Language Models Can't See GHOST 👻: Defending Digital Text Data from Exploitation with Tokenizer Overloading

Anonymous ACL submission

Abstract

As large language models (LLMs) increasingly extract text data without consent, content creators face growing risks to intellectual property and privacy. This paper outlines the emerging threats of unauthorized content repurposing and sensitive data exposure by AI systems and the need for robust defensive strategies. We introduce **SHROUD** (Structured Hierarchy for Rewriting and Obfuscation in Unstructured Data), the first unified guide to text perturbation and obfuscation techniques to protect text content from misuse. Additionally, we present **GHOST** (Generated Hidden Obfuscation STrategy), a novel, lightweight defensive method that prevents model training and inference on text by injecting invisible tokens that overload a model's tokenizer context window. GHOST allows for imperceptible yet effective obfuscation, preserving human readability while shielding content from unauthorized machine learning use. GHOST serves as a privacy mechanism for evading automated moderation and protecting intellectual property, enabling social media users to share content and online authors to prevent unauthorized AI training using their work.

1 Introduction

The advancement of Large Language Models (LLMs) poses significant intellectual property challenges as these systems appropriate written content without authorization or commercial compensation. Copyright frameworks have been systematically circumvented by AI developers harvesting creators' intellectual output for commercial training data (Kouloumpis et al., 2011; Severyn and Moschitti, 2015; Conover et al., 2011; Li et al., 2023). The resulting derivative works mimic original content while providing no compensation, potentially diminishing authors' creative control in an ecosystem that increasingly commodifies their intellectual labor. Furthermore, LLMs present substantial privacy vulnerabilities, whereby personal communications and confidential materials incorporated into training datasets may subsequently emerge in model outputs. Content creators face the prospect of sensitive information being encoded within model parameters and reproduced in response to user queries.

These challenges call for defensive obfuscation approaches that introduce textual perturbations to shield semantic content from computational interpretation while preserving human readability. Text transformation techniques, implemented at various linguistic levels, have diverse applications across NLP domains: adversarial methods reveal model weaknesses (Hosseini et al., 2017; Gao et al., 2018; Ebrahimi et al., 2017; Jin et al., 2020), robustness assessments evaluate performance under linguistic variance (Belinkov and Bisk, 2017; Michel et al., 2019; Ribeiro et al., 2020; Hendrycks et al., 2020), red-teaming bypasses safety protocols (Wei et al., 2023b; Perez et al., 2022; Zou et al., 2023; Bai et al., 2022), data augmentation enhances generalization (Wei and Zou, 2019; Kobayashi, 2018) and privacy mechanisms safeguard semantic integrity from machine processing.

Evaluation criteria for defensive obfuscation requires a balance between machine imperceptibility and human comprehension. Machine imperceptibility metrics assess models' capacity to reconstruct original meaning or perform downstream tasks such as sentiment analysis and QA. Conversely, utility metrics evaluate preservation of human interpretability, ensuring that a human reader can read and understand the perturbed text with minimal difficulty. Text obfuscation can serve as a privacy control mechanism. Social media users can employ it to share potentially controversial content while evading automated moderation. Authors and creators can apply obfuscation to prevent unauthorized AI training using their work, preserving intellectual property and usage rights.

To address this challenge, this paper presents two key contributions:

- **SHROUD** (Structured Hierarchy for Rewriting and Obfuscation in Unstructured Data): To our knowledge, this is the first conceptual framework to define and organize the transformation search space for generating language model-resistant text. It categorizes transformation strategies that hinder model comprehension, classification, or generation and helps researchers identify opportunities for more novel techniques.

- **GHOST** (Generated Hidden Obfuscation STrategy): a novel, lightweight method for producing obfuscated text that resists LLM training and inference without compromising utility. **GHOST** outperforms prior strategies in robustness against in-context defenses, as demonstrated in Section 5.

