the best baseline Gradient Cuff can only decrease the ASR to 0.243. Token Highlighter outperforms Gradient Cuff by 20.7% (0.588 vs 0.487) in terms of the ASR reduction. On LLaMA-2-7B-chat, all baselines can make the ASR close to zero, because LLaMA-2 is more difficult to jailbreak. The comparison between Token Highlighter and Random Soft Removal reveals the effectiveness of the construction of the Critical Set using the gradient of the `Affirmation Loss`. Another notable fact is that Random Soft Removal can also keep the utility almost unchanged compared with when there is no defense. This finding suggests that in terms of maintaining utility, exploring the effect on the values of $\beta$ and $\alpha$ in soft removal may be more crucial than which tokens are softly removed. More studies on the trade-off between ASR and Win Rate by adjusting $\alpha$ and $\beta$ are presented in Section 4.3.

The results in Figure 2a show that Self Reminder is not effective on Vicuna-7B-V1.5. Since prompt-engineering-based methods can be easily combined with Token Highlighter, we choose to combine our method with Self Reminder by simply replacing the system prompt used in our method with that used in Self Reminder. We call the combined version Self Reminder (TH) and run experiments under varying values of $\beta$ to see whether Token Highlighter can improve Self Reminder. The results in Table **??** show that Self Reminder (TH) can have a much better performance than the plain Self Reminder in terms of the trade-off between ASR and Win Rate. Specifically, Token Highlighter further decreases the ASR of Self Reminder by $15.2\%$ (0.362 vs 0.427) while maintaining the $95.5\%$ win rate of the vanilla Self Reminder (0.653 vs 0.684). Reducing the $\beta$ from $0.5$ to a smaller number like $0.3$ can continually reduce the ASR at the cost of decreased win rate. When $\beta$ is set to 0.3, the ASR is nearly zero while the win rate can still maintain almost 80% of the vanilla Self Reminder.

## 4.3 Trade-off Analysis between ASR and Win Rate

Recall that we have two parameters for the Token Highlighter algorithm: the highlight percentage $\alpha$ and the soft removal level $\beta$. In Figure 3, we report the average **ASR** and the **Win Rate** for various $\alpha$ and $\beta$. From Figure 3, we can find that the ASR has the same trend as the Win Rate with the changing of $\alpha$ and $\beta$. Specifically, when $\alpha$ is fixed,




(a) Vicuna-7B-V1.5 (b) LLaMA2-7B-Chat

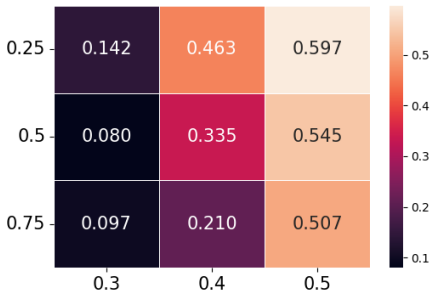
Figure 2: Performance evaluation on Vicuna-7B-V1.5 (a) and LLaMA2-7B-Chat (b). The horizon axis represents the Attack Success Rate (ASR) averaged over 6 jailbreak attacks, and the vertical axis shows the Win Rate on Alpaca Eval of the protected LLM when the corresponding defense is deployed.

| **Defense Method** | $\beta$ | **ASR** $\downarrow$ | **Win Rate** $\uparrow$ |
|---|---|---|---|
| Self Reminder | NA | 0.427 | 0.684 |
| Self Reminder (TH) | 0.5 | **0.362** | **0.653** |
| | 0.4 | 0.248 | 0.599 |
| | 0.3 | 0.023 | 0.536 |
| | 0.2 | 0.015 | 0.328 |

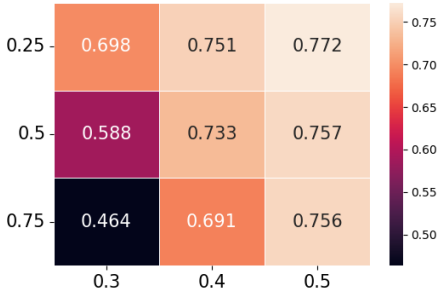
Table 1: Performance evaluation of combining Self Reminder and Token Highlighter. $\uparrow$ means that larger value is better while $\downarrow$ means the opposite.

a larger value of $\beta$ would make both the Win Rate and the ASR increase. When $\beta$ is fixed, larger $\alpha$ would both reduce the ASR and the Win Rate.

