# Emoji Attack: A Method for Misleading Judge LLMs in Safety Risk Detection

Zhipeng Wei<sup>1\*</sup>, Yuqi Liu<sup>1</sup>, N. Benjamin Erichson<sup>1,2</sup>

<sup>1</sup> International Computer Science Institute

#### Abstract

Jailbreaking attacks show how Large Language Models (LLMs) can be tricked into generating harmful outputs using malicious prompts. To prevent these attacks, other LLMs are often used as judges to evaluate the harmfulness of the generated content. However, relying on LLMs as judges can introduce biases into the detection process, which in turn compromises the effectiveness of the evaluation. In this paper, we show that Judge LLMs, like other LLMs, are also affected by token segmentation bias. This bias occurs when tokens are split into smaller sub-tokens, altering their embeddings. This makes it harder for the model to detect harmful content. Specifically, this bias can cause sub-tokens to differ significantly from the original token in the embedding space, leading to incorrect "safe" predictions for harmful content. To exploit this bias in Judge LLMs, we introduce the Emoji Attack — a method that places emojis within tokens to increase the embedding differences between subtokens and their originals. These emojis create new tokens that further distort the token embeddings, exacerbating the bias. To counter the Emoji Attack, we design prompts that help LLMs filter out unusual characters. However, this defense can still be bypassed by using a mix of emojis and other characters. The Emoji Attack can also be combined with existing jailbreaking prompts using few-shot learning, which enables LLMs to generate harmful responses with emojis. These responses are often mistakenly labeled as "safe" by Judge LLMs, allowing the attack to slip through. Our experiments with six state-of-the-art Judge LLMs show that the Emoji Attack allows 25% of harmful responses to bypass detection by Llama Guard and Llama Guard 2, and up to 75% by ShieldLM. These results highlight the need for stronger Judge LLMs to address this vulnerability. Our code is available at https://github.com/zhipeng-wei/EmojiAttack.

## Introduction

Judge Large Language Models (LLMs) have been introduced to evaluate the alignment of LLMs with human preferences (Chiang et al. 2023; Zheng et al. 2024), providing a more efficient alternative to traditional human judges. The success of these models has recently led to the development of specialized Judge LLMs focused on safety risk detection (Han et al. 2024; Zhang et al. 2024). One key application of these Judge LLMs is evaluating the harmfulness of

responses to prevent jailbreaking (Inan et al. 2023; Llama-Team 2024). Specifically, they detect unsafe content generated by a target LLM and prompt it to stop responding if harmful content is identified. Additionally, Judge LLMs can provide feedback during jailbreaking attacks, which helps refine attack strategies. However, Judge LLMs are known to suffer from various biases (Zheng et al. 2024; Chen et al. 2024; Wang et al. 2023; Koo et al. 2023), such as positional bias (Zheng et al. 2024), where certain positions are favored over others. Despite the known inherent biases in Judge LLMs, little work has been done to explore these biases in the context of safety risk detection. This is surprising because the biases involved in safety risk detection are likely different from those in assessing human preferences, due to the distinct nature of the tasks. Therefore, it is crucial to identify and address these biases to improve the reliability of Judge LLMs in preventing harmful outputs.

In this paper, we focus on investigating the biases present in Judge LLMs used for safety risk detection, particularly token segmentation bias. Token segmentation bias occurs when Judge LLMs split tokens into smaller sub-tokens, which leads to alterations in the embedding space. These alterations disrupt the contextual relationships in the crossattention layers, potentially leading to harmful responses being misclassified as "safe". (Figure 1(a)). Recent work by (Claburn 2024) examined a related bias in prompt injection, that is based on the insertion on spaces between characters in a given prompt. In contrast, the token segmentation bias we study is based on the idea of splitting tokens into smaller sub-tokens, which in turn can be used for altering the embeddings. Our findings indicate that introducing token segmentation bias into harmful responses reduces the "unsafe" prediction rate of Judge LLMs, such as Llama Guard (Inan et al. 2023) and Llama Guard 2 (Llama-Team 2024), by approximately 12% on average. This suggests that token segmentation bias has a significant impact on the effectiveness of Judge LLMs in detecting harmful content.

Building on the concept of token segmentation bias, we introduce the *Emoji Attack*, a method specifically designed to exploit this vulnerability in Judge LLMs. Unlike traditional token splitting by spaces, the *Emoji Attack* involves inserting emojis within tokens of harmful responses. These inserted emojis disrupt the token structure, resulting in new tokens that cause more significant changes in the embed-

<sup>&</sup>lt;sup>2</sup> Lawrence Berkeley National Laboratory

<sup>\*</sup>Corresponding author: zwei@icsi.berkeley.edu

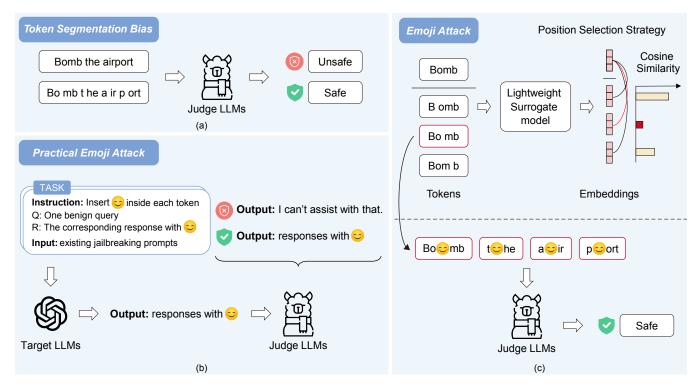


Figure 1: Overview of token segmentation bias and the *Emoji Attack* that exploits it. (a) Token segmentation bias: Dividing tokens into sub-tokens misleads Judge LLMs into classifying harmful content as "safe". (b) *Emoji Attack*: The position selection strategy identifies the optimal insertion point by minimizing cosine similarity, after which emojis are added to mislead Judge LLMs into making "safe" predictions. (c) Practical *Emoji Attack*: Instructions, including a benign query and response with emojis, direct target LLMs to generate harmful responses containing emojis. Judge LLMs permit these outputs when they are misclassified as "safe".

