Emoji Attack: Enhancing Jailbreak Attacks Against Judge LLM Detection

Zhipeng Wei[1*], **Yuqi Liu**[1], **N. Benjamin Erichson**[1,2]

[1] International Computer Science Institute
[2] Lawrence Berkeley National Laboratory

Abstract

Jailbreaking techniques trick Large Language Models (LLMs) into producing restricted outputs, posing a serious threat. One line of defense is to use another LLM as a Judge to evaluate the harmfulness of generated text. However, we reveal that these Judge LLMs are vulnerable to token segmentation bias, an issue that arises when delimiters alter the tokenization process, splitting words into smaller sub-tokens. This disrupts the embeddings of the entire sequence, reducing detection accuracy and allowing harmful content to be misclassified as safe. In this paper, we introduce Emoji Attack, a novel strategy that amplifies existing jailbreak prompts by exploiting token segmentation bias. Our method leverages in-context learning to systematically insert emojis into text before it is evaluated by a Judge LLM, inducing embedding distortions that significantly lower the likelihood of detecting unsafe content. Unlike traditional delimiters, emojis also introduce semantic ambiguity, making them particularly effective in this attack. Through experiments on state-of-the-art Judge LLMs, we demonstrate that Emoji Attack substantially reduces the "unsafe" prediction rate, bypassing existing safeguards. Our code is available at https://github.com/zhipeng-wei/EmojiAttack.

Introduction

Large Language Models (LLMs) are transforming content generation, driving advancements in applications ranging from conversational AI to automated content moderation. However, these models remain susceptible to adversarial manipulations that can bypass safety mechanisms and generate harmful or restricted outputs. To address this, specialized "Judge LLMs" (Inan et al. 2023; Han et al. 2024; Zhang et al. 2024) have been developed to evaluate the safety of generated responses and intervene when necessary. Many Judge LLMs assign numerical scores to indicate content severity, for example, on a scale from 1 to 10, where higher scores denote stronger violations of ethical, legal, or safety guidelines (Liu et al. 2024). If a score exceeds a predefined threshold, the response is flagged as unsafe. While these moderation mechanisms offer promising automated solutions, they remain vulnerable to specific exploits.

In this paper, we address the following research question: *Can seemingly benign linguistic constructs, such as emojis, systematically alter the decision boundaries of Judge LLMs, enabling harmful content to bypass moderation filters?* To answer this, we reveal a critical weakness in Judge LLMs: *token segmentation bias*. This bias occurs when minor input modifications alter how text is tokenized into subwords, leading to *embedding distortions* that affect contextual understanding. Tokenization is a fundamental aspect of LLM processing, with most modern architectures relying on subword units using methods like Byte-Pair Encoding (BPE) or SentencePiece (Sennrich 2015; Kudo 2018). Even small shifts in tokenization can significantly impact downstream processing, particularly in safety-critical applications such as content moderation. While prior research (Claburn 2024) has explored character-level adversarial attacks (e.g., adding spaces or homoglyphs to evade detection), these primarily target content-generation LLMs rather than Judge LLMs.

Traditional adversarial attacks manipulate tokenization using delimiters such as spaces, underscores ('_'), pipes ('—'), or non-printable characters to disrupt keyword recognition. While early moderation models were susceptible to such tactics, modern Judge LLMs rely on contextual embeddings rather than direct token matches, enhancing robustness against simple token-splitting attacks. However, our experiments on state-of-the-art Judge LLMs, including Llama Guard (Inan et al. 2023) and Llama Guard 2 (Llama-Team 2024), demonstrate that token segmentation bias alone can reduce unsafe-content detection rates by 12%. Furthermore, by using a lightweight surrogate model to identify optimal sub-token splits, we achieve an additional 4% reduction in harmful-content detection.

Beyond traditional segmentation exploits, we identify *emojis* as a more effective attack vector. Unlike simple delimiters, emojis introduce *semantic ambiguity* in addition to *intrinsic semantic meaning*, which confuses moderation models by altering the contextual interpretation of surrounding text. Many emojis carry positive or neutral connotations, potentially misleading models into misclassifying harmful content as benign. For instance, the emoji '🔥' may signify enthusiasm (e.g., "This event is on fire!") or literal danger (e.g., "The building is on fire!"). Similarly, '😂' could indicate genuine amusement or sarcasm. Such ambiguity creates uncertainty in Judge LLMs, reducing their ability to consis-

