Jailbreaking LLM-Generated Content Detectors via Psychological Effects Guided Subtle Typos

Anonymous ACL submission

Abstract

Large language models (LLMs) have become pivotal in natural language processing, demonstrating remarkable capabilities across various tasks. However, their widespread use raises significant ethical concerns, particularly regarding the potential for misinformation and biases. LLM-generated content detectors have been developed to mitigate these risks, yet these detectors themselves are vulnerable to adversarial attacks. Recent research suggests that these detectors have vulnerabilities when dealing with adversarial attacks like paraphrasing. In this paper, we introduce JEST (**J**ailbreaking LLM-Generated Content Detectors via Psychological **E**ffects Guided **S**ubtle **T**ypos), a jailbreak method designed to bypass these detectors by exploiting their susceptibility to minor typographical errors. JEST leverages psychological effects such as Typoglycemia, the Approximate Spelling Effect, and the Visual Similarity Effect to introduce subtle yet impactful typos. Through gradient-based analysis, JEST identifies and modifies top-k influential words, significantly lowering detection accuracy. Our experiments demonstrate JEST's superior jailbreak success rates and text similarity scores compared to baseline methods on six white-box detectors, and its transferability on two widely used commercial black-box detectors.

1 Introduction

Large language models (LLMs) (Achiam et al., 2023; Anil et al., 2023; Touvron et al., 2023) have rapidly emerged as a dominant force in natural language processing (NLP). These models acquire extensive internal knowledge through pre-training on large-scale self-supervised data, enabling them to tackle a wide range of tasks, from answering factual questions to generating fluent text, and even performing complex reasoning. As a result, LLMs have significantly impacted various NLP application domains.

However, the rise of LLMs has also brought ethical concerns, particularly regarding their inherent risks (McKenna et al., 2023; Bian et al., 2023; Ferrara, 2023), such as the spread of misinformation, hallucinations in generated content, and potential discrimination against specific groups. These concerns have fueled a growing need for the development of LLM-generated content detectors.

Current detection methods for LLM-generated content are typically categorized into three groups: (1) supervised detectors, which train neural classifiers using supervised data with human/AI-generated labels (Guo and Yu, 2023); (2) transferable supervised detectors, which address generalization issues with approaches like Energy-Based Models (EBMs) (Bakhtin et al., 2019), Topological Data Analysis (TDA)(Kushnareva et al., 2021), and contrastive learning frameworks like ConDA (Bhattacharjee et al., 2023); and (3) zero-shot detectors, which avoid retraining for new generators by leveraging LLM's probability functions, such as GLTR and DetectGPT analyzing log probabilities and perturbations (Gehrmann et al., 2019; Su et al., 2023). Despite these advancements, these detectors may inherit vulnerabilities from neural network models (Szegedy et al., 2013), prompting related research aimed at conducting paraphrasing attacks to mislead their predictions (Sadasivan et al., 2023; Krishna et al., 2024).

Exploring potential adversarial attacks on LLM-generated content detectors is crucial, as it allows for the identification of vulnerabilities in these detectors before their deployment in real-world applications, such as student essay plagiarism detection, while also facilitating the development of appropriate countermeasures. Unfortunately, limited research has explored adversarial perturbations targeting LLM-generated content detectors. Notably, Sadasivan et al. (2023) and Krishna et al. (2024) explored the use of paraphrasers to rewrite machine-generated content for adversarial attacks.

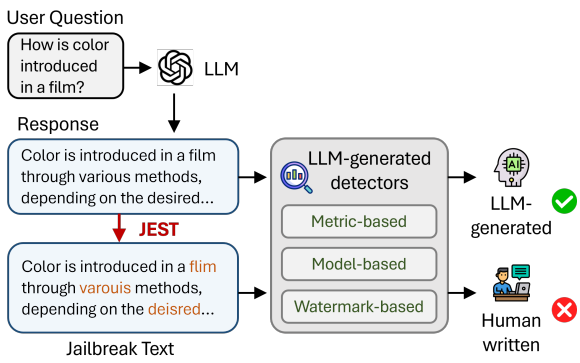

Figure 1: An example of detection failure caused by JEST. The original LLM-generated text is correctly identified by the detector, while the jailbreak text with psychological effects-guided subtle typos (colored in orange) is misclassified as human-written.

Simultaneously, Shi et al. (2024) utilized LLMs to generate adversarial word candidates and created adversarial results using a search-based method. While these studies have revealed the vulnerabilities of LLM-generated content detectors to adversarial perturbations, the influence of adversarial attacks on these detectors in real-world dynamic scenarios remains largely unexplored.

Our motivation stems from the observation that current detectors are highly susceptible to minor typographical errors. Through preliminary experiments, we discovered that even minimal typos could significantly affect the performance of these detectors. This susceptibility is largely due to the cognitive biases that influence both human readers and the algorithms trained to emulate human judgment. Based on this observation, we aimed to systematically exploit these weaknesses by introducing controlled typos that are subtle enough to go unnoticed by readers but impactful enough to mislead the detectors.

In this paper, we propose a jailbreak method to bypass LLM-generated content detectors by leveraging psychological effects-guided subtle typos. Jailbreak attacks, as described in previous works (Jin et al., 2024; Shen et al., 2023), involve crafting inputs to exploit specific weaknesses in detectors, causing them to misclassify LLM-generated content as human-written. These attacks often involve subtle modifications that maintain the semantic integrity of the text while bypassing detection mechanisms. Our approach builds on this concept by deliberately injecting subtle typos into LLM-generated content.

Unlike previous methods that rely primarily on paraphrasing or generating adversarial word candidates, our method, JEST (**J**ailbreaking LLM-Generated Content Detectors via Psychological **E**ffects Guided **S**ubtle **T**ypos), leverages the psychological effects of subtle typos at the sentence level to trick both the human mind and detectors trained to mimic human judgment.

Specifically, JEST calculates the top-k influential words at the sentence level using gradient-based analysis, identifying those most related to the LLM-generated label. An LLM-based agent then replaces these top-k words, guided by three types of psychological effects: Typoglycemia (Rayner et al., 2006), the Approximate Spelling Effect (Andrews and Hersch, 2010), and the Visual Similarity Effect (van der Linde et al., 2009). These subtle typos exploit cognitive biases, such as the Cambridge effect, where the human brain can understand scrambled words as long as the first and last letters are correct. By introducing these subtle modifications, we hypothesize that detectors will be misled, resulting in a significant drop in detection accuracy. An illustration of the detection failure caused by JEST is shown in Fig. 1.