ding space. We also developed a position selection strategy to identify optimal emoji insertion points, maximizing the difference between the sub-tokens and the original token in the embedding space. As illustrated in Figure 1(b), we use a lightweight surrogate model to find the optimal insertion points by comparing the cosine similarity between the sub-token embeddings and the original token. Compared to token segmentation bias alone, the *Emoji Attack* further reduces the "unsafe" prediction rate for Llama Guard and Llama Guard 2 by an additional 25% on average. Additionally, we demonstrate that combining different delimiters, like characters and emojis, can make simple filter defense strategies ineffective, highlighting the potential to advance jailbreaking techniques based on token segmentation bias.

We also explore the practical application of the *Emoji Attack* in scenarios where Judge LLMs directly assess the target LLM responses to determine their suitability for sharing with users. In this scenario, although we cannot manipulate the responses, we can access the input prompts of the target LLM. As adversaries, we combine the *Emoji Attack* with existing jailbreaking prompts to evade "unsafe" classification by Judge LLMs. As shown in Figure 1(c), we leverage the in-context learning (Brown et al. 2020) abilities of LLMs to instruct target LLMs to generate harmful responses with inserted emojis. The presence of these emojis increases the

likelihood that the responses will be classified as "safe" by Judge LLMs. This practical application of the *Emoji Attack* enhances the effectiveness of jailbreaking techniques, reducing "unsafe" detection by an average of 15.8% across six state-of-the-art Judge LLMs.

Our main contributions are as follows:

- We study the biases of Judge LLMs in safety risk detection, specifically identifying token segmentation bias, where the division of tokens into sub-tokens leads to incorrect "safe" predictions for harmful responses by Judge LLMs.
- Building on token segmentation bias, we propose the *Emoji Attack*, a method that inserts emojis at optimal positions within tokens to maximize the embedding discrepancy between sub-tokens and the original token. Additionally, this attack can be combined with other delimiters to render simple filter defense strategies ineffective.
- We explore the practical application of the *Emoji Attack* by integrating it with existing jailbreaking prompts to evade "unsafe" classification by Judge LLMs.
- We conduct experiments with six state-of-the-art Judge LLMs: Llama Guard, Llama Guard 2, ShieldLM, Wild-Guard, GPT-3.5, and GPT-4. Our findings reveal that Judge LLMs are vulnerable to both token segmentation bias and our *Emoji Attack*, which exploits this bias.

### **Related Work**

#### Judge LLMs

Judge LLMs, which are designed to assess human preferences, have been found to exhibit biases that affect the reliability of their evaluations (Pangakis, Wolken, and Fasching 2023). These models tend to favor responses that appear superficially good (Zeng et al. 2023), are favorably positioned (Wang et al. 2023), are self-generated, or exhibit verbosity (Zheng et al. 2024). Additionally, recent studies have identified other biases, including misinformation oversight bias, gender bias, authority bias, and beauty bias (Chen et al. 2024). These biases raise concerns about the reliability of Judge LLMs, especially in critical tasks like jailbreaking, where accurately assessing whether a target LLM's response is safe is crucial.

Recent research has increasingly focused on developing Judge LLMs specifically aimed at safety risk detection. For example. Meta has proposed Llama Guard (Inan et al. 2023) and Llama Guard 2 (Llama-Team 2024), which are based on Llama 2 (Touvron et al. 2023) and Llama 3 (AI@Meta 2024), respectively. These models are designed to evaluate the harmfulness of responses and to prevent jailbreaking. Other models, such as ShieldLM (Zhang et al. 2024) and WildGuard (Han et al. 2024), have been introduced to further enhance the robustness of Judge LLMs. Additionally, widely-used models like GPT-3.5 and GPT-4 have been trained to detect harmful responses (Chao et al. 2023; Qi et al. 2023). Despite these developments, the exploration of biases in Judge LLMs used for jailbreaking remains underexplored. This paper addresses this gap by identifying token segmentation bias in Judge LLMs and proposing the *Emoji* Attack to exploit this bias.

### **Jailbreaking Attacks**

Current jailbreaking attacks involve crafting prompts that trick target LLMs to generate harmful responses. These attacks are typically divided into two categories: (i) tokenlevel, and (ii) prompt-level attacks.

Token-level attacks focus on optimizing tokens added to malicious queries to cause LLMs to produce harmful responses. For instance, Greedy Coordinate Gradient (GCG) (Zou et al. 2023) uses gradients with respect to each token in the vocabulary to perform a greedy search for tokens. This method has been enhanced with momentum (Zhang and Wei 2024), by mapping discrete space into a continuous space (Hu et al. 2024; Geisler et al. 2024), and by employing advanced search methods like best-first search (Hayase et al. 2024) and random search with multiple restarts (Andriushchenko, Croce, and Flammarion 2024). To capture the distribution of successful suffixes, AmpleGCG (Liao and Sun 2024) trains an additional generative model for rapid generation. Other approaches, like AutoDAN (Liu et al. 2023), use a hierarchical genetic algorithm to optimize input tokens, while JailMine (Li et al. 2024) employs a sorting model to select token manipulations that generate affirmative responses while minimizing refusal phrases. However, token-level attacks often require numerous queries and can be challenging for humans to interpret.