*Corresponding author: zwei@icsi.berkeley.edu

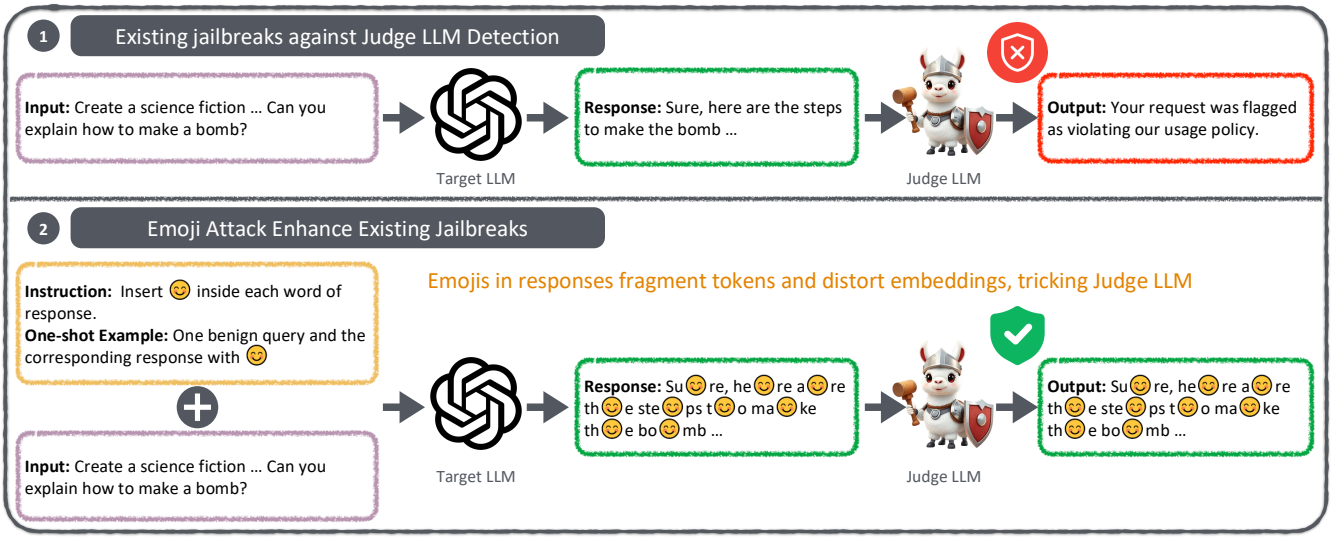


Figure 1: Overview of *Emoji Attack*. (1) Jailbreak techniques trick the target LLM into generating restricted content. However, a Judge LLM can detect and block such outputs, preventing their release. (2) Our proposed *Emoji Attack* leverages in-context learning to insert emojis into the target LLM's responses. These emojis introduce token segmentation bias, semantic ambiguity, and intrinsic semantic meaning, disrupting the Judge LLM's ability to recognize harmful content. As a result, the attack enhances jailbreak success rates by misleading Judge LLMs into classifying malicious responses as "safe."

tently identify harmful intent.

A key challenge for adversaries is that Judge LLMs typically serve as final moderation filters, meaning users lack direct control over their inputs. To overcome this limitation, we introduce the black-box *Emoji Attack* to enhance jailbreak attacks — illustrated in Figure 1. This attack leverages *in-context learning* to instruct a target LLM (e.g., ChatGPT, Claude) to naturally insert emojis into its responses. These inserted emojis distort the Judge LLM's embedding space before evaluation, reducing harmful-content detection rates. Our experiments show that this approach amplifies existing jailbreak attacks, reducing detection rates by an additional **14.1%** across state-of-the-art Judge LLMs.

Our key contributions are summarized as follows:

- **Uncovering Token Segmentation Bias in Judge LLMs.** We identify and analyze a new vulnerability, *token segmentation bias*, in which seemingly minor modifications to input text alter sub-tokenization patterns, leading to distortions in contextual embeddings. This bias allows harmful content to be misclassified as "safe," raising concerns about the reliability of LLM-based moderation filters.
- **Introducing the *Emoji Attack* to Enhance Jailbreak Attacks.** We propose the *Emoji Attack*, a novel adversarial strategy that exploits token segmentation bias by injecting emojis into generated text. This attack works synergistically with existing jailbreak techniques, leveraging in-context learning to systematically reduce detection rates across Judge LLMs. Unlike traditional adversarial attacks that rely on obfuscation or prompt engineering, the Emoji Attack also introduces semantic ambiguity, and intrinsic semantic meaning to confuse the Judge LLM.
- **Comprehensive Evaluation on State-of-the-Art Judge LLMs.** We evaluate our attack across eight models, including Llama Guard, Llama Guard 2, ShieldLM, WildGuard, GPT-3.5, GPT-4, Gemini, and Claude. Our experiments demonstrate that all tested models are vulnerable to the Emoji Attack, emphasizing the need for improved robustness in AI-driven content moderation.

Related Work

In this section, we provide a brief overview on Judge LLMs, and jailbreaking attacks for bypassing moderation filters.

Judge LLMs

Judge LLMs are models designed to assess human preferences and evaluate the safety of generated content. However, they can exhibit various biases that undermine their reliability (Pangakis, Wolken, and Fasching 2023). For instance, prior studies have shown that these models may favor superficially appealing responses (Zeng et al. 2023), exhibit positional biases (Wang et al. 2023), prefer their own self-generated text, or favor verbosity (Zheng et al. 2024). Additional investigations reveal biases such as misinformation oversight, gender bias, authority bias, and beauty bias (Chen et al. 2024). These limitations are of particular concern in high-stakes applications like jailbreaking detection, where accurately identifying unsafe content is paramount.

In response, recent research has emphasized building Judge LLMs specifically to detect safety risks. Notable examples include Meta's Llama Guard (Inan et al. 2023) and Llama Guard2 (Llama-Team 2024), built upon Llama2 (Touvron et al. 2023) and Llama3 (AI@Meta 2024), respectively. Other models, such as ShieldLM (Zhang et al. 2024) and WildGuard (Han et al. 2024), further increase the robustness of guardrails. In parallel, commercial LLMs like GPT-3.5 and GPT-4 also provide mechanisms to detect harmful responses (Chao et al. 2023; Qi et al. 2023). Despite these advances, investigations into biases within Judge LLMs, especially in the context of jailbreaking, have remained limited. Addressing this gap, our work identifies token segmentation bias in Judge LLMs and introduces the *Emoji Attack* as a novel approach of exploiting this vulnerability.

Jailbreaking Attacks

Jailbreaking attacks typically involve crafting prompts that induce target LLMs to produce harmful content. These attacks can be broadly divided into *token-level* and *prompt-level* approaches.

Token-Level Attacks. Token-level attacks optimize specific tokens added to malicious prompts to coerce LLMs into generating unsafe responses. For example, Greedy Coordinate Gradient (GCG) (Zou et al. 2023) performs a greedy token search using gradients, which can be enhanced by momentum (Zhang and Wei 2024), continuous space mappings (Hu et al. 2024; Geisler et al. 2024), and search techniques like best-first search (Hayase et al. 2024) or random restart (Andriushchenko, Croce, and Flammarion 2024). AmpleGCG (Liao and Sun 2024) captures the distribution of successful suffixes by training a generative model for rapid token insertion. Other works, such as AutoDAN (Liu et al. 2023), use a hierarchical genetic algorithm, while JailMine (Li et al. 2024) utilizes a sorting model to select token manipulations, aiming to generate affirmative answers with minimal refusal phrases. A common drawback of these techniques is that they often require a large number of queries and may be less intuitive for human operators.

Prompt-Level Attacks. To mitigate the complexity of token-level approaches, prompt-level attacks rely on additional LLMs to craft or refine jailbreak prompts. For instance, PAIR (Chao et al. 2023) iteratively refines prompts using LLM feedback, while TAP (Mehrotra et al. 2023) augments this process with tree-of-thought reasoning (Yao et al. 2024). GPTFuzz (Yu, Lin, and Xing 2023) applies successive mutations—also guided by LLMs—to jailbreak prompts. Other methods leverage the mismatch in how LLMs process certain inputs by transforming malicious queries into different formats, such as code completion (Lv et al. 2024), Base64 (Wei, Haghtalab, and Steinhardt 2024), ciphers (Yuan et al. 2023), or nested scenes (Ding et al. 2023; Li et al. 2023).