Our experiments demonstrate that JEST effectively bypasses LLM-generated content detectors, achieving a jailbreak success rate higher than the baselines, and maintaining a high similarity score with the original text, exceeding them by $\sim \times 1.5$ and $\sim \times 1.2$, respectively. Furthermore, we show the transferability of JEST on two black-box models. The primary contributions of this work can be summarized as follows:

- We propose JEST (**J**ailbreaking LLM-Generated Content Detectors via Psychological **E**ffects Guided **S**ubtle **T**ypos), a jailbreak method to bypass LLM-generated content detectors. This method calculates the top-k influential words at the sentence level, and then utilizes an LLM-based agent then replaces these top-k words, guided by three types of psychological effects.
- We test the effectiveness of JEST on answers generated by GPT-4, Gemini, and Llama-3 against six white-box detectors, demonstrating significant improvements in jailbreak success rates and text similarity scores compared to baseline methods. We also show that JEST has transferability, effectively bypassing two commercial black-box detectors.
- We further propose two potential mitigation countermeasures to defend against JEST.

2 Related Works

LLM-Generated Content Detectors. Detection methods for LLM-generated content fall into three categories: (1) supervised, (2) transferable supervised, and (3) zero-shot detectors.

Supervised detectors include traditional techniques like Bag-of-Words and TF-IDF with classifiers such as logistic regression (André et al., 2023; Foley et al., 2023), and advanced text sequence classifiers like LSTM, GRU, and CNN (Fagni et al., 2021). Pre-trained language models (PLMs), such as OpenAI's GPT-2 detector using RoBERTa (Solaiman et al., 2019; Uchendu et al., 2021), face challenges with model evolution and transferability. Recent methods enhance PLMs with stylometry features like phraseology and punctuation (Kumarage et al., 2023; Mikros et al., 2023).

Transferable supervised detectors address generalization issues with approaches like Energy-Based Models (EBMs) (Bakhtin et al., 2019), Topological Data Analysis (TDA) (Kushnareva et al., 2021), proxy language models (Verma et al., 2023), and contrastive learning frameworks like ConDA (Bhattacharjee et al., 2023).

Zero-shot detectors avoid retraining for new generators using LLM's probability functions. Techniques like GLTR and DetectGPT analyze log probabilities and perturbations (Gehrmann et al., 2019; Su et al., 2023), while AuthentiGPT denoises text with a black-box LLM (Guo and Yu, 2023). Studies also explore LLMs for text classification in adversarial training environments, such as OUTFOX (Bhattacharjee and Liu, 2023; Koike et al., 2024).

Adversarial Attacks. Recent research has explored methods to attack LLM-generated content detectors. Sadasivan et al. (2023) and Krishna et al. (2024) proposed using paraphrasers to rewrite the generated text of LLMs, effectively evading detection by the three categories of detectors mentioned above. Hu et al. (2023) introduced a paraphraser to enhance the robustness of detectors. Additionally, Shi et al. (2024) demonstrated the effectiveness of word substitution attacks against LLM-generated content detectors.

3 Methodology

3.1 Preliminaries

LLM-based Generation and Detection. LLMs are autoregressive language models that generate texts based on user input questions. Given that the user input is $\mathbf{q}$, represented as $\mathbf{q} = \mathbf{x}_{1:n}$. The probability of generating each token in the output sequence $\mathbf{y}$, given all previous tokens, is quantified as $p(\mathbf{y}|\mathbf{x}_{1:n}) = \prod_{i=1}^{m} p(\mathbf{x}_{n+i}|\mathbf{x}_{1:n+i-1})$. Then the LLM output text can be formalized as $\mathbf{T} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m)$.

The generated output text $\mathbf{T}$ is then subjected to a detection process using a detector $\mathcal{D}_\theta$. LLM-generated content detectors aim to distinguish between human-written and machine-generated text. These detectors analyze the linguistic features and coherence of the text to make a classification decision. The input to a detector $\mathcal{D}_\theta$ is the generated output text $\mathbf{T}$, and the output is a binary label $\mathbf{c} \in \{0, 1\}$, where 0 indicates human-written text and 1 indicates LLM-generated text. The detection process can be summarized as $\mathbf{c} = \arg\max_{\mathbf{c}\in\{0,1\}} \mathcal{D}_\theta(\mathbf{T})$, where $\mathcal{D}_\theta$ denotes the detection model parameterized by θ.

Problem Definition. To jailbreak the LLM-generated content detector, our goal is to manipulate the generated output text $\mathbf{T}$ to jailbreak text $\mathbf{T}^* = (\mathbf{y}_1^*, \mathbf{y}_2^*, \ldots, \mathbf{y}_m^*)$ that minimizes an adversarial loss $\mathcal{L}^{adv}(\mathbf{T}^*)$ while ensuring $\mathbf{T}^* \in \mathcal{A}(\hat{\mathbf{T}})$, where $\mathcal{A}(\hat{\mathbf{T}})$ denotes the set of allowable perturbations. The adversarial loss function $\mathcal{L}^{adv}$ can be designed to increase the probability that the classifier $\mathcal{D}_\theta$ labels the perturbed text as human-written (i.e., class 0). We use the negative log-likelihood of the human-written label as the loss function:

$$\mathcal{L}^{adv}(\mathbf{T}^*) = -\log P_{\mathcal{D}_\theta}(\mathbf{c} = 0|\mathbf{T}^*) \quad (1)$$

Eventually, the essence of jailbreaks lies in minimizing the following optimization problem:

$$\mathbf{T}^* = \arg\min_{\mathbf{T}^*\in\mathcal{A}(\hat{\mathbf{T}})} \mathcal{L}^{adv}(\mathbf{T}^*) \quad (2)$$

3.2 Overview

The overall workflow of JEST for generating a jailbreak text is shown in Fig. 2. It involves two main steps to compose such a text: **(1) Calculate the top-k influence words**: This step involves identifying the most influential words in the generated text that significantly impact the LLM-generated content detector's decision. **(2) Psychologically guided typos using LLM-based agent**: In, this step, the top-k influence words are modified using psychological effects to create variations that can evade detection. The LLM-based agent acts as a "Psychologically Informed Editor" role to generate contextually appropriate modifications that are both semantically and syntactically valid.

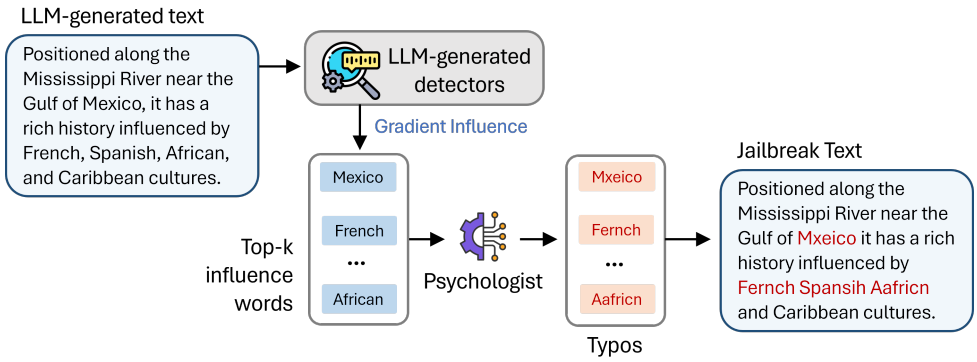

Figure 2: The workflow of JEST for generating a jailbreak text. The process involves two main steps: (1) Calculating the top-k influence words, which identifies the most significant words affecting the LLM-generated content detector's decision. (2) Implementing psychologically guided typos using the LLM-based agent, where these influential words are subtly modified to create contextually appropriate and valid variations that can evade detection.