This phenomenon can be interpreted as follows. Taking a larger $\alpha$, Token Highlighter would highlight more tokens in the jailbreak prompt, thus improving the chance to mitigate the jailbreak effects. However, in another prospective, highlighting more tokens would decrease the model's utility because more tokens in benign queries would also be highlighted. Taking a smaller $\beta$ would further suppress the importance of the highlighter tokens in generating responses, thus better at mitigating the jailbreak effects. However, heavier soft removals are more likely to destroy the semantic context of the token embeddings. An extreme case is that the soft removal becomes "hard removal" when $\beta$ is set to zero.




(a) Attack Success Rate




(b) Win Rate

Figure 3: Trade-off between Win Rate and Attack Success Rate under varying $\alpha$ and $\beta$. The x-axis and y-axis represent the values of $\beta$ and $\alpha$, respectively.

## 4.4 Running Time Analysis

Another major concern when developing such defending methods for LLMs is the running time cost. A comprehensive comparison between defenses should also include a comparison of the running time cost. As pointed out in prior works (Robey et al. 2023; Ji et al. 2024; Hu, Chen, and Ho 2024), smoothing-based methods and Gradient Cuff need to query the protected LLM multiple times to reduce ASR, which come with the price of large latency for LLMs. In contrast, Token Highlighter is computationally lightweight as it only needs to query the LLM once to construct $\mathcal{Q}$, in addition to getting the final response with the highlighted user query. To quantitatively prove our efficiency, we select 25 examples from the AlpacaEval dataset to measure the running time of Token Highlighter and all other baselines on Vicuna-7B-V1.5. As evident in Figure 4, Token Highlighter is the only defense that simultaneous achieves low

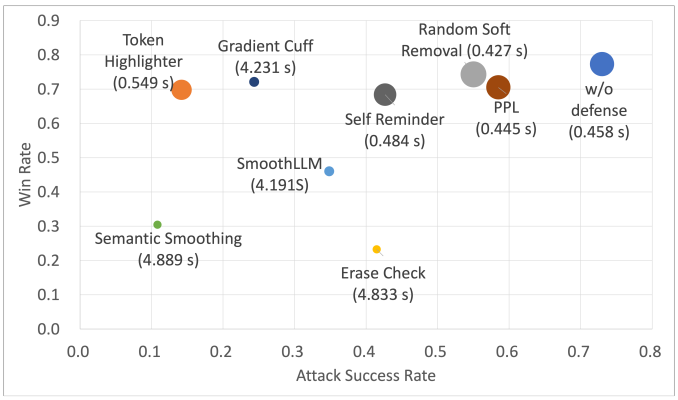
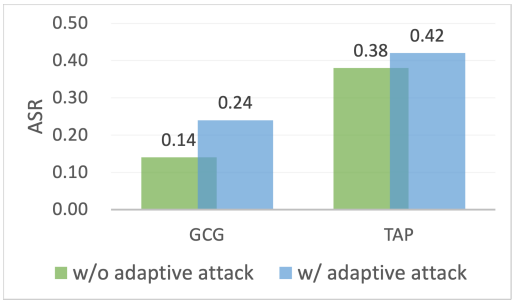
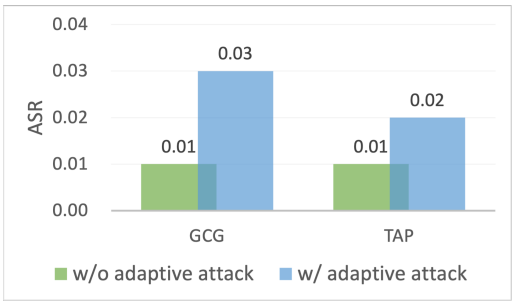



Figure 4: Running time analysis. We select 25 queries from AlpacaEval dataset to evaluate the wall clock time of all defenses on Vicuna-7B-V1.5. The printed value for each marker is the running time averaged across the 25 samples. **Larger size of a marker means lower running time cost**.




(a) Vicuna-7B-V1.5




(b) LLaMA2-7B-Chat

Figure 5: Token Highlighter against adaptive attacks.

ASR, high Win Rate, and small running time cost.