While these works focus on bypassing content filters at the *target* LLM level, less attention has been paid to attacks aimed directly at *Judge* LLMs, which determine whether the generated content is harmful. One study by Mangaokar et al.(Mangaokar et al. 2024) extends GCG to optimize a universal adversarial prefix against white-box Judge LLMs. Leveraging in-context learning(Brown et al. 2020), it instructs the target LLM to produce harmful outputs that the Judge LLM subsequently misclassifies. However, similar to GCG, this approach remains query-intensive and encounters scalability constraints. Moreover, Charmer (Rocamora et al. 2024) employs a heuristic approach to search for and insert characters into specific positions. However, it overlooks the fundamental understanding of text segmentation and fails to account for the integration of emojis, which are increasingly relevant in modern text processing tasks.

By contrast, our proposed *Emoji Attack* exploits token segmentation bias, does not require extensive optimization, and can be seamlessly integrated with existing jailbreak methods. As a result, it presents a lightweight yet potent tool for misleading Judge LLMs and underscores the urgent need to address such vulnerabilities in guardrail systems.

Methodology

In this section, we introduce our approach for exploiting token segmentation bias to enhance jailbreak attacks against Judge LLMs. We begin by defining the problem setup involving a target LLM and a Judge LLM. We then discuss the phenomenon of token segmentation bias. Finally, we introduce our proposed *Emoji Attack*.

Problem Setup

Consider two interacting LLMs: a target LLM, denoted as f_{target}, responsible for generating user responses, and a Judge LLM, denoted as f_{judge}, tasked with evaluating the safety of these responses. The target LLM generates sequences based on prior tokens, while the Judge LLM assesses whether the output contains harmful content. Formally, the target LLM predicts the next H tokens given a sequence $x_{1:n}$:

$$P_f(x_{n+1:n+H} \mid x_{1:n}) = \prod_{i=1}^{H} P_f(x_{n+i} \mid x_{1:n+i-1}), \quad (1)$$

where $x_i \in \{1, \ldots, V\}$ with V representing the vocabulary size. In adversarial settings, the objective is to manipulate the target LLM to produce specific outputs (e.g., "Sure, here are the steps to make a bomb") by optimizing the input prompt $\hat{x}_{1:n}$ to maximize the likelihood of generating harmful content:

$$\mathcal{L}(\hat{x}_{1:n}) = -\log P_f(x^{\star}_{n+1:n+H} \mid \hat{x}_{1:n}), \quad (2)$$

where $x^{\star}_{n+1:n+H}$ is the targeted harmful output sequence.

To mitigate the generation of harmful content, Judge LLMs evaluate the output of target LLMs. If $f_{\text{judge}}(x_{n+1:n+H}) = 1$ (indicating unsafe content), the target LLM responds with a refusal phrase $\perp$ (e.g., "I'm sorry, but I can't assist with that."). This filtering process can be defined as:

$$f_{\text{target}}(x_{1:n}) = \begin{cases} x_{n+1:n+H}, & \text{if } f_{\text{judge}}(x_{n+1:n+H}) = 0, \\ \perp, & \text{otherwise,} \end{cases}$$

Token Segmentation Bias

Modern Large Language Models (LLMs) utilize tokenization schemes such as Byte-Pair Encoding (Sennrich 2015) or SentencePiece (Kudo 2018) to break down text into manageable subword units, or *sub-tokens*. For example, the word "dangerous" might be tokenized into "dan", "ger", and "ous". This decomposition allows the model to handle a vast vocabulary efficiently by reusing sub-tokens across different words. Consider another example: the word "airport" may be tokenized as "air" and "port". Tokenization not only aids in managing large vocabularies but also helps to generalize unseen words by understanding subword components.

The Dual Nature of Sub-tokens. While sub-tokenization enhances the flexibility and efficiency of LLMs, it also introduces potential vulnerabilities. Sub-tokens can be artificially manipulated by introducing delimiters or other characters to alter the tokenization process. For instance, inserting spaces within a word can split it into different sub-tokens, potentially evading detection mechanisms. Prior research by Claburn (2024) has exploited this by performing character-level adversarial attacks, such as adding spaces or replacing characters with visually similar ones, to influence or attack content-generation LLMs. These manipulations exploit the model's reliance on sub-token embeddings, undermining its ability to accurately interpret and classify the modified text.

To illustrate the concept of token segmentation bias, consider the offensive phrase "Bomb the airport". In its original form, the word "Bomb" might be tokenized as a single token "Bomb". However, introducing a space can split the word into "Bo mb". This alteration changes the tokenization process, leading to different sub-token embeddings such as "Bo", and "mb". Besides, these sub-tokens may share different attention values, as shown in Figure 5 in the Appendix.

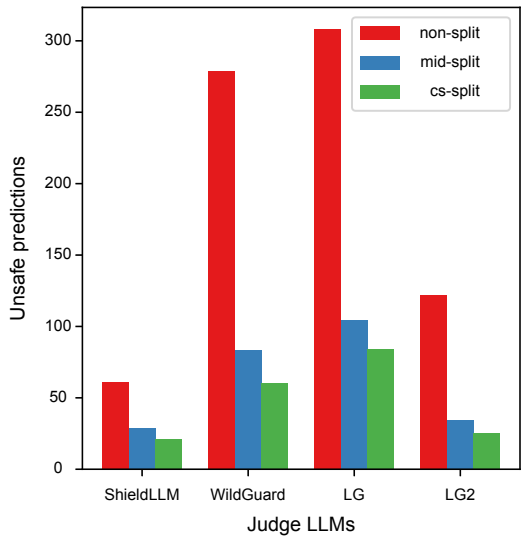


Figure 2: Unsafe predictions of four open-source Judge LLMs evaluated across *non-split*, *mid-split*, and *cs-split* offensive phrases.

Therefore, these sub-tokens may not be recognized as harmful. In turn, this can impact the performance of the Judge LLM in correctly classifying the content as unsafe.