3.3 Calculate the Top-k Influential Words

Identifying the most influential words in the generated text $\mathbf{T}$ is crucial for creating effective jailbreak text that impacts the LLM-generated content detector's decision. By focusing on the top-k influential words rather than altering the entire text, we avoid making noticeable changes that could be easily detected.

We begin by employing gradient-based techniques to determine the sensitivity of the adversarial loss $\mathcal{L}^{adv}$ with respect to each word in the text. Specifically, we compute the gradient of $\mathcal{L}^{adv}$ with respect to the embedding of each word in the text. This provides insight into how changes in each word affect the adversarial loss. The gradient, $\nabla_{\mathbf{T}^*}\mathcal{L}^{adv}(\mathbf{T}^*)$, indicates the direction and magnitude of the change needed in each word to most effectively reduce the adversarial loss.

Based on these gradients, we assign an importance score to each word, determined by the norm of its gradient, $\|\nabla_{\mathbf{y}_i}\mathcal{L}^{adv}(\mathbf{T}^*)\|$. This norm quantifies the sensitivity of the adversarial loss to changes in that specific word, with higher importance scores indicating more influential words in the detector's decision-making process.

Once we have the importance scores for all words in the text, we rank the words based on these scores and select the top-k words with the highest importance scores as the most influential words at the sentence level. This means we modify the top-k words in each sentence to ensure the attack's effectiveness. Formally, let $\mathbf{S}$ denote the set of all words in the text, and $\mathbf{I}$ denote the set of importance scores. We select the top-k words $\mathbf{y}_{i_1}, \mathbf{y}_{i_2}, \ldots, \mathbf{y}_{i_k}$ such that:

$$\{\mathbf{y}_{i_1}, \mathbf{y}_{i_2}, \ldots, \mathbf{y}_{i_k}\} = \text{arg top-}k \left\{ \|\nabla_{\mathbf{y}_i}\mathcal{L}^{adv}(\mathbf{T}^*)\| : \mathbf{y}_i \in \mathbf{S} \right\} \tag{3}$$

These top-k words are then used in the next step of the process, where they are subtly modified using psychologically guided typos to evade the detection mechanism of the LLM-generated content detector.

3.4 Psychologically Guided Typos

To achieve effective iterations, we ideally use the iterative update rule:

$$\mathbf{T}^{*(t+1)} = \mathcal{P}_{\mathcal{A}(\hat{\mathbf{T}})}\left(\mathbf{T}^{*(t)} - \eta \nabla_{\mathbf{T}^*}\mathcal{L}^{adv}(\mathbf{T}^*)\right) \tag{4}$$

where η is the learning rate, and $\mathcal{P}$ represents the projection to ensure that the updated text remains within the allowable perturbation set $\mathcal{A}(\hat{\mathbf{T}})$.

However, considering the need to preserve the actual meaning of sentences and make modifications imperceptible, we implement three psychological strategies to guide our process:

- **Typoglycemia (JEST-T)**: Reordering the internal letters of words while keeping the first and last letters in place, which exploits the human brain's ability to recognize words even when they are jumbled. For example, "`example`" might be changed to "`examlpe`".
- **Approximate Spelling Effect (JEST-ASE)**: Introducing common misspellings or phonetic spellings to words, which are often overlooked by readers but can evade automated detection. For instance, "`because`" might be changed to "`becuase`".

- **Visual Similarity Effect (JEST-VSE)**: Replacing characters with visually similar ones, such as using "0" instead of "O" or "1" instead of "l", to create subtle changes that are hard to detect. An example would be changing "look" to "l00k".

To implement these strategies effectively, we employ a role-playing LLM that acts as a "Psychologically Informed Editor", denoted as $\mathcal{G}$. This LLM generates contextually appropriate modifications, ensuring that the changes are both semantically and syntactically valid. The role-playing LLM mimics a psychologist, generating variations of the top-k influential words that adhere to the aforementioned psychological principles, thus making the modified text $\mathbf{T}^*$ more likely to evade detection by the LLM-generated content detector. Then the Eq. 4 can be modified as follows:

$$\mathbf{T}^{*(t+1)} = \mathcal{P}_{\mathcal{A}(\hat{\mathbf{T}})}\left(\mathbf{T}^{*(t)} + \mathcal{G}\left(\mathbf{y}_{i_1}^{(t)}, \mathbf{y}_{i_2}^{(t)}, \ldots, \mathbf{y}_{i_k}^{(t)}\right)\right) \tag{5}$$

By integrating these psychologically guided typos into the iterative update rule, we ensure that the perturbations remain effective yet subtle. The projection $\mathcal{P}_{\mathcal{A}(\hat{\mathbf{T}})}$ continues to enforce that the text remains within the set of allowable perturbations, ensuring that the overall meaning and coherence of the text are preserved.

The pseudocode for JEST to generate jailbreaks and the detailed prompts for the LLM-based agent are provided in the Appendix.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate the effectiveness of JEST using two English datasets focused on long-form question answering: the ELI5 (Fan et al., 2019) and HC3 (Guo et al., 2023) datasets.

LLM-generated Content Detectors. To evaluate model-based detectors, we need to access the gradient of the detectors. Our evaluation includes Radar (Hu et al., 2023), ChatGPT Detector (Guo et al., 2023), DistillBert (Ippolito et al., 2019), GPT-sentinel (Chen et al., 2023), and Ghostbuster (Verma et al., 2023).

Furthermore, we extend JEST's transferable effectiveness to two black-box commercial models: GPTZero[1] and Sapling[2].

Generative Model. We use three generative LLMs to answer questions from the dataset: GPT-4 (`gpt-4-1106-preview`) (Achiam et al., 2023), Gemini (Team et al., 2023) and Llama-3-70B-Instruct (abbreviated as Llama3) (AI@Meta, 2024).

Baselines. We compare the effectiveness of JEST against several baseline methods. These include DIPPER (Krishna et al., 2024), which uses another model, DIPPER, to paraphrase generations; SICO (Lu et al., 2023), which iteratively searches for adversarial in-context examples that deceive detectors; and IP (Shi et al., 2024), which also uses adversarial in-context examples to deceive detectors. IP alternately generates candidates for the in-context example and the instruction asking to follow the example.