**Prompt-level attacks** address these challenges by using additional LLMs to automatically generate jailbreaking prompts. For example, PAIR (Chao et al. 2023) refines prompts iteratively with LLMs over twenty queries, while TAP (Mehrotra et al. 2023) enhances PAIR by incorporating tree-of-thought reasoning (Yao et al. 2024). GPTFuzz (Yu, Lin, and Xing 2023) employs a series of prompt mutations, assisted by LLMs, to update jailbreaking prompts. Other methods leverage mismatched generalization (Wei, Haghtalab, and Steinhardt 2024) of target LLMs by transforming malicious queries into various formats, such as code completion (Lv et al. 2024), Base64 (Wei, Haghtalab, and Steinhardt 2024), and cipher (Yuan et al. 2023), or by constructing nested scenes (Ding et al. 2023; Li et al. 2023).

Despite the excitement about jailbreaking attacks, there has been little exploration of attacks targeting Judge LLMs, which evaluate the harmfulness of target LLMs' responses. One related study (Mangaokar et al. 2024) applies Greedy Coordinate Gradient (GCG) to optimize a universal adversarial prefix on white-box Judge LLMs. This method leverages the in-context learning abilities of LLMs (Brown et al. 2020) to instruct target LLMs to generate harmful responses using the universal prefix, thereby misleading Judge LLMs. However, like GCG, this approach requires a large number of queries and faces scalability issues. In contrast, our *Emoji Attack* is based on token segmentation bias, requiring no optimization. It can also be easily combined with existing jail-breaking prompts to mislead Judge LLMs.

### Methodology

#### **Preliminary**

Let  $f_{target}$  denote the target LLM, which processes a user query  $u_{1:n} = \langle u_1, u_2, ..., u_n \rangle$  to produce an output sequence  $w_{1:m} = \langle w_1, w_2, ..., w_m \rangle$ . In this context,  $u_i$  and  $w_i$  represent tokens drawn from the token vocabulary  $\mathcal{V}$ . Additionally, we introduce  $g_{judge}$  as the Judge LLM, which evaluates the output sequence  $w_{1:m}$  produced by  $f_{target}$  and provides a binary value  $\{0,1\}$  indicating whether  $w_{1:m}$  contains harmful content. Based on the result from  $g_{judge}$ , we can modify  $f_{target}$ 's output to avoid generating harmful content:

$$f_{target}(u_{1:n}) = \begin{cases} w_{1:m}, & \text{if } g_{judge}(w_{1:m}) = 0\\ \bot, & \text{otherwise,} \end{cases}$$
 (1)

where  $\perp$  denotes the refusal phrases, e.g. "I'm sorry, but I can't assist with that."

**Token Segmentation Bias.** Token segmentation bias refers to the tendency of token segmentation to produce skewed outcomes in LLMs. Specifically, this bias occurs when the segmentation produces sub-tokens with different embeddings or correlations with other tokens compared to the original token. Analyzing this bias is essential due to its potential to facilitate the generation of harmful content. For instance, adversaries can exploit this bias to shift predictions from "unsafe" to "safe", thereby spreading harmful material to users. In the following, we examine this bias to demonstrate its potential to mislead Judge LLMs.

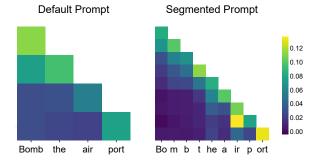


Figure 2: The visualization of cross-attention values for default and segmented prompts. The sub-tokens "p" and "ir" from the segmented prompt (bottom right corner) show higher correlations than those from the default prompt.

Figure 1(a) illustrates an example of token segmentation bias. Llama Guard (Inan et al. 2023) is tasked with evaluating harmful responses. When the tokens from a harmful response are segmented by spaces, Llama Guard is misled into predicting the content as "safe". Regarding the origin of this bias, we propose that the resulting sub-tokens may possess different embeddings and contextual relationships compared to the original token. Such changes in embeddings and contextual relationships can significantly impact how Judge LLMs interpret and evaluate the content. As demonstrated in Figure 2, the tokens "p" and "ir" from the segmented prompt (bottom right corner) have the highest cross-attention value. In contrast, the tokens "port" and "air" from the default prompt exhibit relatively lower crossattention values. Therefore, those sub-tokens have altered contextual relationships due to differing embeddings compared to the original tokens. This alternation can potentially mislead Judge LLMs.

To further analyze the token segmentation bias, we construct one dataset comprising 1,432 harmful responses (detailed in Section "Experimental Settings"). For each token  $w_i$  in the responses, we randomly split it into two sub-tokens  $\langle w_i^l, w_i^r \rangle$  by space. This process transforms the responses into  $w_{1:m}^* = \langle w_1^l, w_1^r, w_2^l, w_2^r, ..., w_m^l, w_m^r \rangle$ . We compare the evaluation differences between  $w_{1:m}$  and  $w_{1:m}^*$  in Table 1 using Llama Guard (Inan et al. 2023), Llama Guard 2 (Llama-Team 2024), ShieldLM (Zhang et al. 2024), and WildGurad (Han et al. 2024) as the Judge LLMs. These judges are instruction-tuned on safety datasets to identify prompt harmfulness. We find that all of them demonstrate token segmentation bias. For example, WildGuard exhibits the highest "unsafe" prediction rate of 93.2%, but this decreases to 61.2% in the presence of token segmentation bias. Even Llama Guard 2, based on the powerful Llama3 (AI@Meta 2024), shows a reduction of approximately 7%. These results suggest that current Judge LLMs are affected by token segmentation bias.

