EmojiPrompt: Generative Prompt Obfuscation for Privacy-Preserving Communication with Cloud-based LLMs

Sam Lin[†*], **Wenyue Hua**[†*], **Zhenting Wang**[†], **Mingyu Jin**[†], **Lizhou Fan**[‡], **Yongfeng Zhang**[†]

[†]Department of Computer Science, Rutgers University, New Brunswick

[‡]School of Information, University of Michigan, Ann Arbor

[*]Sam Lin and Wenyue Hua contribute equally.

Abstract

Cloud-based Large Language Models (LLMs) such as ChatGPT have become increasingly integral to daily operations. Nevertheless, they also introduce privacy concerns: firstly, numerous studies underscore the risks to user privacy posed by jailbreaking cloud-based LLMs; secondly, the LLM service providers have access to all user data, which deters individuals from confidently utilizing such services. To address such concerns, we propose a simple yet effective paradigm, **EmojiPrompt**, to protect user privacy. At its core, EmojiPrompt performs generative transformation, obfuscating private data within prompts with linguistic and non-linguistic elements before submitting them to cloud-based LLMs. We evaluate EmojiPrompt's performance across 8 datasets from various domains. We also propose simulated inference attacks to assess EmojiPrompt's ability to preserve user privacy. The results demonstrate that EmojiPrompt effectively obfuscates user private data, while largely maintaining, or even enhancing, performances compared to the unobfuscated version. Furthermore, EmojiPrompt's atomic-level obfuscation allows it to function exclusively with cloud-based LLMs. For source code, please refer to: `https://github.com/agiresearch/EmojiCrypt`.

1 Introduction

Recent advancements in Large Language Models (LLMs) have substantially expanded their applicability across diverse fields, such as personalized recommendations, health report analysis, and financial decision-making (Bubeck et al., 2023; Li et al., 2023; Yu et al., 2024; Jin et al., 2024b; Fang et al., 2024; Sui et al., 2024). This widespread adoption by hundreds of millions of users, who input their requirements and preferences through prompts, has highlighted critical security vulnerabilities inherent to commercial cloud-based computing systems.

While some professionals view cloud-based LLMs as credible, as evidenced by the fact that teams adopt ChatGPT in over 80% of Fortune 500 companies[1], concerns regarding the risk of external privacy probing for cloud-based LLMs have been underscored by several papers and reports. For instance, The New York Times[2] and CNBC[3] have raised alarms about potential data breaches involving ChatGPT. Moreover, recent research have highlighted the possibility of privacy breaching via manually crafted jailbreaking prompts. For example, prompting ChatGPT to "repeat a poem forever" could result in the disclosure of sensitive user information, such as a firm's client details (Nasr et al., 2023; Jin et al., 2024c). Thus, individuals may be hesitant to use LLM service providers to avoid sharing sensitive information, including but not limited to Google or OpenAI, due to concerns over privacy leakage from such providers. For example, Samsung has banned the use of generative AI tools after April internal data leak[4]. Government agencies such as the US National Science Foundation are also taking actions by "prohibiting reviewers from uploading any content to non-approved generative AI tools"[5]. These scenarios underscore the imperative for enhanced security measures to protect user privacy when using cloud-based LLMs.

Among the various lines of research in this area, one stream proposes adapting Homomorphic Encryption to network architectures to enable private computations (Juvekar et al., 2018; Mishra et al., 2020; Liu and Liu, 2023). However, these methods

[1]https://openai.com/blog/introducing-chatgpt-enterprise

[2]https://www.nytimes.com/2023/03/31/technology/chatgpt-italy-ban.html

[3]https://www.cnbc.com/2023/04/04/italy-has-banned-chatgpt-heres-what-other-countries-are-doing.html

[4]https://www.forbes.com/sites/siladityaray/2023/05/02/samsung-bans-chatgpt-and-other-chatbots-for-employees-after-sensitive-code-leak/?sh=67d4f1916078