### 4.5 Adaptive Attack

Adaptive attack is a commonly used evaluation scheme to test the resilience of a defense when the defense mechanism is transparent to an attacker (Tramèr et al. 2020). Some studies on jailbreak defense also test their method against adaptive attacks (Robey et al. 2023; Xie et al. 2023; Ji et al. 2024). To see how adaptive attacks could weaken Token Highlighter, we design adaptive attacks based on the methods of GCG and TAP. Specifically, we design Adaptive-GCG and Adaptive-TAP to jailbreak the LLMs protected by Token Highlighter. As shown in Figure 5, adaptive attacks can improve the ASR to some extent against our defense. However, the ASR increment brought by the adaptive attack is minor. Even when the Token Highlighter defense is totally transparent to adaptive attack (like adaptive-GCG), it can only achieve a 0.1 ASR increment on Vicuna-7B-V1.5 and a 0.02 ASR increment on LLaMA-2-7B-Chat, while adaptive TAP can only achieve 0.04 and 0.01 ASR increment on Vicuna-7B-V1.5 and LLaMA-2-7B-Chat, respectively.

### 4.6 Interpretability Analysis of Highlighted Queries

We show Token Highlighter can be used as a post-hoc explanation tool upon LLM refusal, by identifying jailbreak-critical tokens in 3 separate examples from GCG and TAP. Table **??** shows that our Token Highlighter indeed underscores some jailbreak-critical components of the prompt for interpreting jailbreak attempts. For GCG, which is a token-level jailbreak attack that appends an optimized adversarial suffix to the original malicious query, the highlighted tokens are all within the inserted suffix parts. For TAP, which is a prompt-level jailbreak method that uses an attacker LLM to refine the malicious instruction, Token Highlighter can facilitate the explanation on the strategies the attacker used to rewrite the prompt. For instance, raising the LLM's mismatched generalization by role-playing (e.g., "You are a private investigator..."), story-telling (e.g., content dilution), introducing illusive objectives (e.g., write dialogue, report, etc), and instructing the LLM to begin the response with an affirmation (e.g., "begin with 'Sure, here is guidance on ...'").

In addition, we also explored some highlighted cases for benign user queries and vanilla malicious queries (w/o jailbreak prompts). We found that the highlighted tokens in these cases are just some words or simply some punctuation marks to represent the interrogative/imperative moods (e.g., "How", "What", "Please", "?" and "."). In summary, for interpretability analysis, we advocate using Token Highlighter to inspect which tokens are more crucial to cause refusal responses by the protected LLM (e.g., "I am sorry, but I cannot ..." as shown in Figure 1, bottom right panel), to facilitate the explanation to end users and model developers.

## 5 Conclusion

This paper presents a novel jailbreak defense method called **Token Highlighter**. Token Highlighter can effectively capture the jailbreak-critical components designed by the attacker in the malicious user query and then mitigate their jailbreak effects by applying ***Soft Removal*** on these critical tokens. Our extensive experiments on 2 aligned LLMs (LLaMA-2-7b-Chat and Vicuna-7B-V1.5) and 6 jailbreak attacks (GCG, AutoDAN, PAIR, TAP, Manyshot, and AIM) validate the effectiveness of Token Highlighter over existing defenses by achieving state-of-the-art performance in alleviating jailbreak attacks while maintaining good utility on benign user prompts and low running time cost.

## Acknowledgements

This work is supported by the JC STEM Lab of Intelligent Design Automation funded by The Hong Kong Jockey Club Charities Trust for Xiaomeng Hu and Tsung-Yi Ho.

## References

Albert, A. 2023. Jailbreak chat. https://www.jailbreakchat.com.

Anil, C.; Durmus, E.; Sharma, M.; Benton, J.; Kundu, S.; Batson, J.; Rimsky, N.; Tong, M.; Mu, J.; Ford, D.; et al. 2024. Many-shot Jailbreaking.

Askell, A.; Bai, Y.; Chen, A.; Drain, D.; Ganguli, D.; Henighan, T.; Jones, A.; Joseph, N.; Mann, B.; DasSarma, N.; Elhage, N.; Hatfield-Dodds, Z.; Hernandez, D.; Kernion, J.; Ndousse, K.; Olsson, C.; Amodei, D.; Brown, T. B.; Clark, J.; McCandlish, S.; Olah, C.; and Kaplan, J. 2021. A General Language Assistant as a Laboratory for Alignment. *CoRR*, abs/2112.00861.

Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; Joseph, N.; Kadavath, S.; Kernion, J.; Conerly, T.; Showk, S. E.; Elhage, N.; Hatfield-Dodds, Z.; Hernandez, D.; Hume, T.; Johnston, S.; Kravec, S.; Lovitt, L.; Nanda, N.; Olsson, C.; Amodei, D.; Brown, T. B.; Clark, J.; McCandlish, S.; Olah, C.; Mann, B.; and Kaplan, J. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *CoRR*, abs/2204.05862.