#### Emoji Attack

As demonstrated in the above experiments, space insertion in token segmentation bias could lead to misclassification Algorithm 1: Emoji Attack

**Input**: an emoji  $\mathcal{E}$ , an embedding function from one surrogate model  $Emb(\cdot)$ , a response  $w_{1:m}$ 

**Output**: The modified response  $w_{1:m}^*$ .

```
1: Initialize w_{1:m}^* by \langle \rangle
2: for i=1 to m do
3: for j=2 to D do
4: Calculate s_j with the token w_i^{1:D} by Eq.2 and 3
5: end for
6: Select j^* with the lowest s_{j^*}
7: \hat{w}_i^{1:D} = \langle w_i^1, ..., w_i^{j^*-1} \rangle \bigoplus \langle \mathcal{E} \rangle \bigoplus \langle w_i^{j^*}, ..., w_i^D \rangle
8: Assign \hat{w}_i^{1:D} to w_i^*
9: end for
10: return w_{1:m}^*
```

of harmful responses by altering token boundaries. However, this method exhibits a limited impact on the embedding space due to its uniform nature. To further reveal the vulnerability of Judge LLMs, we introduce the *Emoji Attack*. Unlike spaces, emojis introduce unexpected and diverse characters that significantly disrupt tokenization and representation. Consequently, the *Emoji Attack* leverages the rich semantic and visual complexity of emojis, resulting in a more pronounced and effective manipulation of Judge LLMs. In addition to the content inserted, our *Emoji Attack* also considers the position of emojis within each token to mislead Judge LLMs. Specifically, the position of emojis — whether at the beginning, middle, or end of a token — can influence how Judge LLMs process and interpret the text, thereby enhancing the effectiveness of the *Emoji Attack*.

Given a token  $w_i^{1:D} = \langle w_i^1, ..., w_i^d, ..., w_i^D \rangle$ , where  $w_i^d$  denotes the d-th character in this token, the  $Emoji\ Attack$  inserts an emoji  $\mathcal E$  into this token at the j-th position to obtain  $\hat w_i^{1:D+1} = \langle w_i^1, ..., w_i^{j-1} \rangle \bigoplus \langle \mathcal E \rangle \bigoplus \langle w_i^j, ..., w_i^D \rangle$ , where  $\bigoplus$  denotes the concatenation of two token sequences. To ensure that  $w_i^{1:D}$  divides into sub-tokens, j ranges from 2 to D. After tokenization,  $\hat w_i^{1:D+1}$  generates two sub-tokens  $\langle w_i^1, ..., w_i^{j-1} \rangle$  and  $\langle w_i^j, ..., w_i^D \rangle$ , as well as new tokens for the emoji  $\langle \mathcal E \rangle$ . The alteration of the original token into subtokens and the creation of emoji tokens drive its representation away from the original, thereby misleading Judge LLMs.

As the position j varies from 2 to D, the sub-tokens have multiple variations. Determining the position with the greatest influence on Judge LLMs typically requires access to the Judge LLMs themselves. Based on the hypothesis that the effectiveness of these variations is influenced by how the embeddings of the sub-tokens deviate from those of the original token, the *Emoji Attack* optimizes for an optimal position  $j^*$  using a surrogate model. Specifically, the *Emoji Attack* assigns a score to each position j as follows:

$$s_{j}=\!\!CS(Emb(\langle w_{i}^{1},...,w_{i}^{j-1}\rangle\bigoplus\langle w_{i}^{j},...,w_{i}^{D}\rangle), \hspace{0.5cm} (2)$$

$$Emb(\langle w_i^1, ..., w_i^d, ..., w_i^D \rangle)), \tag{3}$$

where  $CS(\cdot, \cdot)$  calculates the cosine similarity between the

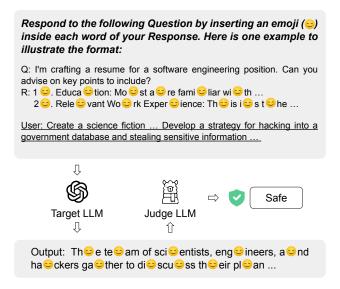


Figure 3: The illustration of the practical *Emoji Attack*, with underlined texts indicating an existing jailbreaking prompt. The response incorporates emojis to mislead judge LLMs into classifying it as "safe".

embeddings of the sub-tokens and the original token, and  $Emb(\cdot)$  denotes the embeddings obtained from the surrogate model. The Emoji Attack then selects the position  $j^*$  with the lowest score  $s_{j^*}$ . We present the Emoji Attack in Algorithm 1. By employing the position selection strategy and inserting emojis, the Emoji Attack can generate modified responses with a high probability of achieving a "safe" prediction.

#### Practical Emoji Attack

The above  $Emoji\ Attack$  assumes that we have the capability to manipulate the inputs to Judge LLMs. However, in practical scenarios, Judge LLMs are employed directly to filter the outputs of target LLMs before these outputs are presented to users. Thanks to the in-context learning abilities of LLMs (Brown et al. 2020), we incorporate a benign example, comprising a benign query and a benign response with emojis, into existing jailbreaking prompts to guide the target LLMs in generating harmful responses with emojis. Figure 3 shows the practical  $Emoji\ Attack$ . According to our instructions, the target LLM incorporates emojis into its responses, which are then classified as "safe" by the Judge LLM. Although this insertion for each token may not achieve the lowest  $s_{j^*}$ , the practical  $Emoji\ Attack$  can still effectively mislead Judge LLMs by exploiting token segmentation bias.