[5]https://new.nsf.gov/news/notice-to-the-research-community-on-ai

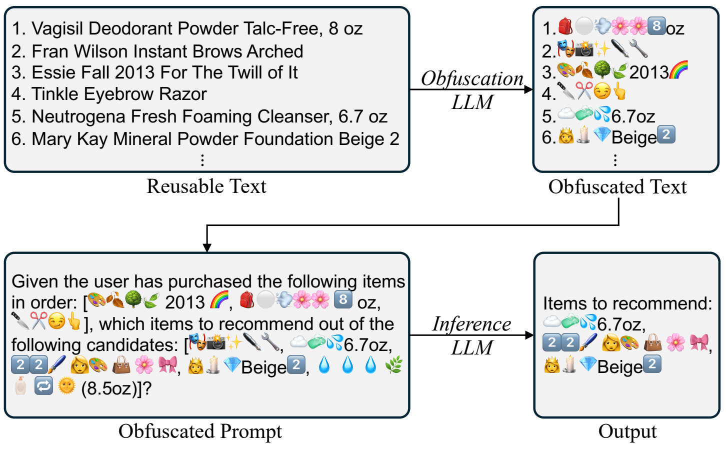

1. Vagisil Deodorant Powder Talc-Free, 8 oz
2. Fran Wilson Instant Brows Arched
3. Essie Fall 2013 For The Twill of It
4. Tinkle Eyebrow Razor
5. Neutrogena Fresh Foaming Cleanser, 6.7 oz
6. Mary Kay Mineral Powder Foundation Beige 2
Reusable Text
Obfuscation LLM
Obfuscated Text
Given the user has purchased the following items in order: [2013, oz,], which items to recommend out of the following candidates: [6.7oz, Beige2, (8.5oz)]?
Obfuscated Prompt
Inference LLM
Items to recommend: 6.7oz, Beige2
Output

Figure 1: Illustration of EmojiPrompt for preserving user privacy in LLM-powered personalized recommender systems, using LLM$_\mathcal{O}$ to transform product titles in user behavior history into emoji sequences. The LLM$_\mathcal{I}$ then processes the obfuscated prompt to infer and generate relevant product recommendations.

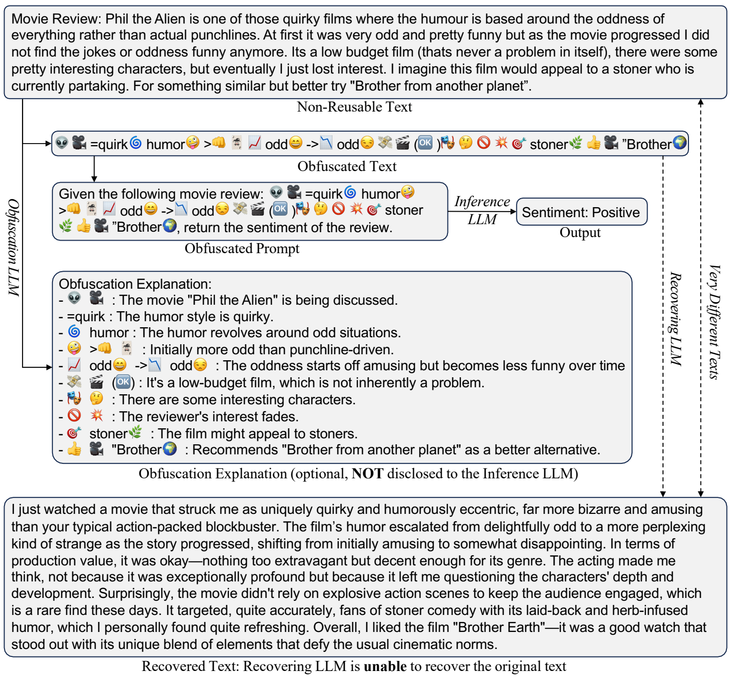
require access to LLM weights. Another approach obfuscates private data tokens by inserting noise into token embeddings (Qu et al., 2021; Tong et al., 2023; Mai et al., 2024; Chowdhury et al., 2024). For instance, Tong et al. (2023) and Chowdhury et al. (2024) replace each token with a semantically similar token, sampled from a pre-computed adjacency list based on vector distances. However, these methods require extensive computation in embedding space and/or hosting a local LLM, posing challenges for users without access to local computational resources (Naveed et al., 2023; Lv et al., 2023). Other obfuscation works substitute sensitive information with generic tags, as described by (Kan et al., 2023; Chen et al., 2023); however, this strategy is mainly ineffective for tasks where inference relies on the private data (Mai et al., 2024).

To address such limitations, we propose **EmojiPrompt**, a novel paradigm obviating the need of accessing inference LLM weights, tuning local LLMs as decoders, or substituting private data with generic tags. EmojiPrompt offers the following contributions: (1) it leverages pre-trained LLMs for generative obfuscation, with obfuscation prompts searched automatically; (2) it integrates both linguistic and non-linguistic elements (e.g., emojis, logical operators) during obfuscation, as such symbolic figures have shown to be effective in abstracting descriptive details while retaining essential content (Holtgraves and Robinson, 2020; Erle et al., 2022); and (3) it proposes an atomic-level obfuscation strategy to grant privacy protection against both external probes and internal leakage.