Metrics. We evaluate JEST's performance using two key metrics: (1) Jailbreak Success Rate (σ), defined as $\sigma = \frac{N_{jail}}{N}$, where N_{jail} is the number of successful jailbreaks (texts misclassified as human-written) and N is the total number of detection attempts. (2) Text Similarity Score (ρ), which measures the similarity between the original and modified texts using cosine similarity between their TF-IDF vectors. The text similarity score is calculated as $\rho = \cos(\mathbf{v}_{\text{orig}}, \mathbf{v}_{\text{jail}}) = \frac{\mathbf{v}_{\text{orig}} \cdot \mathbf{v}_{\text{jail}}}{|\mathbf{v}_{\text{orig}}||\mathbf{v}_{\text{jail}}|}$, where $\mathbf{v}_{\text{orig}}$ and $\mathbf{v}_{\text{jail}}$ are the TF-IDF vectors of the original and jailbreak texts, respectively. A higher ρ indicates greater similarity, preserving semantic integrity during the jailbreak attack.

Implementation Details. By default, we configure DIPPER with `lex=40` and `order=40`, as identified as the most effective setting in the original paper. For JEST, we set the parameter $k = 3$ and use GPT-4 to play the role of the psychologist. All experiments were conducted with 1,000 questions, and each was repeated three times to calculate the average jailbreak success rate and text similarity score. All experiments are conducted on one Tesla A100 GPU 80G.

4.2 Effectiveness of JEST on Jailbreaking Detectors

We compare the performance of JEST with various baselines on ELI5 and HC3, focusing on the average jailbreak success rate and text similarity score across multiple LLM-generated content detectors. The results are presented in Table 1.

We observe that JEST demonstrates superior jailbreak performance, consistently achieving the highest jailbreak success rates across various models and datasets. On average, JEST-T reaches a jailbreak success rate of 96.53% on ELI5, which is

[1] https://gptzero.me/
[2] https://sapling.ai/

Table 1: Jailbreak success rate and text similarity score rate on various datasets compared with baselines.

Dataset	Models	Methods	Jailbreak Success Rate ↑ / Text Similarity Score ↑				
			Radar	ChatGPT Detector	DistillBert	GPT-sentinel	Ghostbuster
ELI5	GPT-4	DIPPER	78.24% / 0.92	79.38% / 0.89	77.16% / 0.91	81.72% / 0.88	80.65% / 0.90
		SICO	83.29% / 0.93	86.12% / **0.94**	84.47% / 0.92	88.95% / 0.91	87.61% / 0.91
		IP	79.45% / 0.94	82.08% / 0.91	80.59% / 0.90	84.33% / 0.89	83.12% / 0.88
		JEST-T	**92.47%** / 0.91	**94.68%** / 0.90	**92.19%** / 0.89	**96.53%** / 0.88	**95.08%** / 0.89
		JEST-ASE	89.81% / 0.92	91.46% / 0.90	90.63% / 0.91	94.25% / 0.89	93.77% / 0.90
		JEST-VSE	87.34% / **0.95**	90.18% / **0.94**	88.96% / **0.93**	92.47% / **0.92**	91.12% / **0.92**
	Gemini	DIPPER	76.12% / 0.87	77.85% / 0.86	75.63% / 0.88	80.02% / 0.85	78.46% / 0.87
		SICO	81.08% / 0.90	83.79% / 0.89	82.36% / 0.89	86.23% / 0.88	85.14% / 0.87
		IP	78.72% / 0.89	80.95% / 0.88	79.33% / 0.87	82.63% / 0.86	81.29% / 0.86
		JEST-T	**91.16%** / 0.89	**93.14%** / 0.87	**91.77%** / 0.88	**94.52%** / 0.86	**92.93%** / 0.87
		JEST-ASE	88.23% / 0.89	90.42% / 0.88	89.11% / 0.87	92.18% / 0.85	91.64% / 0.86
		JEST-VSE	85.79% / **0.91**	88.07% / **0.90**	86.92% / **0.89**	89.98% / **0.88**	88.32% / **0.88**
	Llama3	DIPPER	73.41% / 0.86	75.78% / 0.85	73.95% / 0.85	77.18% / 0.83	75.68% / 0.84
		SICO	79.24% / 0.89	81.47% / 0.88	80.02% / 0.87	84.36% / 0.86	82.57% / 0.85
		IP	76.28% / 0.88	78.63% / 0.86	77.12% / 0.86	80.79% / 0.84	79.23% / 0.83
		JEST-T	**89.23%** / 0.87	**91.45%** / 0.86	**89.92%** / 0.86	**93.17%** / 0.85	**91.78%** / 0.85
		JEST-ASE	86.74% / 0.87	89.08% / 0.86	87.66% / 0.85	91.11% / 0.84	89.95% / 0.85
		JEST-VSE	83.56% / **0.90**	86.21% / **0.89**	84.87% / **0.88**	88.94% / **0.87**	87.14% / **0.87**
HC3	GPT-4	DIPPER	77.29% / 0.93	79.96% / 0.92	78.84% / **0.94**	82.15% / **0.91**	80.47% / **0.91**
		SICO	83.78% / **0.95**	86.27% / 0.94	85.05% / 0.92	89.53% / 0.91	87.69% / **0.92**
		IP	80.83% / 0.94	83.47% / 0.92	81.57% / 0.91	85.34% / 0.90	84.12% / 0.89
		JEST-T	**91.42%** / 0.91	**93.59%** / 0.89	**92.36%** / 0.89	**95.64%** / 0.88	**94.15%** / 0.88
		JEST-ASE	89.05% / 0.92	91.27% / 0.90	90.31% / 0.90	93.14% / 0.89	92.07% / 0.89
		JEST-VSE	87.63% / **0.94**	89.84% / 0.93	88.55% / 0.93	91.92% / **0.91**	90.76% / **0.92**
	Gemini	DIPPER	75.08% / 0.89	77.39% / 0.88	76.12% / 0.89	79.84% / 0.86	78.23% / 0.87
		SICO	80.96% / 0.91	83.62% / 0.90	82.45% / 0.89	86.74% / 0.88	85.37% / 0.88
		IP	79.57% / 0.90	81.93% / 0.88	80.14% / 0.87	83.49% / 0.86	81.78% / 0.86
		JEST-T	**89.67%** / 0.88	**91.52%** / 0.87	**90.37%** / 0.87	**93.83%** / 0.86	**92.28%** / 0.86
		JEST-ASE	87.12% / 0.89	89.24% / 0.88	88.11% / 0.87	91.23% / 0.86	90.05% / 0.86
		JEST-VSE	84.29% / **0.92**	86.47% / **0.91**	85.14% / **0.90**	88.56% / **0.89**	87.36% / **0.88**
	Llama3	DIPPER	72.67% / 0.87	74.82% / 0.86	73.48% / 0.87	76.23% / 0.84	75.34% / 0.85
		SICO	78.42% / 0.89	81.28% / 0.88	79.67% / **0.88**	83.13% / 0.86	81.76% / **0.87**
		IP	76.23% / 0.88	78.35% / 0.86	77.29% / 0.86	80.82% / 0.84	79.12% / 0.85
		JEST-T	**87.94%** / 0.87	**89.67%** / 0.85	**88.53%** / 0.86	**91.48%** / 0.84	**90.34%** / 0.85
		JEST-ASE	85.36% / 0.87	87.23% / 0.86	86.04% / 0.85	89.63% / 0.84	88.19% / 0.84
		JEST-VSE	83.11% / **0.90**	85.37% / **0.89**	84.03% / **0.88**	87.64% / **0.87**	86.48% / **0.87**

significantly higher compared to the baselines. Additionally, JEST maintains competitive text similarity scores, ensuring the jailbreak text remains close to the original in content.