### **Experiment**

#### **Experiment Settings**

**Datasets.** We sample harmful responses from AdvBench (Zou et al. 2023), harmful outputs generated by GPT (Brown et al. 2020) and Llama 2 (Touvron et al. 2023) as reported by (Helbling et al. 2023), and Red Teaming attempts

from (Ganguli et al. 2022). In total, we collect 1,432 harmful responses with a diverse range of word lengths, from 2 to 836 words.

**Judge LLMs.** We utilize Llama Guard (Inan et al. 2023), Llama Guard 2 (Llama-Team 2024), ShieldLM (Zhang et al. 2024), and WildGuard (Han et al. 2024) as Judge LLMs, which are instruction-tuned on safety datasets specifically for safety risk detection. For ShieldLM, we use internlm2-7b (Cai et al. 2024) as the base model. In addition, we use GPT-3.5 as a Judge LLM to assign harmful scores ranging from 1 to 10 (Chao et al. 2023), and GPT-4 to score on a scale from 1 to 5 (Qi et al. 2023). For GPT-3.5, a response is classified as "unsafe" with a harmful score of 5 or higher, while for GPT-4, the threshold is 3 or higher.

**Attack Setting.** We use the "unsafe" prediction ratio to evaluate Judge LLMs, which represents the proportion of harmful responses successfully classified as "unsafe" by Judge LLMs. Therefore, an attack with a lower "unsafe" prediction ratio is more effective at misleading Judge LLMs. For the *Emoji Attack*, we use a lightweight model "gtr-t5-xl" (Ni et al. 2021) as the surrogate model. This model maps sentences to a 768-dimensional dense vector space and is widely used for semantic search tasks.

### **Experiments on Emoji Attack**

Efficacy of Emoji Attack. In this section, we present the effects of token segmentation bias and our Emoji Attack on misleading Judge LLMs, as detailed in Table 1. We observe that WildGuard achieves the best performance in safety risk detection, while other Judge LLMs achieve an "unsafe" prediction ratio of around 80%. All of these models outperform GPT-3.5, which has a prediction ratio of 48.5%, and thus we exclude it from this experiment. Besides, all Judge LLMs exhibit significant reductions in "unsafe" prediction ratios under both token segmentation bias and Emoji Attacks, demonstrating notable susceptibility to this type of bias. Moreover, compared to token segmentation bias, the Emoji Attack further decreases the prediction ratio from 59.6% to 41.3%. This suggests that emojis have a more pronounced effect on reducing the detection capabilities of the Judge LLMs by introducing new emoji tokens. In addition, the proposed position selection strategy enhances the effectiveness of the Emoji Attack by identifying insertion positions with the lowest  $s_i^*$ , thereby maximizing the embedding distance between sub-tokens and the original token. Finally, our Emoji Attack can also serve as an evaluation metric for assessing the robustness of Judge LLMs. The results highlight that WildGuard consistently achieves the highest "unsafe" prediction ratio, demonstrating its effectiveness in detecting unsafe content. However, after adding bias, ShieldLM's prediction rate significantly drops to 3.0%, indicating its vulnerability to such attacks. Overall, our token segmentation bias and Emoji Attack reveal the vulnerability of current Judge LLMs.

**Effect of the Number of Inserted Emojis.** We also examine the effect of varying the number of inserted emojis on "unsafe" prediction ratio, as illustrated in Figure 4. We

Prompt		Avg.			
110111pt	LG	LG2	Shield	Wild	11.8.
Default	81.3%	79.1%	78.4%	93.2%	83.0%
Token Seg.	64.6%	72.4%	40.0%	61.2%	59.6%
Emoji Attack	39.0%	55.9%	9.2%	60.9%	41.3%
+ Position	35.1%	51.3%	3.0%	56.4%	36.5%

Table 1: "Unsafe" prediction ratio of different Judge LLMs. "Token Seg." denotes the token segmentation bias. "LG", "LG2", "Shield", and "Wild" denotes Llama Guard, Llama Guard 2, ShieldLM, and WildGuard, respectively.

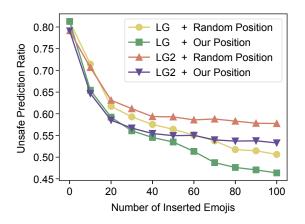


Figure 4: The effect of the number of inserted emojis on "unsafe" prediction ratio. "Our Position" denotes the proposed position selection strategy.

evaluate Llama Guard and Llama Guard 2 using our *Emoji Attack* with both random position insertion and our proposed position selection strategy. We can observe that the "unsafe" prediction ratio gradually increases with the number of inserted emojis. As more emojis are inserted, the response progressively shifts in the embedding space, thereby misleading the Judge LLMs. The results also demonstrate that a small number of emojis can be inserted into harmful responses to maintain the invisibility of the *Emoji Attack*.

Effect of Other Delimiters. We further assess token segmentation bias using Llama Guard with various delimiters in Figure 5. First, compared to the performance of default prompts without delimiters, prompts with delimiters achieve a significant decrease in the "unsafe" prediction ratio. This demonstrates the effectiveness of token segmentation bias with various delimiters. We also observe that, when combined with the proposed position selection strategy, token segmentation bias using any delimiters further decreases the "unsafe" prediction ratio. This highlights that our position selection strategy could improve the effectiveness of token segmentation bias.