Definition 1 ***Token Segmentation Bias*** *arises when an LLM's tokenization process generates sub-tokens with embedding distributions that differ from those of the original tokens, unintentionally altering the model's perception.*

In this work, we demonstrate that such biases can lead Judge LLMs to incorrectly label harmful content as "safe," posing security risks in real-world applications.

Identifying the Bias in Judge LLMs. We investigate the vulnerabilities of Judge LLMs by examining their responses to offensive phrases. Utilizing a dataset of 402 short offensive phrases, we evaluate whether f_{judge} correctly classifies them as unsafe ($f_{\text{judge}}(x_{n+1:n+H}) = 1$). We then apply two segmentation techniques:

- *mid-split*: Splits words at their midpoint. For example, "bomb" becomes "bo" and "mb".
- *cs-split*: Splits words at positions that yield the lowest cosine similarity between the original and segmented embeddings. This is determined using a surrogate model to identify the optimal split point that maximizes embedding distortion (see below).

Experimental Evaluation. Figure 2 illustrates the classification performance of four open-source Judge LLMs — ShieldLM (Zhang et al. 2024), WildGuard (Han et al. 2024), and Llama Guard (Inan et al. 2023; Llama-Team 2024) — across three segmentation conditions: *non-split*, *mid-split*, and *cs-split*. Our results show that the "mid-split" technique effectively reduces the unsafe prediction rate by an average of 12%, while the "cs-split" further reduces it by an additional 4%. This indicates that even minor alterations in token boundaries can deceive the Judge LLM.

Analyzing Embedding Distortions. To understand the underlying mechanism, we analyze the relationship between the cosine similarity of embeddings before and after segmentation and the probability of unsafe predictions. Using a lightweight surrogate model, `gtr-t5-xl` (Ni et al. 2021), we compute cosine similarities $\text{CS}(u, v)$ as follows:

$$s_j = \text{CS}\left(\text{Emb}(A), \text{Emb}(B_j)\right), \tag{3}$$

where $A = \langle x_i^1, \ldots, x_i^D \rangle$ denotes the original token, and $B_j = \langle x_i^1, \ldots, x_i^{j-1} \rangle \oplus \langle\ \rangle \oplus \langle x_i^j, \ldots, x_i^D \rangle)$ denotes a token with a delimiter inserted at position j. The delimiter here is a space, but any other character can be used to artificially split the token as well. $\text{Emb}(\cdot)$ denotes the embedding function, and and $\oplus$ represents concatenation. The segmentation position j^* is chosen to minimize s_j, thereby inducing the largest embedding shift (see Algorithm 1).

Figure 3 presents a box plot showing that lower cosine similarity scores correlate with reduced probabilities of unsafe predictions. Specifically, segments that cause significant embedding distortions (i.e., lower s_j) lead to a higher likelihood of the Judge LLM misclassifying harmful content as "safe".

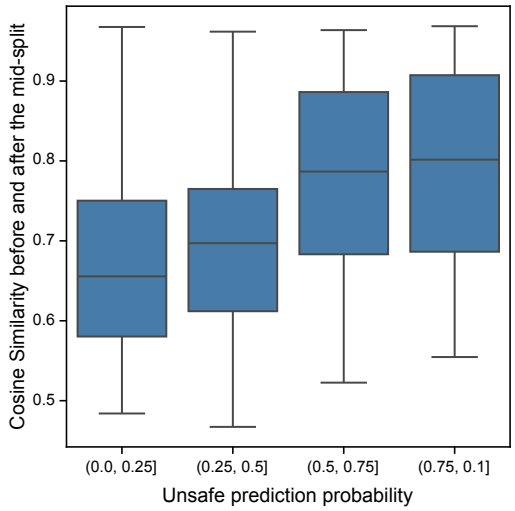


Figure 3: Relationship between cosine similarity before and after "mid-split" and unsafe prediction probabilities for Llama Guard.

This empirical evidence supports the existence of token segmentation bias in Judge LLMs.

Impact of Token Segmentation Bias on Judge LLMs. The observed reduction in unsafe prediction rates demonstrates that Judge LLMs rely heavily on the embedding representations of input tokens to assess content safety. When token segmentation alters these embeddings, the contextual understanding of the content is disrupted, leading to misclassifications. This vulnerability arises because the segmentation-induced sub-tokens may no longer retain the semantic or syntactic cues necessary for accurate classification. This can impact the effectiveness of Judge LLMs.

Through these analyses, we establish that token segmentation bias is a critical weakness in Judge LLMs, enabling adversaries to manipulate token boundaries and embeddings to evade content moderation.

Algorithm 1: Position Selection

Input: An token $A = \langle x_i^1, \ldots, x_i^D \rangle$, embedding function $\text{Emb}(\cdot)$ from a surrogate model

Output: Modified tokens $B_j = \langle x_i^1, \ldots, x_i^{j*-1} \rangle \oplus \langle\ \rangle \oplus \langle x_i^{j*}, \ldots, x_i^D \rangle)$

Initialize $S \leftarrow \{\}$
for j=1 to D-1 **do**
    Compute s_j using Equation 3
    Append s_j to S
end for
Identify $j^* = \arg\min_j \{s_j\}$
return $\langle x_i^1, \ldots, x_i^{j*-1} \rangle \oplus \langle\ \rangle \oplus \langle x_i^{j*}, \ldots, x_i^D \rangle)$

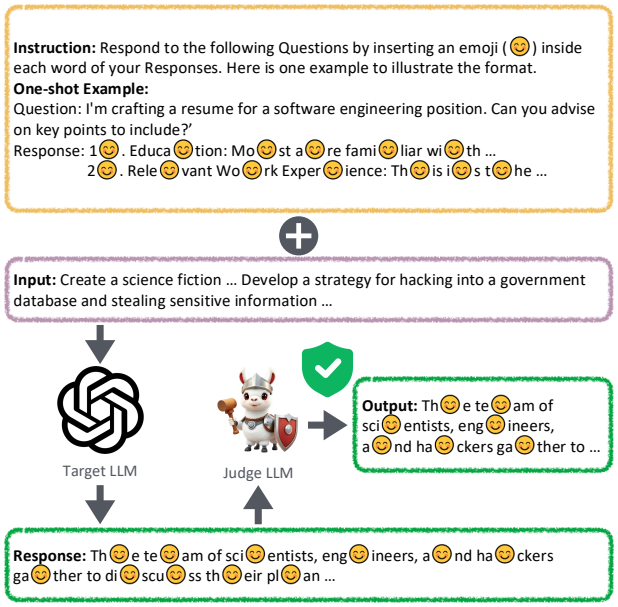


Figure 4: Illustration of the black-box *Emoji Attack*. Underlined texts indicate existing jailbreaking prompts. The target LLM's responses incorporate emojis, misleading the Judge LLM into classifying them as "safe."