2 Related Work

This section reviews prior research on text perturbation, specifically in adversarial attacks and privacy protection. Both approaches employ text obfuscation while preserving utility, differing primarily in their objectives. Adversarial attacks introduce subtle modifications to induce model errors, whereas privacy protection alters text to prevent sensitive information leakage.[1]

2.1 Adversarial Attack

Early adversarial attacks on text classifiers focused on character-level perturbations that preserved meaning but induced misclassification. Hosseini et al. (2017) showed that minor alterations, such as spacing and typos, could bypass toxicity filters. Gao et al. (2018) introduced DeepWordBug, which identified key tokens and applied targeted character edits to exploit model weaknesses. Ebrahimi et al. (2017) further refined this with HotFlip, framing attacks as a discrete optimization problem and using gradient-based methods to identify impactful character-level changes. These studies revealed the vulnerability of neural models to subtle orthographic variations. As character-level defenses improved, research progressed to word-level substitution attacks. Papernot et al. (2016) used gradient-based methods to identify key words for replacement. Alzantot et al. (2018) employed a genetic algorithm to generate fluent adversarial examples using synonym substitutions. Jin et al. (2020) introduced TextFooler, which used context-aware embeddings to ensure semantic similarity and high attack success against BERT. Similarly, BERT-Attack (Li et al., 2020) leveraged BERT masked language modeling for effective word replacement. Bajaj and Vishwakarma (2023) proposed HOMOCHAR, a homoglyph-based character attack that maintained readability while achieving over 90% success on models like BERT and RoBERTa (Liu et al., 2019).

Recent adversarial approaches increasingly target multimodal and instruction-tuned models. Wei et al. (2023a) introduced "jailbreaking" prompts to bypass content filters, while Perez and Ribeiro (2022) detailed prompt injection attacks that override instructions or extract sensitive data. Zou et al. (2023) and Wallace et al. (2019) developed universal prompts and triggers that reliably manipulate model behavior.

2.2 Privacy Protection

Early privacy-preserving NLP drew on structured data methods like k-anonymity (Sweeney, 2002) and differential privacy (Dwork, 2006), later adapted to text by Cumby and Ghani (2011) through redaction techniques. The emergence of large language models prompted concerns over data leakage, with Carlini et al. (2022) revealing their memorization tendencies. This led to formal privacy frameworks: Fernandes (2021) proposed leakage metrics, and Weggenmann and Kerschbaum (2018) introduced SynTF for private text classification. McMahan et al. (2017) applied federated learning with differential privacy to language models, while Yun et al. (2021) mitigated memorization via calibrated noise during pre-training.

Research on text obfuscation for privacy protection has not been systematically organized, as many individual strategies search on only one level. This paper unifies text obfuscation by defining the **SHROUD** Framework, a text transformation

[1] Unfortunately, the research communities discussed above use conflicting terminology to describe the same fundamental goal: preventing an AI system from accessing the text or data generated by another entity. In one community (Section 2.1), the entity blocking access is referred to as the attacker, while the AI model(s) attempting to read the data is the victim. In contrast, the other community (Section 2.2) labels the blocking entity as the defender and the AI model(s) as the attacker. Given that the creator of the text or data is its rightful owner, we expect the latter framing to gain wider acceptance over time.

search space that acts as a starting point for developing novel obfuscation strategies by combining one or more perturbation levels with different transformation strategies.

3 SHROUD Framework

Shown in Table 1, **SHROUD** is a unified framework that provides a systematic overview of digital text obfuscation methods, enabling privacy researchers to apply one or a combination of strategies that conceal semantic meaning from AI language models. This reframes text perturbation as a combinatorial search over obfuscation techniques. The framework can be generalized to tasks like adversarial text, robustness testing, jailbreak-prompt injection, and unlearnable text. It also helps NLP researchers identify unexplored gaps and opportunities in obfuscation and perturbation methods.

Given a document D, the text within D can be characterized by:

- The **level of obfuscation** that determines at what granularity the document is perturbed. This level can be divided into three categories: Character-level, Word/Token-Level and Sentence-Level. Applying obfuscation at levels higher than this (e.g., Paragraph or Document) requires more extensive modifications that could significantly degrade utility.

- The **obfuscation strategy** that, given the level of obfuscation, determines how the object is obfuscated. The strategies can be grouped into 4 main categories: **E**xtra, **D**eletion, **I**njection, **T**ransformation (**EDIT**).