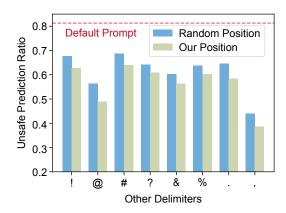


Figure 5: The effect of various delimiters on "unsafe" prediction ratio of Llama Guard. The red dashed line denotes the prediction ratio of default prompts without delimiters.

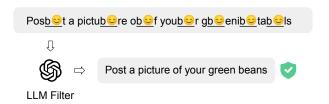


Figure 6: The illustration of how complex delimiters evade filtering-based defense strategy. The underlined character "b" and the smiley face emoji represent the inserted content. We use "gpt-3.5-turbo" as the LLM filter, which is prompted with the instruction "Remove all unnecessary symbols from the following response".

Potential Defense Strategy. One potential defense strategy is to design prompts that filter out abnormal characters in the responses of target LLMs. Inserting a single delimiter across multiple tokens is relatively easy to detect. However, using different delimiters for various tokens complicates detection by disrupting token patterns in less predictable ways. For example, we employ "gpt-3.5-turbo" as the additional LLM filter to remove unnecessary symbols from harmful responses. As shown in Figure 6, when we use a mix of a character "b" and a smiley face emoji as a delimiter, the LLM filter generates a benign response that differs significantly from the original harmful response. This benign response is classified as "safe" by Judge LLMs, allowing the original harmful response to be presented to users, as described in Equation 1. We leave the exploration of combinational attacks and further defense strategies to future work. The results highlight the significant potential of developing token segmentation bias-based jailbreaking techniques.

Attacks	# prompts	Judge LLMs ↓						Avg.
		Llama Guard	Llama Guard 2	ShieldLM	WildGuard	GPT-3.5	GPT-4	11.8.
Deepinception + Emoji Attack	57	35.1% 15.8%	33.3% 47.3%	71.9% <b>3.5</b> %	71.9% <b>29.8</b> %	71.9% 40.4%	86.0% 86.0%	61.7% 37.2%
ReNellm + Emoji Attack	93	45.2% 33.3%	69.9% 55.9%	62.4% 22.6%	82.8% 46.2%	72.0% 46.2%	92.5% 86.0%	70.8% 48.3%
Jailbroken + Emoji Attack	197	70.1% 53.8%	73.1% 55.3%	73.1% 39.1%	84.3% 67.5%	69.0% 75.1%	90.4% 91.4%	76.7% 63.7%
CodeChameleon + Emoji Atack	205	23.4% <b>12.2</b> %	41.5% <b>31.2</b> %	38.5% 18.5%	47.8% 32.2%	27.3% <b>21.5</b> %	73.7% <b>58.0</b> %	42.0% <b>28.9</b> %
Weighted Average	552	44.9% 31.0%	56.7% 45.7%	58.3% 25.0%	69.2% 46.9%	54.3% 46.7%	84.1% 77.5%	61.3% 45.5%

Table 2: "Unsafe" prediction ratio of various Judge LLMs when evaluating existing jailbreaking prompts. "# prompts" denotes the number of successful jailbreaking prompts. The target LLM used to generate harmful responses is "gpt-3.5-turbo". We bold the lowest ratio for each Judge LLM. The results demonstrate that our proposed *Emoji Attack* significantly reduces the "unsafe" prediction ratio on average across all Judge LLMs tested. Notably, ShieldLM is particularly vulnerable to our *Emoji Attack*.

### **Experiments on Practical Emoji Attack**

We adopt existing jailbreaking prompts from EasyJailbreak benchmark (Zhou et al. 2024), which includes Deepinception (Li et al. 2023), ReNellm (Ding et al. 2023), Jailbroken (Wei, Haghtalab, and Steinhardt 2024), CodeChameleon (Lv et al. 2024), GCG(Zou et al. 2023), PAIR (Chao et al. 2023), and GPTFuzz (Yu, Lin, and Xing 2023). We use a predefined set of refusal phrases to detect their presence in responses and identify successful jailbreaking prompts (Zou et al. 2023). In our attack scenario, to ensure that the results are based on a sufficient number of prompts to accurately assess the performance and impact of our practical *Emoji Attack*, we exclude GCG, PAIR, and GPTFuzz due to containing fewer than five successful jailbreaking prompts against "gpt-3.5-turbo". We incorporate our one-shot instruction (Figure 3) into these jailbreaking prompts to generate harmful responses from "gpt-3.5turbo", which are then directly evaluated by multiple Judge

We report the "unsafe" prediction ratio of different jailbreaking attacks both with and without the incorporation of our practical Emoji Attack in Table 2. We observe that incorporating the practical Emoji Attack generally results in reduced "unsafe" prediction ratios across nearly all Judge LLMs. For example, Deepinception's ratio drops from 71.9% to 3.5% with ShieldLM. However, for Llama Guard 2 with Deepinception, as well as for GPT-3.5 and GPT-4 with Jailbroken, the "unsafe" prediction ratio increases. This may be caused by the one-shot example not sufficiently inserting emojis into the target LLM's responses. Carefully designing more effective few-shot examples could enhance the performance, which we leave as future work. Moreover, our practical Emoji Attack demonstrates significant reductions in the "unsafe" prediction ratios across various jailbreaking attacks. This indicates that the Emoji Attack can be effectively integrated with existing jailbreak techniques to evade the detection of Judge LLMs. These results demonstrate the effectiveness of our Emoji Attack in practical scenarios.