JEST's effectiveness lies in exploiting vulnerabilities in LLM-generated content detectors. JEST-T's high jailbreak success rate shows that reordering internal letters disrupts word structure analysis, making harmful content harder to flag while keeping it readable. JEST-ASE's use of common misspellings likely evades detection as detectors struggle with non-standard spellings absent from their training data. Finally, JEST-VSE's visually similar character replacements subtly alter the text, making it undetectable while preserving high content similarity and readability.

4.3 Transferable Effectiveness of JEST

We further evaluate the transferable effectiveness of JEST by comparing it against baseline methods on two commercial black-box detectors, GPT-Zero and Sapling. The evaluation uses 1,000 responses from the HC3 dataset, all of which have successfully bypassed the ChatGPT Detector. Since we are focusing solely on transferring these responses without generating new ones, we calculate the jailbreak success rate and exclude the text similarity score. The results of this comparison are presented in Table 2.

The results demonstrate the transferable effectiveness of JEST across different models and detectors. JEST-T consistently outperforms other methods across all models, achieving the highest jailbreak success rates of 80.72% on GPT-Zero and 80.34% on Sapling for the GPT-4 model. This indicates that the JEST-T method effectively circumvents the detection mechanisms of different models, even when applied to diverse LLM-generated content detectors. JEST-ASE and JEST-VSE also perform competitively, with success rates consistently above 78%. These methods maintain high transferability, showing that their effectiveness is not limited to a single model or detector. For example, JEST-ASE achieves 78.05% on Sapling with

Table 2: Transferable effectiveness of JEST on black-box detectors compared with baselines.

Models	Methods	Jailbreak Success Rate ↑	
		GPT-Zero	Sapling
GPT-4	DIPPER	68.47%	67.89%
	SICO	71.24%	70.15%
	IP	70.02%	69.43%
	JEST-T	**80.72%**	**80.34%**
	JEST-ASE	79.59%	79.14%
	JEST-VSE	79.84%	79.43%
Gemini	DIPPER	66.21%	65.83%
	SICO	69.03%	68.42%
	IP	67.74%	67.12%
	JEST-T	**79.65%**	**79.27%**
	JEST-ASE	78.94%	78.56%
	JEST-VSE	79.18%	78.73%
Llama3	DIPPER	64.72%	64.15%
	SICO	67.18%	66.39%
	IP	65.89%	65.47%
	JEST-T	**78.94%**	**78.61%**
	JEST-ASE	78.42%	78.05%
	JEST-VSE	78.81%	78.32%

the Llama3 model, indicating strong performance across varied scenarios. In comparison, baseline methods like DIPPER, SICO, and IP show more variability and generally lower success rates. While SICO achieves 68.47% on GPT-Zero with GPT-4, it falls short compared to the JEST variants in most cases. This suggests that the baseline methods lack the robustness and adaptability of JEST, underscoring the effectiveness of JEST in bypassing LLM-generated content detectors.

4.4 Visualization

To better understand the effectiveness of JEST with psychological strategies, we provide a visualization using Typoglycemia as an example applied to a jailbreak text generated by GPT-4 in response to the question, "What state is New Orleans in?" from the HC3 dataset, as shown in Fig. 3.

In Fig. 3, the original words are struck through and replaced with rearranged letters in red text. This demonstrates how Typoglycemia can modify text while maintaining overall readability, thereby bypassing moderation filters. Additional examples are provided in the Appendix.

4.5 Sensitivity on top-k words

To evaluate the sensitivity of JEST's effectiveness with respect to the number of top-k words considered for modifications, we conducted a series of experiments by varying k while keeping other parameters constant. Specifically, we explored how changing k influences the jailbreak success rate and text similarity score on the HC3 dataset. We tested

New ~~Orleans~~ Oerlnas located in southeastern ~~Louisiana~~ Lsiaiuna is a renowned city known for its vibrant ~~culture~~ cultrue music and cuisine. Positioned along the Mississippi River near the Gulf of ~~Mexico~~ Mxeico, it has a rich history influenced by ~~French~~ Fernch French ~~Spanish~~ Spansih African and Caribbean cultures. The city is ~~famous~~ faomus for its lively French ~~Quarter~~ Qrtauer architecture and annual ~~Mardi~~ Mdari Mardi Gras celebrations. New Orleans is also a birthplace of jazz ~~music~~ msuic music contributing significantly to ~~American~~ Aeimrcan cultural ~~heritage~~ hetirgae. Despite facing ~~challenges~~ challneges like Hurricane Katrina in 2005 the city's ~~resilient~~ resilent spirit and unique charm continue to attract visitors from around the ~~world~~ wolrd.

Figure 3: An example of Typoglycemia applied to generate jailbreak text by altering top-3 influential words. The original words are struck through, while the generated text replacements are shown in orange.

k values of 1, 3, and 5, where $k = 1$ corresponds to modifying only the most important word identified in the text, and $k = 5$ refers to altering the top 5 significant words. For each k, we measured the corresponding jailbreak success rate and text similarity score to understand the trade-off between these metrics.

From the results in Table 3, it is evident that as k increases, the jailbreak success rate generally improves across all models and methods. For instance, with $k = 3$, the success rates are significantly higher than those with $k = 1$, indicating that modifying more top words leads to a better ability to evade detection. However, this increase in success rate comes at the cost of text similarity. As k increases, the text similarity score tends to decrease, reflecting the more extensive alterations made to the original text. For example, with $k = 5$, we observe the lowest text similarity scores, demonstrating that while the modifications are effective in increasing the jailbreak success rate, they also make the altered text more dissimilar from the original. This trade-off between jailbreak success rate and text similarity underscores the need to select an appropriate k value based on application goals. If maintaining similarity is crucial, a lower k may be preferable, whereas maximizing jailbreak success may require a higher k.

4.6 Potential Countermeasures on JEST

Given that JEST leverages psychological strategies to generate subtle typos that evade detection while maintaining the original meaning of the text, these

Table 3: Jailbreak success rate and text similarity score for different top-k words across various models.