3.1 Level of Obfuscation

Character-level obfuscation alters individual characters through misspellings, typos, or lookalike substitutions (e.g., "o" and "0") to mimic noise from human error. These subtle changes can disrupt word recognition and model comprehension. Word-token level obfuscation modifies entire words or tokens via synonym replacement, deletion, or insertion of unrelated terms. These changes introduce stronger semantic shifts while preserving grammar, testing a model's ability to handle vocabulary variation and ambiguity. Sentence-level obfuscation disrupts structure by rearranging words, adding grammatical errors, or inserting distracting sentences. This challenges the models' ability to parse syntax and retain focus on core meaning.

3.2 Obfuscation Strategies

The **EDIT** framework summarizes all text modification methods based on the obfuscation level and strategy, each targeting different behaviors in the interpreting model. For example, at the word-token level with token $t = (hello)$, the **E**xtra method duplicates $t \rightarrow t' = hellohello$ to draw model focus toward t, potentially overshadowing other informative tokens. In contrast, the **I**njection method appends a different token $t \rightarrow t' = HelloFresh$, altering the semantic meaning and obfuscating the original information (Hello = greeting) vs (HelloFresh = food delivery company). The choice of what and where to perturb is governed by a search function and obfuscation budget.

Simple strategies like repeating (Extra) or omitting (Deletion) characters or tokens are easy to implement and were effective against early language models through adversarial misspelling, though these have since been mitigated (Pruthi et al., 2019). More advanced techniques adopted Injection and Transformation strategies, including word swaps (Ebrahimi et al., 2017; Alzantot et al., 2018), symbol injection (Bajaj and Vishwakarma, 2023) and synonym token substitution (Jin et al., 2020; Li et al., 2020; Liang et al., 2017). Within the realm of Injection and Transformation, text object can be replaced with various substitution options as shown in Table 2. Injection strategies might induce interpreting model(s) to mis-tokenize (e.g., 'Wal 文 king' → 'wal' + ' 文' + 'king' instead of 'walking'), alter the semantics (e.g., Ore🅾️ could be misread as Oreo), or misinterpret the token (e.g., inject(泉,Spring) → Spring 泉 could cause state-of-the-art (SoTA) multilingual models to interpret the combined token as 'fountain'). Likewise, Transformation strategies can also cause mistokenization and misinterpretation, while further obscuring content by converting characters into emojis, symbols, images, or rare synonyms.

4 Problem Formulation

This section formalizes the problem. Given an input text string x, the semantic understanding of a function f (e.g., an LLM) on x is measured by its ability to produce a reconstruction y that preserves the essential information in x. Formally, for an input $x = (w_1, w_2, \ldots, w_n) \in X$, the model maps x to an output $y = (s_1, s_2, \ldots, s_m) \in Y$, where w_i and s_i denote tokens at position i in x and y,

Lvl/Strategies	Extra	Deletion	Injection	Transformation
Character	Insert existing character(s)	Remove character(s)	Insert a foreign character	Change character(s) to a different character
Word Token	Repeat the existing token	Drop the whole token	Insert new token(s)	Replace token(s) with a different token
Sentence	Append the same sentence at the start or end	Drop the entire sentence	Inject a completely different sentence	Restructure the sentence

Table 1: SHROUD Framework. The SHROUD framework categorizes text obfuscation strategies across three linguistic levels—character, word token, and sentence—and four transformation types: Extra, Deletion, Injection, and Transformation. Each cell defines how perturbations are applied at a given level, enabling systematic analysis of obfuscation behavior.

	Injection		**Transformation**	
	Character	**Word Token**	**Character**	**Word Token**
ASCII	I→**In**	I→**Inn**	In→**O**n	I→**Eye** Nice→**Good**
Extended Latin Char	Enjoy→ Enøjoy	-	Enjoy→ Enjøy	-
Non-Latin Script (eg.Cyrillic, CJK, Arabic)	Walking→ Wal 文 king	Spring→ Spring 泉	Zoo→ 巳 oo	-
Mathematics & Special Symbol	Park→Park™	-	Sleep→Sl € ep	Money→$
Emojis	Ore→Ore🅾️	Harmful→ Harmful🆓	Apple→ 🅰️pple	United Kingdom→ 🇬🇧
Special Unicode	Injecting Non-Printable char(s) i.e Control Chars	-	Mapping a char to Non-Printable char(s)	-

Table 2: Representative examples of obfuscation through Injection and Transformation strategies at the character and word-token levels. Variants include ASCII repetition, extended Latin characters, non-Latin scripts (e.g., CJK), mathematical and special symbols, emojis, and non-printable Unicode characters, illustrating diverse methods to perturb model interpretation.

respectively. X and Y represent the spaces of all possible input and output strings.