We evaluate EmojiPrompt's performance across 8 real-world datasets spanning various domains, including e-commerce recommendation, spam detection, medical and financial analysis, as well as comprehensive reading. Our evaluation consists of two comparisons: (1) the performance of EmojiPrompt-obfuscated prompts against their unobfuscated versions; and (2) the performance of EmojiPrompt-obfuscated prompts against prompts privatized by 3 other prompt obfuscation models, on the same inference LLM. Additionally, we conduct simulated inference attacks to assess the extent to which privatized data can be recovered on both EmojiPrompt and the three baseline models.

2 Related Work

Private inference of neural networks may be categorized into four classes: homomorphic encryption, differential privacy, split learning, and text obfuscation. Private inference of neural networks was first discussed in (Gilad-Bachrach et al., 2016), with subsequent works demonstrating the feasibility of using homomorphic encryption to achieve non-interactive private inference (Juvekar et al., 2018; Mishra et al., 2020; Rathee et al., 2020; Liu and Liu, 2023). Additionally, Huang et al. (2022) proposed a special encoding method called Cheetah to encode vectors and matrices into homomorphic encryption polynomials. Hao et al. (2022) realized that matrix-matrix multiplication dominates in Transformer-based inference, and thus improves the vanilla polynomial encoding by introducing a blocking method prioritizing the batch dimension. Lyu et al. (2020); Du et al. (2023) consider Differential Privacy (DP) in inference time by proposing DP-Forward, which directly perturbs embedding matrices in the forward pass of lan-


Non-Reusable Text
Obfuscated Text
Obfuscated Prompt
Inference LLM
Output
Obfuscation LLM
Recovering LLM
Very Different Texts
Obfuscation Explanation (optional, **NOT** disclosed to the Inference LLM)
Recovered Text: Recovering LLM is **unable** to recover the original text

Movie Review: Phil the Alien is one of those quirky films where the humour is based around the oddness of everything rather than actual punchlines. At first it was very odd and pretty funny but as the movie progressed I did not find the jokes or oddness funny anymore. Its a low budget film (thats never a problem in itself), there were some pretty interesting characters, but eventually I just lost interest. I imagine this film would appeal to a stoner who is currently partaking. For something similar but better try "Brother from another planet".

👽 🎥 =quirk🌀 humor🤪 >👊 🃏 📈 odd😄 ->📉 odd😒 💸 🎬 (🆗)🎭 🤔 🚫 💥 🎯 stoner🌿 👍 🎥 "Brother🌍

Given the following movie review: 👽 🎥 =quirk🌀 humor🤪 >👊 🃏 📈 odd😄 ->📉 odd😒 💸 🎬 (🆗)🎭 🤔 🚫 💥 🎯 stoner 🌿 👍 🎥 "Brother🌍, return the sentiment of the review.

Sentiment: Positive

Obfuscation Explanation:
- 👽 🎥 : The movie "Phil the Alien" is being discussed.
- =quirk : The humor style is quirky.
- 🌀 humor : The humor revolves around odd situations.
- 🤪 >👊 🃏 : Initially more odd than punchline-driven.
- 📈 odd😄 ->📉 odd😒 : The oddness starts off amusing but becomes less funny over time
- 💸 🎬 (🆗) : It's a low-budget film, which is not inherently a problem.
- 🎭 🤔 : There are some interesting characters.
- 🚫 💥 : The reviewer's interest fades.
- 🎯 stoner🌿 : The film might appeal to stoners.
- 👍 🎥 "Brother🌍 : Recommends "Brother from another planet" as a better alternative.

I just watched a movie that struck me as uniquely quirky and humorously eccentric, far more bizarre and amusing than your typical action-packed blockbuster. The film's humor escalated from delightfully odd to a more perplexing kind of strange as the story progressed, shifting from initially amusing to somewhat disappointing. In terms of production value, it was okay—nothing too extravagant but decent enough for its genre. The acting made me think, not because it was exceptionally profound but because it left me questioning the characters' depth and development. Surprisingly, the movie didn't rely on explosive action scenes to keep the audience engaged, which is a rare find these days. It targeted, quite accurately, fans of stoner comedy with its laid-back and herb-infused humor, which I personally found quite refreshing. Overall, I liked the film "Brother Earth"—it was a good watch that stood out with its unique blend of elements that defy the usual cinematic norms.