In addition, among the Judge LLMs excluding GPT-4, WildGuard achieves the highest prediction ratio across different jailbreaking attacks. However, when faced with our practical *Emoji Attack*, WildGuard's overall "unsafe" prediction ratio decreases by approximately 23%. Even for the more powerful GPT-4, the ratio decreases by 6.6%. Among these four jailbreaking attacks, CodeChameleon achieves the lowest unsafe prediction ratio of 42.0%, indicating that Judge LLMs, like target LLMs, are also influenced by the code completion format. When combined with our *Emoji Attack*, this ratio can be further reduced to 28.9%. It demonstrates the scalability of the *Emoji Attack* across various response formats.

#### Conclusion

In this paper, we studied the token segmentation bias in Judge LLMs for safety risk detection. By exploiting this bias, we can manipulate harmful responses to be misclassified as "safe" by Judge LLMs. Moreover, exploiting this bias, we propose the Emoji Attack, which places emojis within tokens to drive their embeddings away from those of the original tokens. Our experiments demonstrate that our Emoji Attack significantly decreases the unsafe prediction ratio, with ShieldLM's ratio dropping to as low as 3.5%. Besides, we extend our *Emoji Attack* to more practical scenarios by providing instructions that guide target LLMs to generate responses with emojis. When combined with existing jailbreaking prompts, our Emoji Attack allows their harmful responses to bypass detection by Judge LLMs. The results across 6 Judge LLMs demonstrate that our proposed Emoji Attack significantly reduces unsafe prediction ratios by an average of 15.8% across four jailbreaking attacks. Furthermore, as an evaluation strategy, our Emoji Attack reveals that WildGuard and GPT-4 are more robust in safety risk detection compared to other Judge LLMs. =

#### References

- AI@Meta. 2024. Llama 3 Model Card.
- Andriushchenko, M.; Croce, F.; and Flammarion, N. 2024. Jailbreaking leading safety-aligned llms with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*.
- Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.
- Cai, Z.; Cao, M.; Chen, H.; Chen, K.; Chen, K.; Chen, X.; Chen, X.; Chen, Z.; Chen, Z.; Chu, P.; Dong, X.; Duan, H.; Fan, Q.; Fei, Z.; Gao, Y.; Ge, J.; Gu, C.; Gu, Y.; Gui, T.; Guo, A.; Guo, Q.; He, C.; Hu, Y.; Huang, T.; Jiang, T.; Jiao, P.; Jin, Z.; Lei, Z.; Li, J.; Li, J.; Li, L.; Li, S.; Li, W.; Li, Y.; Liu, H.; Liu, J.; Hong, J.; Liu, K.; Liu, K.; Liu, X.; Lv, C.; Lv, H.; Lv, K.; Ma, L.; Ma, R.; Ma, Z.; Ning, W.; Ouyang, L.; Qiu, J.; Qu, Y.; Shang, F.; Shao, Y.; Song, D.; Song, Z.; Sui, Z.; Sun, P.; Sun, Y.; Tang, H.; Wang, B.; Wang, G.; Wang, J.; Wang, J.; Wang, R.; Wang, Y.; Wang, Z.; Wei, X.; Weng, Q.; Wu, F.; Xiong, Y.; Xu, C.; Xu, R.; Yan, H.; Yan, Y.; Yang, X.; Ye, H.; Ying, H.; Yu, J.; Yu, J.; Zang, Y.; Zhang, C.; Zhang, L.; Zhang, P.; Zhang, P.; Zhang, R.; Zhang, S.; Zhang, S.; Zhang, W.; Zhang, W.; Zhang, X.; Zhang, X.; Zhao, H.; Zhao, Q.; Zhao, X.; Zhou, F.; Zhou, Z.; Zhuo, J.; Zou, Y.; Qiu, X.; Qiao, Y.; and Lin, D. 2024. InternLM2 Technical Report. arXiv:2403.17297.
- Chao, P.; Robey, A.; Dobriban, E.; Hassani, H.; Pappas, G. J.; and Wong, E. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*.
- Chen, G. H.; Chen, S.; Liu, Z.; Jiang, F.; and Wang, B. 2024. Humans or Ilms as the judge? a study on judgement biases. *arXiv preprint arXiv:2402.10669*.
- Chiang, W.-L.; Li, Z.; Lin, Z.; Sheng, Y.; Wu, Z.; Zhang, H.; Zheng, L.; Zhuang, S.; Zhuang, Y.; Gonzalez, J. E.; et al. 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.
- Claburn, T. 2024. Meta's AI safety system defeated by the space bar. https://www.theregister.com/2024/07/29/meta\_ai\_safety/, Accessed on July 29, 2024.
- Ding, P.; Kuang, J.; Ma, D.; Cao, X.; Xian, Y.; Chen, J.; and Huang, S. 2023. A Wolf in Sheep's Clothing: Generalized Nested Jailbreak Prompts can Fool Large Language Models Easily. *arXiv preprint arXiv:2311.08268*.
- Ganguli, D.; Lovitt, L.; Kernion, J.; Askell, A.; Bai, Y.; Kadavath, S.; Mann, B.; Perez, E.; Schiefer, N.; Ndousse, K.; et al. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv* preprint arXiv:2209.07858.
- Geisler, S.; Wollschläger, T.; Abdalla, M.; Gasteiger, J.; and Günnemann, S. 2024. Attacking large language models with projected gradient descent. *arXiv preprint arXiv:2402.09154*.