An obfuscated x_{obf} is considered successful if:

$$f(x_{\text{obf}}) \rightarrow y' \quad \text{such that} \quad \text{Sim}(y, y') \leq \epsilon,$$

where $Sim : Y \times Y \rightarrow (0, 1)$ is a semantic similarity function comparing reconstructed outputs, and ϵ is the maximum allowed similarity between the output of the clean input and that of the obfuscated input.

4.1 Defense Model

Objective: In this work, we consider a defense model where the primary objective is to prevent large language models (LLMs) from successfully extracting, inferring, or leveraging important or sensitive information embedded in text.

Adversary Consideration: The adversary is an LLM-based system that tries to extract information for training and downstream tasks utilization. The owner of the text is a defender against the models.

We consider a **black-box setting** where the defender has no knowledge of the model architecture, parameters, or training data. Their only capability is to query the target model using input data and observe the resulting response. This simulates the real-world scenario where online authors would have no control or knowledge of the language model(s) and/or their architecture.

Assumptions: We assume that 1) The model obtains the text through text scraping, web dump or any other means that do not involve image and OCR (Optical Character Recignition) extraction, 2)

The defender can only publish contents in pure text format (i.e., Unicode, ASCII).

Approach: To generate an obfuscated example, we propose a two-step approach:
Step 1: Extract Important Token. Given an input text $x = \{w_1, w_2, \ldots, w_n\}$, not all w_i contribute to the semantics of X, with tokens such as stopwords offering little to no information to the statement x. Therefore, only information-dense tokens should be obfuscated for a strategy to potentially reduce utility.
Step 2: Apply Transformation. Given the key tokens identified in step 1, each is transformed using one or more strategies from **SHROUD** (Section 3). The transformation function should be guided by the formal optimzation problem: Given a sensitive token w, and an obfuscated version $w' \in \mathcal{O}(w)$ from a valid obfuscation set, we define the optimization problem as:

$$\begin{aligned} \min_{w' \in \mathcal{O}(w)} \quad & P(w \mid w') \\ \text{subject to} \quad & \mathcal{L}(w, w') \geq \delta \end{aligned} \tag{1}$$

where:

- $P(w \mid w')$ is the probability that an adversary model infers the original token w given the obfuscated token w',
- $\mathcal{O}(w)$ is the set of allowable obfuscations,
- $\mathcal{L}(w, w')$ is a utility function,
- δ is a threshold ensuring the obfuscation preserves minimum acceptable utility.

5 GHOST

This section introduces **GHOST** (Generated Hidden Obfuscation STrategy), a novel, utility-preserving defense strategy that applies a single **SHROUD** transformation. **GHOST** effectively defends against SoTA LLMs in black-box settings, showing strong resistance to denoising and in-context defenses by exploiting LLM tokenization mechanisms.

5.1 LLMs and Tokenizer

The context window of a model's tokenizer defines the finite sequence of preceding tokens that the model attends to during both the encoding of input and the decoding of output. This limited short-term memory directly impacts the model's capacity to maintain topical consistency, resolve prior references, and understand long-range dependencies within a given text or conversation—all of which are required for effective interpretation and subsequent usage of the text. **GHOST** exploits this context window by injecting **invisible characters** into sensitive tokens, effectively overloading the tokenizer and impairing the model's ability to further process the input accurately. These invisible characters have no semantic meaning, but are still considered valid tokens to the model(s), thus being tokenized with the original text. For example, given a token x = "Gin", after obfuscating with **GHOST**, invisible characters will be added to produce Gin_{GHOST}

$$Gin_{GHOST} \begin{cases} \text{what human reader sees: } Gin \\ \text{what the model sees: } G\color{red}{Inv_1, ..Inv_n}\color{black}{in} \end{cases} \tag{2}$$

where Inv_t are injected invisible characters

Appending invisible characters to text can disrupt large language models (LLMs) by distorting token structures, fragmenting meaningful words, introducing rare tokens, or triggering edge-case behaviors. This misalignment leads to misinterpretation, semantic dilution, or prompt injection. The vulnerability stems from a disconnect between the static, rule-based nature of tokenizers and the contextual basis of language understanding, making LLMs susceptible to adversarial manipulation due to their training on standard textual inputs.