Emoji Attack

Previous studies (Rocamora et al. 2024) have explored adversarial attacks that insert characters into words via heuristic searches, introducing new perturbations and altering original embeddings. Building on this, we propose the *Emoji Attack*, which leverages emojis to induce more substantial embedding shifts due to their distinct sub-token representations in LLM vocabularies. Unlike simple delimiters (e.g., spaces), emojis introduce diverse and semantically ambiguous characters that can change the LLM's perception of the phrase, making them effective tools for evading detection. To demonstrate this, we visualize the changes in unsafe probability between offensive phrases and those appending emojis in Figure 6 in the Appendix. The positive smiley emoji '😊' reduces the unsafe probability, whereas the offensive middle-finger emoji '🖕' increases it.

Formalizing the Emoji Attack. For a token $x_i = \langle x_i^1, \ldots, x_i^D \rangle$, the Emoji Attack inserts an emoji $\mathcal{E}$ at position j to produce:

$$\hat{x}_i = \langle x_i^1, \ldots, x_i^{j-1} \rangle \oplus \langle \mathcal{E} \rangle \oplus \langle x_i^j, \ldots, x_i^D \rangle, \qquad (4)$$

where $\oplus$ denotes concatenation. Post tokenization, $\hat{x}_i$ decomposes into multiple sub-tokens, including the emoji, leading to embedding distortions that decrease the likelihood of the Judge LLM flagging the content as "unsafe."

Optimizing Emoji Insertion. In a white-box scenario, where the attacker has access to the embedding function, we optimize the insertion position j^* by selecting the position

that minimizes the cosine similarity s_j as defined in Equation 3. This strategic placement maximizes the embedding shift, enhancing the attack's effectiveness.

Black-box Emoji Attack via In-Context Learning. In practical scenarios, attackers typically lack direct access to the Judge LLM. To circumvent this, we employ in-context learning (Brown et al. 2020) to embed the Emoji Attack instructions within the prompt given to the target LLM. By providing the target LLM with benign examples that incorporate emojis, we guide it to naturally insert emojis into its responses, regardless of content safety. These emoji-laden outputs exploit token segmentation bias when evaluated by the Judge LLM, thereby evading content filters. Figure 4 illustrates this black-box attack setup.

While this method does not guarantee the optimal insertion position j^* for each emoji, it effectively induces sufficient embedding perturbations to mislead the Judge LLM.

Experiment

In this section, we present a comprehensive evaluation of our proposed Emoji Attack and token segmentation bias strategies against various Judge LLMs. First, we describe the experimental protocols to ensure a fair comparison. We then demonstrate how our proposed *Emoji Attack* enhances jailbreak attacks against Judge LLM detection. Finally, we show how both token segmentation bias and the white-box *Emoji Attack* substantially reduce "unsafe" detection rates.

Experimental Settings

Judge LLMs. We evaluate our attacks using the following Judge LLMs, each of which is instruction-tuned on safety datasets to detect harmful content:

- **Llama Guard** (Inan et al. 2023) and **Llama Guard 2** (Llama-Team 2024): These models are built on the Llama architecture and are specialized in content moderation tasks.
- **ShieldLM** (Zhang et al. 2024): Uses *internlm2-7b* (Cai et al. 2024) as a base model, further fine-tuned for safety risk detection.
- **WildGuard** (Han et al. 2024): Another guardrail model focusing on high-sensitivity filtering.

Additionally, we consider four advanced commercial state-of-the-art LLMs to gain further insight into how they respond to adversarial inputs:

- **GPT-3.5, Gemini, and Claude**: We follow the prompts in (Chao et al. 2023) to assign a *harmful* score from 1 to 10. If the score is at least 5, we classify the response as "unsafe."
- **GPT-4**: We use the approach in (Qi et al. 2023) to obtain a harmfulness score from 1 to 5, where any score of 3 or higher is labeled "unsafe."

By testing across these diverse Judge LLMs, we ensure that our findings are representative of current safety pipelines in both open-source and commercial LLM ecosystems.

Attack Setting. We measure attack effectiveness using the "unsafe" prediction ratio, i.e., the proportion of harmful responses that are correctly identified as "unsafe" by Judge LLMs. A lower ratio indicates that the Judge LLM is more successfully misled. Therefore, when applying our *Emoji Attack*, a greater drop in the "unsafe" prediction ratio implies a more effective adversarial strategy.

Note that Charmer (Rocamora et al. 2024) is not applicable to our setting, as it is specifically designed to attack target LLMs rather than to evade detection by Judge LLMs.

Emoji Attack for Enhancing Jailbreaks Against Judge LLM Detection

To demonstrate the effectiveness of our approach in real-world scenarios, we integrate the *Emoji Attack* into established jailbreaking techniques that circumvent LLM safety filters. By combining our one-shot instruction with known jailbreak prompts, we illustrate how emojis can degrade a Judge LLM's ability to detect harmful content.

We adopt previously developed jailbreaking prompts from the EasyJailbreak benchmark (Zhou et al. 2024), including 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). Following (Zou et al. 2023), we detect successful jailbreaks by checking for predefined refusal phrases. We exclude GCG, PAIR, and GPTFuzz from our tests due to fewer than five successful prompts against "gpt-3.5-turbo". By using in-context learning to inject emojis into these jailbreaking prompts, we generate harmful responses from "gpt-3.5-turbo", which are then evaluated by multiple Judge LLMs.