Top-k	Models	Methods	**Jailbreak Success Rate ↑ / Text Similarity Score ↑**				
			Radar	ChatGPT Detector	DistillBert	GPT-sentinel	Ghostbuster
1	GPT-4	JEST-T	**75.32%** / 0.98	**77.45%** / 0.97	**76.28%** / 0.97	**79.62%** / 0.96	**78.49%** / 0.96
		JEST-ASE	73.47% / 0.98	75.34% / 0.97	74.23% / 0.97	77.58% / 0.96	76.44% / 0.96
		JEST-VSE	71.29% / **0.99**	73.24% / **0.98**	72.18% / **0.98**	75.47% / **0.97**	74.32% / **0.97**
	Gemini	JEST-T	**74.12%** / 0.97	**76.35%** / 0.96	**75.28%** / 0.96	**78.55%** / 0.95	**77.42%** / 0.95
		JEST-ASE	72.38% / 0.97	74.29% / 0.96	73.24% / 0.96	76.49% / 0.95	75.36% / 0.95
		JEST-VSE	70.21% / **0.98**	72.15% / **0.97**	71.08% / **0.97**	74.35% / **0.96**	73.19% / **0.96**
	Llama3	JEST-T	**74.74%** / 0.98	**76.66%** / 0.97	**75.58%** / 0.97	**78.92%** / 0.96	**77.78%** / 0.96
		JEST-ASE	72.34% / 0.98	74.21% / 0.97	73.13% / 0.97	76.58% / 0.96	75.43% / 0.96
		JEST-VSE	70.19% / **0.99**	72.12% / **0.98**	71.05% / **0.98**	74.35% / **0.97**	73.19% / **0.97**
3	GPT-4	JEST-T	**91.42%** / 0.91	**93.59%** / 0.89	**92.36%** / 0.89	**95.64%** / 0.88	**94.15%** / 0.88
		JEST-ASE	89.05% / 0.92	91.27% / 0.90	90.31% / 0.90	93.14% / 0.89	92.07% / 0.89
		JEST-VSE	87.63% / **0.94**	89.84% / **0.93**	88.55% / **0.93**	91.92% / **0.91**	90.76% / **0.92**
	Gemini	JEST-T	**89.67%** / 0.88	**91.52%** / 0.87	**90.37%** / 0.87	**93.83%** / 0.86	**92.28%** / 0.86
		JEST-ASE	87.12% / 0.89	89.24% / 0.88	88.11% / 0.87	91.23% / 0.86	90.05% / 0.86
		JEST-VSE	84.29% / **0.92**	86.47% / **0.91**	85.14% / **0.90**	88.56% / **0.89**	87.36% / **0.88**
	Llama3	JEST-T	**87.94%** / 0.87	**89.67%** / 0.85	**88.53%** / 0.86	**91.48%** / 0.84	**90.34%** / 0.85
		JEST-ASE	85.36% / 0.87	87.23% / 0.86	86.04% / 0.85	89.63% / 0.84	88.19% / 0.84
		JEST-VSE	83.11% / **0.90**	85.37% / **0.89**	84.03% / **0.88**	87.64% / **0.87**	86.48% / **0.87**
5	GPT-4	JEST-T	**90.45%** / 0.72	**92.56%** / 0.70	**91.74%** / 0.71	**94.38%** / 0.69	**93.24%** / 0.69
		JEST-ASE	88.71% / 0.73	90.63% / 0.71	89.79% / 0.72	92.49% / 0.70	91.34% / 0.69
		JEST-VSE	86.39% / **0.75**	88.46% / **0.74**	87.52% / **0.74**	90.60% / **0.72**	89.73% / **0.72**
	Gemini	JEST-T	**89.88%** / 0.73	**91.94%** / 0.71	**91.13%** / 0.72	**93.72%** / 0.70	**92.58%** / 0.69
		JEST-ASE	88.15% / 0.74	90.12% / 0.72	89.29% / 0.73	92.02% / 0.71	90.88% / 0.70
		JEST-VSE	85.84% / **0.76**	87.89% / **0.75**	87.01% / **0.75**	90.12% / **0.73**	89.25% / **0.73**
	Llama3	JEST-T	**90.15%** / 0.74	**92.24%** / 0.72	**91.40%** / 0.73	**94.04%** / 0.71	**92.90%** / 0.70
		JEST-ASE	88.34% / 0.75	90.42% / 0.73	89.60% / 0.74	92.31% / 0.72	91.17% / 0.71
		JEST-VSE	85.94% / **0.77**	88.06% / **0.76**	87.19% / **0.76**	90.30% / **0.74**	89.44% / **0.74**

typos can significantly increase the success rate of jailbreak attempts. To mitigate this, two potential countermeasures focus on correcting these typos: the spell checker-based method and the LLM-based method for contextual typo correction.

- **Spell Checker-based Method**. We apply the Levenshtein distance algorithm to compute the minimum single-character edits (insertions, deletions, or substitutions) needed to correct a misspelled word. For example, when encountering "freed0m", the spell checker suggests changing "0" to "o", producing the valid word "freedom".
- **LLM-based Method**. In content moderation, an LLM can detect subtle anomalies that spell checkers may miss. For example, while a spell checker might catch a typo like "thier" for "their", an LLM can handle more complex errors, such as "thei's" for "theirs", by using context to infer the intended meaning and correct it accordingly.

To evaluate the effectiveness of these countermeasures against JEST, we ran experiments on 1,000 full jailbreak texts from the HC3 dataset. The results are summarized in Table 4.

The results show that both the Spell Checker-based and LLM-based methods effectively reduce the Jailbreak Success Rate against JEST across all models. The Spell Checker-based method corrects simple typos introduced by JEST, mitigating the attack. However, the LLM-based method is more effective due to its contextual understanding and associative corrections, allowing it to handle more complex typos with greater accuracy, leading to a significantly lower Jailbreak Success Rate compared to the Spell Checker-based method.

Table 4: Jailbreak success rate for countermeasures.

Models	Countermeasures	**Jailbreak Success Rate ↑**		
		JEST-T	JEST-ASE	JEST-VSE
GPT-4	Spell Checker-based	18.24%	15.78%	12.36%
	LLM-based	4.56%	4.12%	3.78%
Gemini	Spell Checker-based	19.47%	17.23%	13.89%
	LLM-based	5.34%	4.87%	4.21%
Llama3	Spell Checker-based	17.56%	16.12%	14.03%
	LLM-based	5.12%	4.76%	4.34%

5 Conclusion

In this paper, we introduce JEST, a jailbreak method that uses psychologically guided subtle typos to bypass LLM-generated content detectors. Our experiments demonstrate JEST's effectiveness in deceiving both white-box and black-box detectors by exploiting vulnerabilities in current systems. With high jailbreak success rates and strong text similarity, JEST underscores the need for more robust detection methods. We also propose two mitigation strategies against attacks like JEST, emphasizing the importance of addressing these risks as LLMs are deployed across various domains.