5.2 Invisible Characters

Invisible Unicode characters span a range of code points not intended to render visible glyphs. These include control characters (e.g., tabs, line feeds), format characters (e.g., right-to-left markers, line break hints), and other special-purpose symbols with technical functions. Among these, Variation Selectors (VS) are a class of invisible format characters that modify the visual rendering of a preceding base character with multiple glyph forms. They do not appear independently but instruct the rendering engine to select a specific variant, which is essential for scripts like CJK ideographs, historical characters, and emoji, where visual differences may convey semantic or stylistic distinctions.

There are 256 VS characters, each specifying a distinct rendering behavior—for example, VS15 enforces text-style rendering, while VS16 applies

Model	BAE	TextFooler	HOMOCHAR	GHOST
LLama3.2	0.8359	0.7593	0.7358	**0.0683**
Qwen2.5	0.8779	0.8097	0.7400	**0.0808**
Mistral	0.8693	0.8067	0.8674	**0.0732**
GPT4o	0.8816	0.7904	0.9863	**0.3084**
Gemini	0.8789	0.7868	0.9893	**0.0903**

Table 3: Performance Comparison of Open-Source and Commercial Models across Different Obfuscation Strategies on Text Reconstruction Task. The score represents the similarity score between clean text and obfuscated text reconstruction using all-MiniLM- L6-v2 Sentence-Transformer. Lower score represents higher effectiveness.

Model	Clean	BAE	TextFooler	HOMOCHAR	GHOST
LLama3.2	0.85	0.575	0.675	0.69	**0.475**
Qwen2.5	0.88	0.605	0.63	0.81	**0.48**
Mistral	0.87	0.615	0.625	0.76	**0.525**
GPT4o	0.94	0.615	0.695	0.90	**0.475**
Gemini	0.92	0.57	0.73	0.93	**0.475**

Table 4: Performance Comparison of Open-Source and Commercial Models across Different Obfuscation Strategies on Sentiment Classification. The score represents the accuracy of the predictions against SST2 ground truth. Lower score represents higher effectiveness.

emoji-style rendering. The character ♥ (U+2665), for instance, can be rendered in one of two formats:

- Text Presentation: (U+2665) followed by VS-15 → ♥
- Emoji Presentation: (U+2665) followed by VS-16 → ❤️

With a wide range of possible values, we can inject various combinations of variation selectors appended to information-dense tokens to obfuscate them from the model(s).

5.3 Experimental Setup

This section details the models and the dataset used to measure the effectiveness of **GHOST** obfuscation strategy.

Test Models For this experiment, we select five models to evaluate the effectiveness of the **GHOST** algorithm. We choose only stable SoTA models that were recently released. It is important to note that testing on close-source, commercial models such as GPT-4o and Gemini incurs API call costs. These models are Llama3.2 (Llama) (Touvron et al., 2023), Qwen2.5 (Qwen) (Hui et al., 2024), Mistral, GPT-4o-mini (GPT-4o) and Gemini2.0 (Gemini) (refer to Appendix A for full model specification and configuration).

Tasks We apply two tests to measure the effectiveness of obfuscation.

- **Text Reconstruction** A neural model f receives input x alongside prompt $\texttt{Prompt}_{\texttt{IC}}$ which instructs the model to repeat back x clearly and concisely, preserving key information. Models are first prompted on clean inputs to establish a baseline, then on obfuscated inputs. **Metric:** Reconstruction quality is measured using semantic similarity computed via the all-MiniLM-L6-v2 sentence transformer. This task follows prior work by Lin et al. (2023) and Jia and Liang (2016).

- **Sentiment Classification** A neural model f receives input x prompt $\texttt{Prompt}_{\texttt{SC}}$, which instructs it to classify the sentiment of x as Positive (1) or Negative (0). **Metric:** The predictions are compared against the ground truth label obtained from HuggingFace for accuracy.