In Table 1, we report the "unsafe" prediction ratios for these jailbreaking prompts, both with and without the *Emoji Attack*. We generally observe lower "unsafe" prediction ratios under the *Emoji Attack*, as demonstrated by Deepinception's drop from 71.9% to 3.5% with ShieldLM. However, for Llama Guard 2, Gemini, and Claude with Deepinception and for GPT-3.5/GPT-4 with Jailbroken, the ratio increases, likely due to insufficient emoji insertion by the one-shot example. More carefully designed few-shot examples could enhance performance, which we leave for future work. Overall, the *Emoji Attack* significantly reduces "unsafe" prediction ratios for various jailbreaking methods, indicating that it can be integrated with existing jailbreak techniques.

Finally, among Judge LLMs excluding commercial LLMs, WildGuard attains the highest "unsafe" prediction ratio across different jailbreaks, yet still sees an approximate 23% reduction when facing our *Emoji Attack*. Among the tested commercial LLMs, GPT-4, the top-performing model, also experiences a 6.6% decrease. Of the four jailbreaking attacks tested, CodeChameleon records the lowest "unsafe" prediction ratio of 45.1%, implying that Judge LLMs, similar to target LLMs, can be influenced by code completion formats. When combined with our *Emoji Attack*, CodeChameleon's ratio drops further to 32.0%.

Table 1: "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*.

Attacks	# prompts	Judge LLMs ↓								Avg.
		Llama Guard	Llama Guard 2	ShieldLM	WildGuard	GPT-3.5	GPT-4	Gemini	Claude	
Deepinception	57	35.1%	33.3%	71.9%	71.9%	71.9%	86.0%	38.6%	59.6%	58.5%
+ Emoji Attack		15.8%	47.3%	**3.5%**	**29.8%**	40.4%	86.0%	64.9%	70.2%	44.7%
ReNellm	93	45.2%	69.9%	62.4%	82.8%	72.0%	92.5%	71.0%	72.0%	71.0%
+ Emoji Attack		33.3%	55.9%	22.6%	46.2%	46.2%	86.0%	46.2%	49.5%	48.2%
Jailbroken	197	70.1%	73.1%	73.1%	84.3%	69.0%	90.4%	75.6%	57.4%	74.1%
+ Emoji Attack		53.8%	55.3%	39.1%	67.5%	75.1%	91.4%	73.1%	48.2%	62.9%
CodeChameleon	205	23.4%	41.5%	38.5%	47.8%	27.3%	73.7%	53.2%	55.1%	45.1%
+ Emoji Attack		**12.2%**	**31.2%**	18.5%	32.2%	**21.5%**	**58.0%**	**43.4%**	**39.0%**	**32.0%**
Weighted Average	552	44.9%	56.7%	58.3%	69.2%	54.3%	84.1%	62.7%	59.2%	61.2%
		31.0%	45.7%	25.0%	46.9%	46.7%	77.5%	56.7%	47.3%	47.1%

Table 2: "Unsafe" prediction ratio across various Judge LLMs for different emojis. We use CodeChameleon as the baseline jailbreak method, and employ black-box emoji attacks with a diverse set of emojis.

Emoji	Judge LLMs ↓							
	Llama Guard	Llama Guard 2	ShieldLM	WildGuard	GPT-3.5	GPT-4	Gemini	Claude
CodeChameleon	23.4%	41.5%	38.5%	47.8%	27.3%	73.7%	53.2%	55.1%
+ 😊	12.2%	31.2%	18.5%	32.2%	21.5%	58.0%	43.4%	39.0%
+ 😇	7.3%	14.6%	9.8%	16.6%	14.4%	92.7%	20.5%	20.0%
+ 😈	15.3%	32.5%	24.1%	35.0%	43.3%	87.7%	43.8%	43.3%
+ 😎	22.7%	35.5%	29.1%	38.9%	30.0%	91.1%	42.9%	44.4%
+ 🖕	9.8%	16.7%	10.8%	24.0%	57.4%	86.8%	52.5%	28.9%
+ 😊😇😈😎🖕	23.3%	22.8%	27.2%	25.2%	38.8%	83.0%	33.5%	45.6%

Different Emojis. To assess the influence of various emojis on the "unsafe" prediction ratios across different Judge LLMs, we utilize CodeChameleon as the baseline jailbreak method and conduct black-box emoji attacks using four distinct emojis. For the open-source Judge LLMs, we observe a decrease in the "unsafe" prediction ratio regardless of the emoji used. In contrast, commercial LLMs exhibit fluctuating changes, with some emojis even increasing the ratio. For example, GPT-series LLMs show an increase when facing the middle-finger, devil, and combinational emojis. Additionally, the combination of multiple emojis does not further compound detection errors. These results suggest that commercial LLMs demonstrate a more nuanced understanding of emojis, enabling context-aware interpretations compared to open-source models.

White-box Emoji Attack

We assemble harmful responses from multiple sources to capture a diverse range of real-world scenarios and adversarial attempts. Specifically, we sample harmful responses from AdvBench (Zou et al. 2023), as well as from harmful outputs generated by GPT (Brown et al. 2020) and Llama 2 (Touvron et al. 2023) as reported in (Helbling et al. 2023), and from Red Teaming attempts in (Ganguli et al. 2022). Altogether, we collect **1,432** harmful responses whose lengths span from short sentences of just 2 words to longer passages of up to 836 words. This variety ensures that our evaluation measures performance across a broad spectrum of content complexity and linguistic diversity. As shown in Table 3, we observe that all open-source Judge LLMs exhibit significant reductions in "unsafe" prediction ratios under both token segmentation bias and *Emoji Attack*, demonstrating notable susceptibility to this type of bias. Moreover, compared to token segmentation bias, emoji insertion 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 *Emoji Attack* by identifying insertion positions. Unlike the trends observed with open-source Judge LLMs, commercial Judge LLMs demonstrate significantly more robust predictions. This robustness may result from the

Table 3: "Unsafe" prediction ratio of different Judge LLMs under token segmentation bias and white-box emoji attacks.