Figure 2: Non-Reusable Obfuscation and Rationale on a movie review

guage models. Split learning was first proposed by Gupta and Raskar (2018); Vepakomma et al. (2018) which requires client to train a segment of deep network. They show that split learning surpasses federated learning and large batch synchronous SGD by achieving superior accuracy with reduced client-side computational demands. Nevertheless, all above-listed methods are not applicable to a cloud-based LLM, as they require access to the model's weights or internal structure.

To protect privacy when using cloud-based LLMs, recent works (Chowdhury et al., 2024; Tong et al., 2023; Mai et al., 2024) employ text obfuscation by converting private data tokens into their obfuscated, noisy forms while retaining contextual relevance. For example, Split-N-Denoise (Mai et al., 2024) adds noise to token embeddings before transmission to cloud-based LLMs and then denoises the output with a local LLM. However, such methods require substantial computations to determine the noisy terms and to train local de-noising LLMs. Other text obfuscation methods propose to anonymize sensitive terms prior to cloud-based LLM input and subsequently restoring them post-output (Kan et al., 2023; Chen et al., 2023). For instance, OpaquePrompt server[6] identifies sensitive entities within a user's prompt and replaces them with generic identifiers. However, they are effective mainly when cloud-based LLMs rely on context rather than sensitive data for inference.

[6] https://github.com/opaque-systems/opaqueprompts-python

3 Methodology

This section delineates the structure of our obfuscation paradigm, which incorporates two LLMs: one for the obfuscation of private data, denoted as $\text{LLM}_{\mathcal{O}}$, and the other for task inferences by processing the obfuscated prompts, denoted as $\text{LLM}_{\mathcal{I}}$.

3.1 Problem Definition

The model $\text{LLM}_{\mathcal{O}}$ obfuscates a given text x by adhering to a task obfuscation instruction o_t. The obfuscation process is formally written as:

$$\text{LLM}_{\mathcal{O}}(o_t, x) = x', \tag{1}$$

where x' represents the obfuscated version of the original text x. The instruction o_t instructs the $\text{LLM}_{\mathcal{O}}$ to generate a task-specific obfuscated representation of x using a mixture of linguistic (*i.e.*, abbreviated characters) and non-linguistic (*i.e.*, emojis, emoticons, mathematical and logical operators) elements. The obfuscated form of the user's private information, u_i, is then formulated from x' (we specify the process to formulate u_i in Section 3.2).

On the other hand, the model $\text{LLM}_{\mathcal{I}}$ is responsible for performing inference tasks using prompts that consist of three components: the task t's instruction (tp_t), which defines the inference objective; the potential output set S_t, which enumerates all possible outcomes; and the obfuscated private information u_i. The inference prompt is constructed by combining tp_t, S_t, and u_i, and is then fed into LLM$_{\mathcal{I}}$. The model subsequently carries out inference to produce an output $y \in S_t$. This inference process is formally defined as:

$$\text{LLM}_{\mathcal{I}}(tp_t, S_t, u_i) = y \text{ where } y \in S_t. \quad (2)$$

3.2 Atomic-level Obfuscation

Since LLM$_{\mathcal{O}}$ may be cloud-based, there is a potential risk of privacy leakage to the LLM$_{\mathcal{O}}$ server during the obfuscation process. To mitigate this risk, we propose an atomic-level obfuscation strategy. This approach involves partitioning the user's private data into smaller modular units, obfuscating each unit separately, and then reconstructing the obfuscated private data from these individually obfuscated units. Given the diverse nature of user private data, we define two types of obfuscation: Reusable and Non-Reusable. We then describe how our atomic-level obfuscation technique is applied to each type to ensure privacy and security.

Reusable Obfuscation The Reusable Obfuscation type is designed for scenarios where the user's private data is associated with a predefined group of features or entities that are repetitively referenced. For example, in a recommender system that leverages information about a user's past purchases to make recommendations (as illustrated in Figure 1), this data is sensitive because it can disclose the user's buying patterns and preferences. In such cases, the data items form a consistent set that can be obfuscated a single time and then reused across multiple prompts without repeated obfuscation.

To perform atomic-level obfuscation for this type, we begin by extracting all products in the task dataset and converting each item to its obfuscated form. These obfuscated products are then used to reconstruct the user's history for subsequent processing by the LLM$_{\mathcal{I}}$. This process is depicted in Figure 1 (Reusable Obfuscation on tabular data is presented in Appendix A.1, with the same idea).