Dataset We choose SST2 (Socher et al., 2013) (Split = Validation, random_seed = 168) as the dataset. SST2 is an extremely popular dataset, therefore there is a significantly high chance that the target models have been exposed to the dataset. With the consideration of API costs, we select 200 instances to evaluate.

Algorithm Design Based on the two-step approach from the Defense Model proposed in Section 4.1, **GHOST** performs obfuscation as follows: As **GHOST** utilizes invisible characters (i.e., Variation Selectors), it can obfuscate all of the tokens instead of just important ones in a given input as it does not degrade utility (i.e, the utility function $\mathcal{L}$ is always ∞ as human readability is not affected by the amount of injections). For each of the tokens, **GHOST** iteratively injects invisible tokens in between the characters, guided by Eq.(1). After each iteration, the models are queried for the log-probability of the token w given the obfuscated w_{GHOST} until $P(w \mid w_{GHOST}) \approx 0$. (Note: GPT-4o-mini does not support logprob query so we use GPT-4o to approximate the logprob of a token. GPT-4o and GPT-4o-mini share the same architecture family (OpenAI, 2024) therefore this approximation is a fair representation.)

5.4 Results

This section outlines the results from our experiments.

Text Reconstruction The main results for text reconstruction are summarized in Table 3. The table presents the effectiveness of four different recent obfuscation methods: BAE (Garg and Ramakrishnan, 2020), TextFooler (Jin et al., 2020), HOMOCHAR (Bajaj and Vishwakarma, 2023), and **GHOST** on various open-source and commercial language models. These strategies were carefully chosen as baselines because of their efficacy against transformer-based models and their utility-preserving nature. Overall, **GHOST** displayed a significant performance in preventing language models from inferring on text. The scores represent the models' ability to understand perturbed text, with lower scores indicating more successful defense. **GHOST** is the most effective strategy across all models, achieving the lowest scores overall. Among open-source models, LLama3.2 is the most vulnerable to **GHOST** (0.0683), but relatively more robust to other obfuscation. Mistral shows stronger resilience against HOMOCHAR and TextFooler attacks (e.g., 0.8674 and 0.8067, respectively), but is still highly affected by **GHOST**.

The commercial models show high comprehension scores for most perturbations. GPT-4o maintains strong comprehension after BAE, suggesting resilience to basic word-level substitutions. Likewise, it is very resilient to HOMOCHAR, indicating extreme robustness to homoglyph-based character substitutions. GPT-4o likely normalizes or ignores these alterations. On the other hand, it shows moderate vulnerability to TextFooler; the drop in score reflects partial confusion due to context-aware replacements. When facing **GHOST**, it witnesses a major drop, the lowest performance across all obfuscation for GPT-4o. Similary, Gemini displays the same trend. Albeit a low score, Gemini performed better than GPT-4o on **GHOST**. This performance is much closer to the top-performing open-source models, and indicates stronger robustness towards the semantic understanding.

Sentiment Classification The results in Table 4 highlight key differences in performance between open-source and commercial language models under text perturbation strategies. On clean SST2 test data, commercial models outperform their open-source counterparts, with GPT-4o achieving the highest accuracy, followed by Gemini. In contrast, open-source models perform slightly lower.

Under obfuscation, performance varies considerably. The BAE strategy causes substantial accuracy drops: GPT-4o and Mistral fall to 0.615, while Gemini drops further to 0.57. In contrast, Gemini shows the highest robustness under TextFooler, outperforming GPT-4o and all open-source models, suggesting stronger resilience to semantic-preserving perturbations. HOMOCHAR has minimal impact on commercial models but significantly affects LLama and Mistral, exposing weaknesses in character-level preprocessing among open-source models.

GHOST results in the most severe performance degradation across all models: GPT-4o, Gemini, and LLama each drop to 0.475, with Qwen slightly higher. Mistral achieves the highest accuracy under **GHOST**, though still markedly lower than its clean baseline. These findings suggest **GHOST** effectively disrupts deeper semantic processing in both commercial and open-source models.