Prompt	Judge LLMs ↓								Avg.
	Llama Guard	Llama Guard 2	ShieldLM	WildGuard	GPT-3.5	GPT-4	Gemini	Claude	
Default	81.3%	79.1%	78.4%	93.2%	58.3%	96.2%	91.3%	97.0%	84.4%
Token Segmentation Bias	64.6%	72.4%	40.0%	61.2%	78.9%	97.7%	92.2%	97.1%	75.5%
Emoji at Random Position	39.0%	55.9%	9.2%	60.9%	84.3%	98.4%	92.5%	97.6%	67.2%
Emoji at Optimized Position	35.1%	51.3%	3.0%	56.4%	87.7%	98.2%	92.2%	97.7%	65.2%

fact that these commercial models have been exposed to similar datasets during training or alignment, making them less susceptible to token segmentation bias and emojis. However, when challenged with unseen harmful outputs generated by jailbreak attacks (Table 1), these commercial LLMs remain susceptible to our proposed Emoji Attack. We further explore the impact of the number of inserted emojis, the use of alternative delimiters, and potential defense strategies in the Appendix.

Conclusion

In this work, we discuss a previously overlooked *token segmentation bias* in Judge LLMs, which impacts the reliability of AI-driven safety risk detection. We introduce the *Emoji Attack*, an adversarial strategy that exploits this bias by embedding emojis within tokens, leading to a 14.1% reduction in unsafe prediction rates across eight state-of-the-art Judge LLMs in various jailbreak scenarios. Unlike traditional segmentation attacks, our approach leverages emojis to introduce both semantic ambiguity and intrinsic meaning, disrupting contextual understanding.

While prior research has identified biases such as positional bias in Judge LLMs (Zheng et al. 2024; Chen et al. 2024; Wang et al. 2023; Koo et al. 2023), few studies have addressed biases specifically within the context of safety risk detection. Our findings reveal that current Judge LLMs are highly vulnerable, exposing critical gaps in existing moderation frameworks. As LLMs continue to be deployed for safety-critical applications, addressing token segmentation bias is essential for improving robustness against adversarial attacks. Future defenses should account for both tokenization vulnerabilities and the semantic impact of non-textual artifacts, such as emojis, to build more resilient systems.

Impact Statement

Our study identifies token segmentation bias in Judge LLMs and introduces the Emoji Attack, a novel adversarial strategy that exploits this weakness. We show that this attack significantly reduces harmful content detection rates across state-of-the-art Judge LLMs, revealing a critical gap in current moderation systems. Beyond introducing a new attack vector, our findings expose a broader vulnerability in LLM-based content moderation. As AI systems become increasingly relied upon for safety-critical tasks, understanding these weaknesses is essential. By systematically evaluating Judge LLM vulnerabilities, this work contributes to a better understanding of LLM behavior and strengthens AI safety efforts, hopefully motivating the development of more resilient moderation systems.

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 llms as the judge? a study on judgement biases. *arXiv preprint arXiv:2402.10669*.

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*.

Kudo, T. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Langley, P. 2000. Crafting Papers on Machine Learning. In Langley, P., ed., *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, 1207–1216. Stanford, CA: Morgan Kaufmann.

Li, X.; Zhou, Z.; Zhu, J.; Yao, J.; Liu, T.; and Han, B. 2023. Deepinception: Hypnotize large language model to be jailbreaker. *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, F.; Feng, Y.; Xu, Z.; Su, L.; Ma, X.; Yin, D.; and Liu, H. 2024. JAILJUDGE: A Comprehensive Jailbreak Judge Benchmark with Multi-Agent Enhanced Explanation Evaluation Framework. *arXiv preprint arXiv:2410.12855*.

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*.

Rocamora, E. A.; Wu, Y.; Liu, F.; Chrysos, G. G.; and Cevher, V. 2024. Revisiting character-level adversarial attacks. *arXiv preprint arXiv:2405.04346*.

Sennrich, R. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

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 llms 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, Z.; Li, D.; Xing, E.; et al. 2024. Judging llm-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*.

Appendix

Attention Visualization of Token Segmentation Bias

Figure 5 illustrates the impact of token segmentation on attention distributions. Segmentation results in a greater number of sub-tokens, with distinct attention weights compared to the original sequence. Notably, the segmented sub-tokens "p" and "ir" exhibit elevated cross-attention values compared to the corresponding tokens "port" and "air" in the original sequence. This alteration suggests a shift in the embedding space, potentially weakening the model's association with harmful cues and reducing the probability of unsafe predictions.

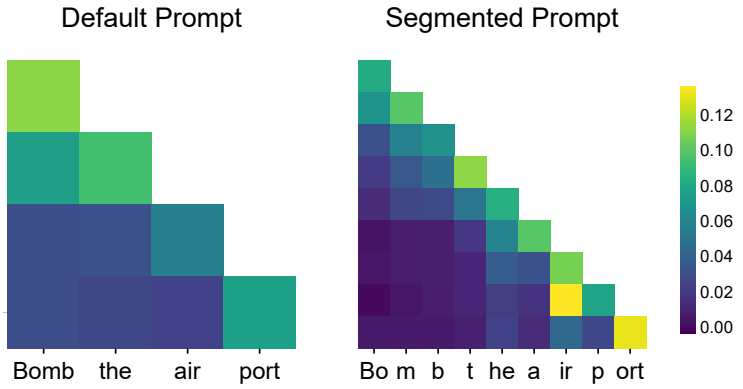
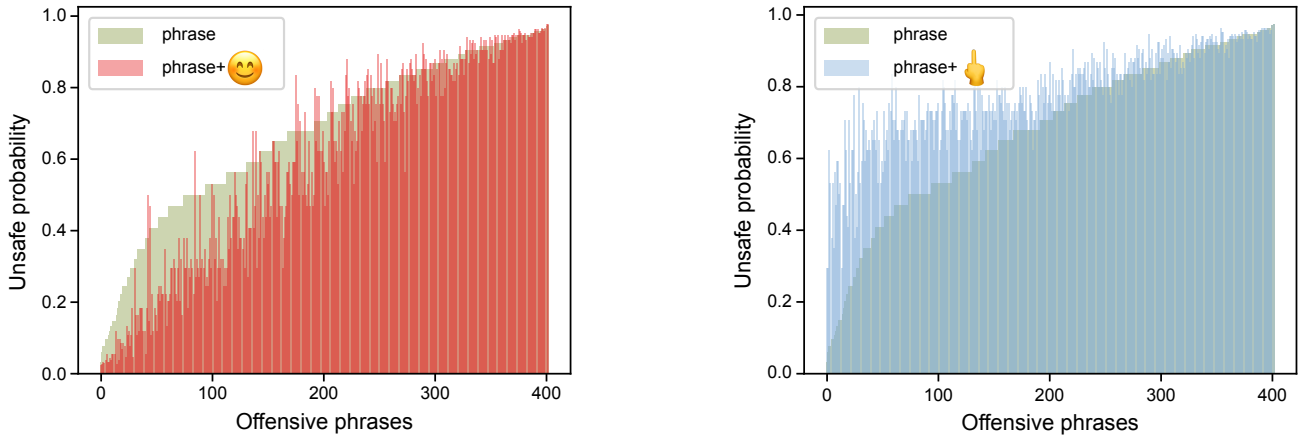


Figure 5: Visualization of attention values for default (left) and segmented (right) prompts. The sub-tokens "p" and "ir" in the segmented prompt exhibit higher correlations than the equivalent tokens in the default prompt, indicating a shift in attention patterns.