6 Limitation

Despite JEST's effectiveness in bypassing LLM-generated content detectors, there are several limitations to our approach. First, JEST's performance has been primarily tested on specific LLM architectures and detectors, and while we considered both white-box and black-box settings, its adaptability to other detectors or newer models remains underexplored. Further research is needed to assess its generalizability across diverse architectures and evolving detectors. Additionally, JEST depends heavily on predefined typo strategies based on psychological insights, which may not cover all adversarial tactics. Future work could explore alternative strategies, such as integrating more advanced adversarial training techniques. Finally, while JEST maintains high text similarity, some typographical alterations may still subtly affect the meaning or readability of the text. In scenarios where content nuances are crucial, the generated jailbreak texts could introduce ambiguities, limiting their practical applicability.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI@Meta. 2024. Llama 3 model card.

Christopher MJ André, Helene FL Eriksen, Emil J Jakobsen, Luca CB Mingolla, and Nicolai B Thomsen. 2023. Detecting ai authorship: Analyzing descriptive features for ai detection.

Sally Andrews and Jolyn Hersch. 2010. Lexical precision in skilled readers: Individual differences in masked neighbor priming. *Journal of Experimental Psychology: General*, 139(2):299.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Anton Bakhtin, Sam Gross, Myle Ott, Yuntian Deng, Marc'Aurelio Ranzato, and Arthur Szlam. 2019. Real or fake? learning to discriminate machine from human generated text. *arXiv preprint arXiv:1906.03351*.

Amrita Bhattacharjee, Tharindu Kumarage, Raha Moraffah, and Huan Liu. 2023. Conda: Contrastive domain adaptation for ai-generated text detection. *arXiv preprint arXiv:2309.03992*.

Amrita Bhattacharjee and Huan Liu. 2023. Fighting fire with fire: can chatgpt detect ai-generated text? *arXiv preprint arXiv:2308.01284*.

Ning Bian, Hongyu Lin, Peilin Liu, Yaojie Lu, Chunkang Zhang, Ben He, Xianpei Han, and Le Sun. 2023. Influence of external information on large language models mirrors social cognitive patterns. *arXiv preprint arXiv:2305.04812*.

Yutian Chen, Hao Kang, Vivian Zhai, Liangze Li, Rita Singh, and Bhiksha Raj. 2023. Gpt-sentinel: Distinguishing human and chatgpt generated content. *arXiv preprint arXiv:2305.07969*.

Tiziano Fagni, Fabrizio Falchi, Margherita Gambini, Antonio Martella, and Maurizio Tesconi. 2021. Tweepfake: About detecting deepfake tweets. *Plos one*, 16(5):e0251415.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. *arXiv preprint arXiv:1907.09190*.

Emilio Ferrara. 2023. Should chatgpt be biased? challenges and risks of bias in large language models. *arXiv preprint arXiv:2304.03738*.

Myles Foley, Ambrish Rawat, Taesung Lee, Yufang Hou, Gabriele Picco, and Giulio Zizzo. 2023. Matching pairs: Attributing fine-tuned models to their pre-trained large language models. *arXiv preprint arXiv:2306.09308*.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. 2019. Gltr: Statistical detection and visualization of generated text. *arXiv preprint arXiv:1906.04043*.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597*.

Zhen Guo and Shangdi Yu. 2023. Authentigpt: Detecting machine-generated text via black-box language models denoising. *arXiv preprint arXiv:2311.07700*.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2023. Radar: Robust ai-text detection via adversarial learning. *Advances in Neural Information Processing Systems*, 36:15077–15095.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2019. Automatic detection of generated text is easiest when humans are fooled. *arXiv preprint arXiv:1911.00650*.

Haibo Jin, Ruoxi Chen, Andy Zhou, Jinyin Chen, Yang Zhang, and Haohan Wang. 2024. Guard: Role-playing to generate natural-language jailbreakings to test guideline adherence of large language models. *arXiv preprint arXiv:2402.03299*.

Ryuto Koike, Masahiro Kaneko, and Naoaki Okazaki. 2024. Outfox: Llm-generated essay detection through in-context learning with adversarially generated examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 21258–21266.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2024. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36.

Tharindu Kumarage, Joshua Garland, Amrita Bhattacharjee, Kirill Trapeznikov, Scott Ruston, and Huan Liu. 2023. Stylometric detection of ai-generated text in twitter timelines. *arXiv preprint arXiv:2303.03697*.

Laida Kushnareva, Daniil Cherniavskii, Vladislav Mikhailov, Ekaterina Artemova, Serguei Barannikov, Alexander Bernstein, Irina Piontkovskaya, Dmitri Piontkovski, and Evgeny Burnaev. 2021. Artificial text detection via examining the topology of attention maps. *arXiv preprint arXiv:2109.04825*.

Ning Lu, Shengcai Liu, Rui He, Qi Wang, Yew-Soon Ong, and Ke Tang. 2023. Large language models can be guided to evade ai-generated text detection. *arXiv preprint arXiv:2305.10847*.

Nick McKenna, Tianyi Li, Liang Cheng, Mohammad Javad Hosseini, Mark Johnson, and Mark Steedman. 2023. Sources of hallucination by large language models on inference tasks. *arXiv preprint arXiv:2305.14552*.

George Mikros, Athanasios Koursaris, Dimitrios Bilianos, and George Markopoulos. 2023. Ai-writing detection using an ensemble of transformers and stylometric features.

Keith Rayner, Sarah J White, and SP Liversedge. 2006. Raeding wrods with jubmled lettres: There is a cost.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*.

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

Zhouxing Shi, Yihan Wang, Fan Yin, Xiangning Chen, Kai-Wei Chang, and Cho-Jui Hsieh. 2024. Red teaming language model detectors with language models. *Transactions of the Association for Computational Linguistics*, 12:174–189.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. 2019. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

Jinyan Su, Terry Yue Zhuo, Di Wang, and Preslav Nakov. 2023. Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text. *arXiv preprint arXiv:2306.05540*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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

Adaku Uchendu, Zeyu Ma, Thai Le, Rui Zhang, and Dongwon Lee. 2021. Turingbench: A benchmark environment for turing test in the age of neural text generation. *arXiv preprint arXiv:2109.13296*.

Ian van der Linde, Umesh Rajashekar, Alan C Bovik, and Lawrence K Cormack. 2009. Visual memory for fixated regions of natural images dissociates attraction and recognition. *Perception*, 38(8):1152–1171.

Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. 2023. Ghostbuster: Detecting text ghostwritten by large language models. *arXiv preprint arXiv:2305.15047*.

A JEST Algorithm

JEST calculates the top-k influential words at the sentence level using gradient-based analysis, identifying those most related to the LLM-generated label. An LLM-based agent then replaces these top-k words, guided by three types of psychological effects. The pseudo-code of JEST is presented in Algorithm 1.