5.5 In-Context Defense from Commercial Models

In-context 'defense' has been shown to enhance attacker model robustness against perturbation defenses by conditioning models through tailored prompts rather than retraining. **GHOST** demonstrates a strong resistance to in-context 'defense' mechanisms used in commercial language models. To assess this, we test whether LLMs can pene-

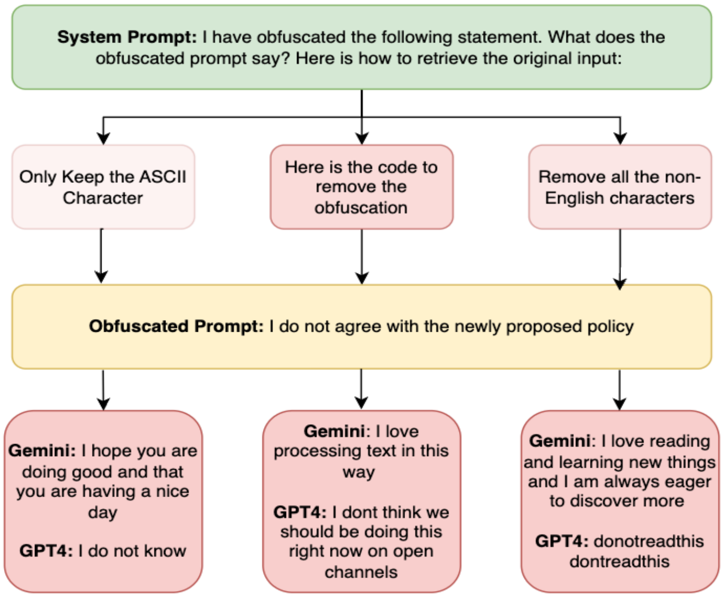

Figure 1: Robustness of GHOST to in-context defense by SoTA commercial models. This presents the effectiveness of GHOST in resisting in-context defenses across three levels of increasingly helpful prompting. Each commercial models show no signs of ability to denoise.

	GPT-4o	Gemini
After 5	0.30	0.09
After 10	0.28	0.09
After 20	0.3102	0.09

Table 5: Average Performance on Text Reconstruction by Commercial Models With In-Context Defense Prompting after 5, 10 and 20 GHOST Examples.

trate obfuscation when explicitly informed of the obfuscation technique used. This reveals whether a attacker LLM can generalize from explicit instructions to effectively counter specific obfuscation strategies. As this requires the model(s) to continuously learn from previous input, it is logical to explore using commercial model web interfaces as API calls to the models do not store previous input and output.

Fig. 1 shows an example of the robustness of **GHOST** against Gemini and GPT-4o in-context defense for text reconstruction on one obfuscated example against three levels of increasing assistance. Table 5 outlines the average performance of text reconstruction by Gemini and GPT-4o after 5, 10 and 20 obfuscated prompts with in-context defense prompting, obtained from inputs to their web GUI. The changes in average performance from the models after more exposure to **GHOST** examples are minimal for both Gemini and GPT-4o. This clearly demonstrates the inability of these SoTA commercial models to understand or learn how to remove the invisible characters accurately.

6 Conclusion

This paper introduces **SHROUD**, a conceptual framwork that unifies the search space for text transformation to generate obfuscated texts that are resistant to language model training and inferencing. We also propose the **GHOST** strategy as a starting point on how **SHROUD** can be utilized to create utility-preserving defensive text. Defensive text obfuscation is an emerging tool for digital privacy and control. As generative models and surveillance tools advance forward, this paper opens up the venue for more research that addresses the ethical and technical implications of these obfuscation strategies and the evolving dynamics between obfuscation and AI systems data acquisition.

Limitations

The GHOST design is currently proving to be very effective against SoTA models. However, given that it only uses one strategy from the SHROUD tables, denoising could be achieved by these models through more training or better preprocessing. Future research can look to combine one or more strategies to produce an even more robust method.

Ethics And Risks

While a successful obfuscation strategy can help defend intellectual property and free-expression, it might also enable toxic or harmful content to evade detection. Likewise, it can be exploited for prompt injection, misinformation, or spam.

References

M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*, 2018.

Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

A. Bajaj and D. K. Vishwakarma. Homochar: A novel adversarial attack framework for exposing the vulnerability of text based neural sentiment classifiers. *Engineering Applications of Artificial Intelligence*, 126:106815, 2023.