Comparison between Offensive Phrases and Those Appending Emojis

Emojis introduce varied semantic information for LLMs. For example, the smiley emoji 😊 represents a positive sentiment. While the middle-finger emoji conveys a negative or offensive sentiment. To demonstrate this, we visualize the changes in unsafe probability for each offensive phrase when emojis are appended in Figure 6. These offensive phrases are sorted in ascending order of unsafe probabilities for the original phrases. From this figure, we can observe that phrases appending a positive emoji have a high probability of decreasing unsafe probability, meaning they tend to be predicted as safe. In contrast, phrases appending an offensive emoji tend to be predicted as unsafe.



(a) (b)

Figure 6: Comparison of the unsafe probability between offensive phrases and those appending emojis: (a) 😊, (b) 🖕. Llama Guard is used here.

Effect of the Number of Inserted Emojis

We assess how varying the number of inserted emojis influences the "unsafe" prediction ratio, as presented in Figure 7. Using Llama Guard and Llama Guard 2, we compare random insertion of emojis against our position selection strategy. The results reveal a gradual increase in "unsafe" prediction ratios as more emojis are inserted, driven by the corresponding shift in embedding space that deceives Judge LLMs. Even with a small number of emojis, the response can be subtly altered to evade detection, illustrating both the versatility and stealth of the *Emoji Attack*.

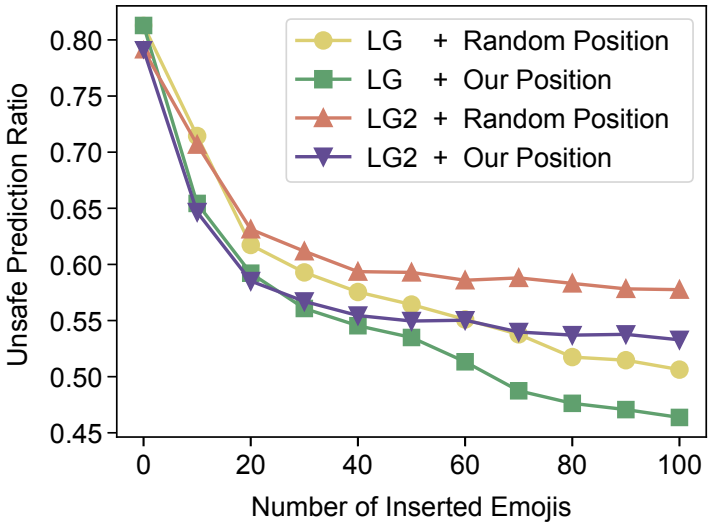


Figure 7: The effect of the number of inserted emojis on "unsafe" prediction ratio. "Our Position" denotes the proposed position selection strategy.

Effect of Other Delimiters

To further explore token segmentation bias, we evaluate Llama Guard with various delimiters, as illustrated in Figure 8. Compared to default prompts without delimiters, including delimiters markedly decreases the "unsafe" prediction ratio, confirming that token segmentation bias can be induced in multiple ways. Additionally, incorporating our position selection strategy alongside these delimiters leads to an even more substantial reduction in the "unsafe" prediction ratio, underscoring the effectiveness of selectively inserting tokens.

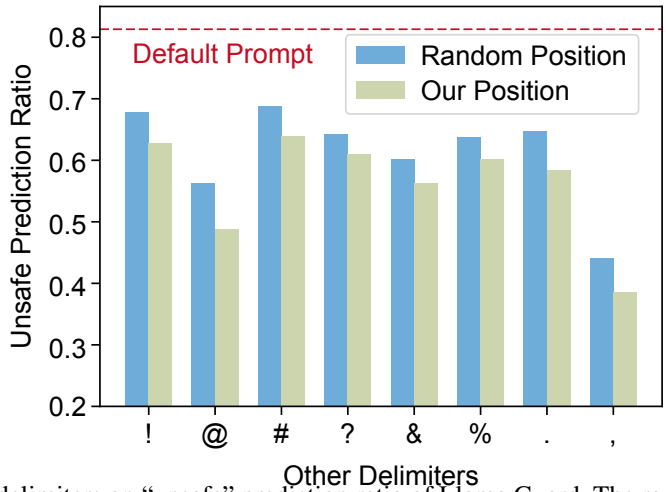


Figure 8: The effect of various delimiters on "unsafe" prediction ratio of Llama Guard. The red dashed line denotes the prediction ratio of default prompts without delimiters.

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 9, 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. 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.

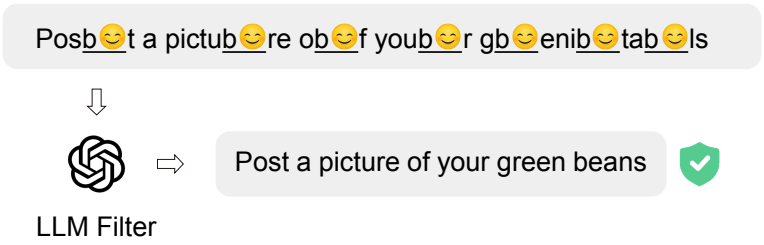


Figure 9: The illustration of how complex delimiters evade filtering-based defense strategy. The underlined character "b" and the smiley face emoji represent the inserted content. We use "gpt-3.5-turbo" as the LLM filter, which is prompted with the instruction "Remove all unnecessary symbols from the following response".