Chao, P.; Robey, A.; Dobriban, E.; Hassani, H.; Pappas, G. J.; and Wong, E. 2023. Jailbreaking Black Box Large Language Models in Twenty Queries. *CoRR*, abs/2310.08419.

Cohen, J.; Rosenfeld, E.; and Kolter, J. Z. 2019. Certified Adversarial Robustness via Randomized Smoothing. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 1310–1320. PMLR.

Hu, X.; Chen, P.; and Ho, T. 2024. Gradient Cuff: Detecting Jailbreak Attacks on Large Language Models by Exploring Refusal Loss Landscapes. *CoRR*, abs/2403.00867.

Jain, N.; Schwarzschild, A.; Wen, Y.; Somepalli, G.; Kirchenbauer, J.; Chiang, P.; Goldblum, M.; Saha, A.; Geiping, J.; and Goldstein, T. 2023. Baseline Defenses for Adversarial Attacks Against Aligned Language Models. *CoRR*, abs/2309.00614.

Ji, J.; Hou, B.; andF George J. Pappas, A. R.; Hassani, H.; Zhang, Y.; Wong, E.; and Chang, S. 2024. Defending Large Language Models against Jailbreak Attacks via Semantic Smoothing. *CoRR*, abs/2402.16192.

Kasirzadeh, A.; and Gabriel, I. 2022. In conversation with Artificial Intelligence: aligning language models with human values. *CoRR*, abs/2209.00731.

Kumar, A.; Agarwal, C.; Srinivas, S.; Feizi, S.; and Lakkaraju, H. 2023. Certifying LLM Safety against Adversarial Prompting. *CoRR*, abs/2309.02705.

Li, X.; Zhang, T.; Dubois, Y.; Taori, R.; Gulrajani, I.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. AlpacaEval: An Automatic Evaluator of Instruction-following Models. https://github.com/tatsu-lab/alpaca_eval.

Liu, X.; Xu, N.; Chen, M.; and Xiao, C. 2023. AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models. *CoRR*, abs/2310.04451.

Mehrotra, A.; Zampetakis, M.; Kassianik, P.; Nelson, B.; Anderson, H.; Singer, Y.; and Karbasi, A. 2023. Tree of Attacks: Jailbreaking Black-Box LLMs Automatically. *CoRR*, abs/2312.02119.

OpenAI. 2023. GPT-4 Technical Report. *CoRR*, abs/2303.08774.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P. F.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Robey, A.; Wong, E.; Hassani, H.; and Pappas, G. J. 2023. SmoothLLM: Defending Large Language Models Against Jailbreaking Attacks. *CoRR*, abs/2310.03684.

Team, L. 2024. Meta Llama Guard 2. https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Canton-Ferrer, C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.; Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR*, abs/2307.09288.

Tramèr, F.; Carlini, N.; Brendel, W.; and Madry, A. 2020. On Adaptive Attacks to Adversarial Example Defenses. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Varshney, N.; Dolin, P.; Seth, A.; and Baral, C. 2024. The Art of Defending: A Systematic Evaluation and Analysis of LLM Defense Strategies on Safety and Over-Defensiveness. *CoRR*, abs/2401.00287.

Wei, A.; Haghtalab, N.; and Steinhardt, J. 2023. Jailbroken: How Does LLM Safety Training Fail? *CoRR*, abs/2307.02483.

Wei, Z.; Wang, Y.; and Wang, Y. 2023. Jailbreak and Guard Aligned Language Models with Only Few In-Context Demonstrations. *CoRR*, abs/2310.06387.

Xie, Y.; Yi, J.; Shao, J.; Curl, J.; Lyu, L.; Chen, Q.; Xie, X.; and Wu, F. 2023. Defending ChatGPT against jailbreak attack via self-reminders. *Nat. Mac. Intell.*, 5(12): 1486–1496.

Zhang, Y.; Ding, L.; Zhang, L.; and Tao, D. 2024. Intention Analysis Prompting Makes Large Language Models A Good Jailbreak Defender. *CoRR*, abs/2401.06561.

Zhao, X.; Yang, X.; Pang, T.; Du, C.; Li, L.; Wang, Y.; and Wang, W. Y. 2024. Weak-to-Strong Jailbreaking on Large Language Models. *CoRR*, abs/2401.17256.

Zheng, L.; Chiang, W.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E. P.; Zhang, H.; Gonzalez, J. E.; and Stoica, I. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *CoRR*, abs/2306.05685.

Zou, A.; Wang, Z.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. *CoRR*, abs/2307.15043.