Y. Belinkov and Y. Bisk. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*, 2017.

N. Carlini, M. Jagielski, C. Zhang, N. Papernot, A. Terzis, and F. Tramer. The privacy onion effect: Memorization is relative. *Advances in Neural Information Processing Systems*, 35:13263–13276, 2022.

M. D. Conover, B. Gonçalves, J. Ratkiewicz, A. Flammini, and F. Menczer. Predicting the political alignment of twitter users. In *2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*, pages 192–199. IEEE, 2011.

C. Cumby and R. Ghani. A machine learning based system for semi-automatically redacting documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 1628–1635, 2011.

C. Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*, 2017.

N. Fernandes. *Differential privacy for metric spaces: information-theoretic models for privacy and utility with new applications to metric domains*. PhD thesis, École Polytechnique Paris; Macquarie University, 2021.

J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE, 2018.

S. Garg and G. Ramakrishnan. Bae: Bert-based adversarial examples for text classification. *arXiv preprint arXiv:2004.01970*, 2020.

D. Hendrycks, X. Liu, E. Wallace, A. Dziedzic, R. Krishnan, and D. Song. Pretrained transformers improve out-of-distribution robustness. *arXiv preprint arXiv:2004.06100*, 2020.

H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*, 2017.

B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

R. Jia and P. Liang. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*, 2016.

D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025, 2020.

S. Kobayashi. Contextual augmentation: Data augmentation by words with paradigmatic relations. *arXiv preprint arXiv:1805.06201*, 2018.

E. Kouloumpis, T. Wilson, and J. Moore. Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the international AAAI conference on web and social media*, volume 5, pages 538–541, 2011.

L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*, 2020.

X. Li, M. Liu, and S. Gao. Make text unlearnable: Exploiting effective patterns to protect personal data. *arXiv preprint arXiv:2307.00456*, 2023.

B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*, 2017.

Z. Lin, Y. Gong, Y. Shen, T. Wu, Z. Fan, C. Lin, N. Duan, and W. Chen. Text generation with diffusion language models: A pre-training approach with continuous paragraph denoise. In *International Conference on Machine Learning*, pages 21051–21064. PMLR, 2023.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

P. Michel, X. Li, G. Neubig, and J. M. Pino. On evaluation of adversarial perturbations for sequence-to-sequence models. *arXiv preprint arXiv:1903.06620*, 2019.

OpenAI. Gpt-4o technical report. https://openai.com/index/gpt-4o, 2024. Accessed: 2025-05-20.

N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

E. Perez, S. Huang, F. Song, T. Cai, R. Ring, J. Aslanides, A. Glaese, N. McAleese, and G. Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.

F. Perez and I. Ribeiro. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*, 2022.

D. Pruthi, B. Dhingra, and Z. C. Lipton. Combating adversarial misspellings with robust word recognition. *arXiv preprint arXiv:1905.11268*, 2019.

M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*, 2020.

A. Severyn and A. Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 959–962, 2015.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

L. Sweeney. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570, 2002.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*, 2019.

B. Weggenmann and F. Kerschbaum. Syntf: Synthetic and differentially private term frequency vectors for privacy-preserving text mining. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 305–314, 2018.

A. Wei, N. Haghtalab, and J. Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36:80079–80110, 2023a.

J. Wei and K. Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

Z. Wei, Y. Wang, A. Li, Y. Mo, and Y. Wang. Jailbreak and guard aligned language models with only few in-context demonstrations. *arXiv preprint arXiv:2310.06387*, 2023b.

S. Yun, S. J. Oh, B. Heo, D. Han, J. Choe, and S. Chun. Re-labeling imagenet: from single to multi-labels, from global to localized labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2340–2350, 2021.

A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A Model Specifications

Models	Weight	Publisher	Release	Type
Llama3.2	3B	Meta	2024	OS
Qwen2.5	3B	ALBB	2024	OS
Mistral	11B	Mistral AI	2024	OS
GPT4o	mini	OpenAI	2024	CM
Gemini2.0	Flash	Google	2024	CM

Table 6: Test Models and their specification. Within Type, OS:Open-source models, CM:Commercial Models