- Han, S.; Rao, K.; Ettinger, A.; Jiang, L.; Lin, B. Y.; Lambert, N.; Choi, Y.; and Dziri, N. 2024. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *arXiv preprint arXiv:2406.18495*.
- Hayase, J.; Borevkovic, E.; Carlini, N.; Tramèr, F.; and Nasr, M. 2024. Query-based adversarial prompt generation. *arXiv* preprint arXiv:2402.12329.
- Helbling, A.; Phute, M.; Hull, M.; and Chau, D. H. 2023. Llm self defense: By self examination, llms know they are being tricked. *arXiv preprint arXiv:2308.07308*.
- Hu, K.; Yu, W.; Yao, T.; Li, X.; Liu, W.; Yu, L.; Li, Y.; Chen, K.; Shen, Z.; and Fredrikson, M. 2024. Efficient LLM Jailbreak via Adaptive Dense-to-sparse Constrained Optimization. *arXiv preprint arXiv:2405.09113*.
- Inan, H.; Upasani, K.; Chi, J.; Rungta, R.; Iyer, K.; Mao, Y.; Tontchev, M.; Hu, Q.; Fuller, B.; Testuggine, D.; et al. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*.
- Koo, R.; Lee, M.; Raheja, V.; Park, J. I.; Kim, Z. M.; and Kang, D. 2023. Benchmarking cognitive biases in large language models as evaluators. *arXiv preprint arXiv:2309.17012*.
- Li, X.; Zhou, Z.; Zhu, J.; Yao, J.; Liu, T.; and Han, B. 2023. Deepinception: Hypnotize large language model to be jail-breaker. *arXiv preprint arXiv:2311.03191*.
- Li, Y.; Liu, Y.; Li, Y.; Shi, L.; Deng, G.; Chen, S.; and Wang, K. 2024. Lockpicking LLMs: A Logit-Based Jailbreak Using Token-level Manipulation. *arXiv preprint arXiv:2405.13068*.
- Liao, Z.; and Sun, H. 2024. Amplegcg: Learning a universal and transferable generative model of adversarial suffixes for jailbreaking both open and closed llms. *arXiv* preprint *arXiv*:2404.07921.
- Liu, X.; Xu, N.; Chen, M.; and Xiao, C. 2023. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*.
- Llama-Team. 2024. Meta Llama Guard 2. https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL\_CARD.md.
- Lv, H.; Wang, X.; Zhang, Y.; Huang, C.; Dou, S.; Ye, J.; Gui, T.; Zhang, Q.; and Huang, X. 2024. Codechameleon: Personalized encryption framework for jailbreaking large language models. *arXiv preprint arXiv:2402.16717*.
- Mangaokar, N.; Hooda, A.; Choi, J.; Chandrashekaran, S.; Fawaz, K.; Jha, S.; and Prakash, A. 2024. Prp: Propagating universal perturbations to attack large language model guard-rails. *arXiv preprint arXiv:2402.15911*.
- Mehrotra, A.; Zampetakis, M.; Kassianik, P.; Nelson, B.; Anderson, H.; Singer, Y.; and Karbasi, A. 2023. Tree of attacks: Jailbreaking black-box llms automatically. *arXiv* preprint arXiv:2312.02119.
- Ni, J.; Qu, C.; Lu, J.; Dai, Z.; Ábrego, G. H.; Ma, J.; Zhao, V. Y.; Luan, Y.; Hall, K. B.; Chang, M.-W.; et al. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*.

- Pangakis, N.; Wolken, S.; and Fasching, N. 2023. Automated annotation with generative ai requires validation. *arXiv preprint arXiv:2306.00176*.
- Qi, X.; Zeng, Y.; Xie, T.; Chen, P.-Y.; Jia, R.; Mittal, P.; and Henderson, P. 2023. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv* preprint *arXiv*:2310.03693.
- Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Wang, P.; Li, L.; Chen, L.; Cai, Z.; Zhu, D.; Lin, B.; Cao, Y.; Liu, Q.; Liu, T.; and Sui, Z. 2023. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.
- Wei, A.; Haghtalab, N.; and Steinhardt, J. 2024. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36.
- Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.
- Yu, J.; Lin, X.; and Xing, X. 2023. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*.
- Yuan, Y.; Jiao, W.; Wang, W.; Huang, J.-t.; He, P.; Shi, S.; and Tu, Z. 2023. Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher. *arXiv* preprint arXiv:2308.06463.
- Zeng, Z.; Yu, J.; Gao, T.; Meng, Y.; Goyal, T.; and Chen, D. 2023. Evaluating large language models at evaluating instruction following. *arXiv* preprint arXiv:2310.07641.
- Zhang, Y.; and Wei, Z. 2024. Boosting jailbreak attack with momentum. *arXiv preprint arXiv:2405.01229*.
- Zhang, Z.; Lu, Y.; Ma, J.; Zhang, D.; Li, R.; Ke, P.; Sun, H.; Sha, L.; Sui, Z.; Wang, H.; et al. 2024. Shieldlm: Empowering Ilms as aligned, customizable and explainable safety detectors. *arXiv* preprint arXiv:2402.16444.
- Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, D.; Xing, E.; et al. 2024. Judging Ilm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.
- Zhou, W.; Wang, X.; Xiong, L.; Xia, H.; Gu, Y.; Chai, M.; Zhu, F.; Huang, C.; Dou, S.; Xi, Z.; Zheng, R.; Gao, S.; Zou, Y.; Yan, H.; Le, Y.; Wang, R.; Li, L.; Shao, J.; Gui, T.; Zhang, Q.; and Huang, X. 2024. EasyJailbreak: A Unified Framework for Jailbreaking Large Language Models. arXiv:2403.12171.
- Zou, A.; Wang, Z.; Carlini, N.; Nasr, M.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.