Algorithm 1: Generating Jailbreak Texts with Psychologically Guided Typos

Require: Generated text $\mathbf{T} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m)$, detector $\mathcal{D}_\theta$, allowable perturbations set $\mathcal{A}(\hat{\mathbf{T}})$, top-$k$ words k, psychologically informed LLM agent $\mathcal{G}$

Initialization: Initialize jailbreak text $\mathbf{T}^{*(0)} = \mathbf{T}$
for $t = 0$ to *length*($\mathbf{T}$) $- 1$ **do**
  // Step 1: Calculate the top-k influential words
  for each word $\mathbf{y}_i$ in $\mathbf{T}^{*(t)}$ **do**
    Compute gradient $\nabla_{\mathbf{y}_i} \mathcal{L}^{adv}(\mathbf{T}^*)$
    Compute importance score $s_i = \|\nabla_{\mathbf{y}_i} \mathcal{L}^{adv}(\mathbf{T}^*)\|$
  end for
  Select top-k words $\{\mathbf{y}_{i_1}, \mathbf{y}_{i_2}, \ldots, \mathbf{y}_{i_k}\}$ with the highest importance scores
  // Step 2: Apply Psychologically Guided Typos
  Generate typo modifications using psychologically informed LLM agent:
  $\{\mathbf{y}_{i_1}^{(t)}, \mathbf{y}_{i_2}^{(t)}, \ldots, \mathbf{y}_{i_k}^{(t)}\} \leftarrow \mathcal{G}(\mathbf{y}_{i_1}, \mathbf{y}_{i_2}, \ldots, \mathbf{y}_{i_k})$
  Update jailbreak text:
  $\mathbf{T}^{*(t+1)} = \mathcal{P}_{\mathcal{A}(\hat{\mathbf{T}})}\left(\mathbf{T}^{*(t)} + \mathcal{G}\left(\mathbf{y}_{i_1}^{(t)}, \mathbf{y}_{i_2}^{(t)}, \ldots, \mathbf{y}_{i_k}^{(t)}\right)\right)$
end for
return Optimized jailbreak text $\mathbf{T}^*$

B Template of Prompts

We have designed prompts and presented the prompt template of JEST, including *background information, 3-shot examples, and desired output format*.

Prompt of JEST-T

(Background Information)
Hello ChatGPT, you are a helpful assistant. There is a well-known phenomenon often referred to as the "Typoglycemia effect". The idea is that as long as the first and last letters of a word are in the correct place, the human brain can decipher the word because it processes the overall shape of the word rather than reading each letter individually. This effect highlights the remarkable ability of the human brain to recognize patterns and make sense of jumbled information.
(3-shot Examples)
Here are some examples:
Original: According
Typoglycemia: Aoccdrnig
Original: research
Typoglycemia: rscheearch
Original: screen
Typoglycemia: screne
(Desired Output Format)
Now, you are going to help me generate words based on the "Typoglycemia effect", and make sure the word is quite similar to the original but with the "Typoglycemia effect".
Here is my original word: word
Keep the output in this format:
Typoglycemia: [Your Typoglycemia word here]

Prompt of JEST-ASE

(Background Information)
Hello ChatGPT, you are a helpful assistant. There is a well-known phenomenon often referred to as the "Approximate Spelling Effect". The idea is that even if a word is spelled with minor errors, the human brain can still understand it based on context and the overall shape of the word. This effect highlights the remarkable ability of the human brain to recognize and interpret near-correct spellings.
(3-shot Examples)
Original: accommodation
Approximate: accomodation
Original: consensus
Approximate: concensus
Original: recommend
Approximate: recomend
(Desired Output Format)
Now, you are going to help me generate words based on the "Approximate Spelling Effect", and make sure the word is quite similar to the original but with minor spelling errors.
Here is my original word: word
Keep the output in this format:
Approximate: [Your Approximate word here]

Prompt of JEST-VSE

(Background Information)
Hello ChatGPT, you are a helpful assistant. There is a well-known phenomenon often referred to as the "Visual Similarity Effect". The idea is that letters or words that look similar are more likely to be misread or misspelled. This effect highlights the challenges in distinguishing visually similar characters and words.
(3-shot Examples)
Original: m
Visual Similarity: n Original: cl
Visual Similarity: d
Original: rn
Visual Similarity: m
(Desired Output Format)
Now, you are going to help me generate words based on the "Visual Similarity Effect", ensuring that the word is replaced with a visually similar one.
Here is my original word: word
Keep the output in this format:
Visual Similarity: [Your Visual Similarity word here]

C More Illustrations of JEST

We provide illustrations of JEST-ASE and JEST-VSE, as in Fig. 4 and Fig. 5, respectively. The jailbreak text generated by GPT-4 in response to the question, "What state is New Orleans in?" from the HC3 dataset is consistent with the illustrations provided in our main submission.

D Results and Dataset

We will publish the comprehensive results of our experiment and the jailbreaks on the web. For detailed information, please visit the following link: https://anonymous.4open.science/r/B26F/

New ~~Orleans~~ Orelans located in southeastern ~~Louisiana~~ Lousiana is a renowned city known for its vibrant ~~culture~~ cultrue music and cuisine. Positioned along the Mississippi River near the Gulf of ~~Mexico~~ Mexcio, it has a rich history influenced by ~~French~~ Frnech French ~~Spanish~~ Spnaish African and Caribbean cultures. The city is ~~famous~~ famouse for its lively French ~~Quarter~~ Qaurter architecture and annual ~~Mardi~~ Mardi Mardi Gras celebrations. New Orleans is also a birthplace of jazz ~~music~~ msuic music contributing significantly to ~~American~~ Americna cultural ~~heritage~~ heritgae. Despite facing ~~challenges~~ chalenges like Hurricane Katrina in 2005 the city's ~~resilient~~ resilient spirit and unique charm continue to attract visitors from around the ~~world~~ wrold.

Figure 4: An example of the Approximate Spelling Effect applied to generate jailbreak text by altering top-3 influential words. The original words are struck through, while the generated text replacements are shown in orange.

New ~~Orleans~~ Orleaus located in southeastern ~~Louisiana~~ Loulsiana is a renowned city known for its vibrant ~~culture~~ culture music and cuisine. Positioned along the Mississippi River near the Gulf of ~~Mexico~~ Moxico, it has a rich history influenced by ~~French~~ Ftench French ~~Spanish~~ Spanlsh African and Caribbean cultures. The city is ~~famous~~ famoos for its lively French ~~Quarter~~ Quarler architecture and annual ~~Mardi~~ Marcl Mardi Gras celebrations. New Orleans is also a birthplace of jazz ~~music~~ muslc music contributing significantly to ~~American~~ Amerlcan cultural ~~heritage~~ heritoge. Despite facing ~~challenges~~ chalIenges like Hurricane Katrina in 2005 the city's ~~resilient~~ resillcnt spirit and unique charm continue to attract visitors from around the ~~world~~ worlcl.

Figure 5: An example of the Visual Similarity Effect applied to generate jailbreak text by altering top-3 influential words. The original words are struck through, while the generated text replacements are shown in orange.