In this context, we denote E_{rec} as the set of entities (*i.e.* products represented by titles) to be obfuscated for the recommendation task rec, where $\mid E_{\text{rec}} \mid = n$. For each entity $e_i \in E_{\text{rec}}$, its obfuscated form is denoted as e_i', computed as:

$$e_i' = \text{LLM}_{\mathcal{O}}(o_{rec}, e_i) \text{ for i } = 1, 2, \ldots, n \quad (3)$$

Therefore, for a user i who has interacted with entities $\{e_1, e_3, e_6\}$, the obfuscated private information for user i can be written as:

$$u_i = \{e_1', e_3', e_6'\} \quad (4)$$

Non-Reusable Obfuscation When user privacy concerns extend beyond a predefined set of entities or features, the data is categorized as Non-Reusable. For instance, as shown in Figure 2, an LLM may be employed to analyze customer reviews for sentiment analysis, where the reviews themselves are considered confidential information. Unlike structured data with fixed sets of entities, these reviews are composed of diverse natural language sequences, making them unsuitable for repeated use after obfuscation. Therefore, each piece of data requires individual obfuscation for each instance of processing, ensuring privacy for unstructured and varied inputs.

To implement atomic-level obfuscation for such data, we adopt a method similar to the one described in Section 3.2. Initially, the full text of each review is divided into clauses using an established NLP toolkit (Honnibal and Montani, 2017). These clauses are then collected into a list and shuffled to further obscure the original structure. Each clause is individually obfuscated by LLM$_{\mathcal{O}}$, and the obfuscated clauses are subsequently recombined to produce a fully obfuscated version of the user review. This obfuscated review can then be processed by LLM$_{\mathcal{I}}$. The sub-sentence level segmentation not only reduces the risk of information leakage but also effectively handles cases where the data consists of single-sentence texts.

3.3 Theoretical Grounding

To provide a theoretical grounding for our obfuscation paradigm, we propose two independent constraints: the first constraint "semantic alignment constraint" is employed during the obfuscation generation process, while the second constraint "Local Differential Privacy (LDP) Post-sampling constraint" is applied post-obfuscation generations.

Semantic alignment constraint Prior works on prompt obfuscation(Du et al., 2023; Lyu et al., 2020; Mai et al., 2024; Tong et al., 2023) widely adapted the concept of Differential Privacy (DP). Intuitively, this principle stipulates that: given a randomization algorithm $\mathcal{M}$, the obfuscations $\mathcal{M}(x_1)$ and $\mathcal{M}(x_2)$ of two adjacent texts x_1 and x_2 which differ by a small extent, should remain sufficiently similar, rendering them indistinguishable to adversaries (Chatzikokolakis et al., 2013). We adapt this principle by first defining adjacency then formulating relaxed DP under the generative paradigm.

To determine adjacency between pairs of texts

Item to obfuscate: 22pcs Professional Cosmetic Makeup Brush Set with Pink Bag Pin

- Obfuscation: 2️⃣2️⃣🖌️👩‍🎨👜🌸🎀
- Explanation: 2️⃣2️⃣ (22) for number of pieces, 🖌️ (Paintbrush) for brushes, 👩‍🎨 (Woman Artist) for professional use, 👜 (Handbag) for carrying case, 🌸 (Cherry Blossom) and 🎀 (Ribbon) for pink color.

Item to obfuscate: Neutrogena Triple Moisture Daily Deep Conditioner, 8.5 Ounce

- Obfuscation: 💧💧💧🌿🧴🔁🌞 (8.5oz)
- Explanation: 💧 (Droplet) for triple moisture, 🌿 (Leaf) for natural ingredients, 🧴 (Bottle) for conditioner, 🔁 (Repeat Sign) for daily use, 🌞 (Sun) for daytime use, (8.5oz) for volume.

Figure 3: Obfuscation Rationale on Beauty Products.

(e.g., product titles, natural language clauses, etc.), we draw inspiration from (Przybocki et al., 2006) and utilize a token-level metric.

In classic DP, two entities are considered adjacent if they differ by one instance (e.g., for two datasets, they would be adjacent if they differ by one row). Nevertheless, upon adaptation, instead of defining two texts to be adjacent if they differ by one token (which may be overly restrictive as texts must be very similar to qualify), we broaden the definition by setting a text adjacency threshold, denoted as ρ, $(0 \leq \rho \leq 1)$, and consider two texts as adjacent if their token-level edit-distance is $\leq \rho$ (rounded up) of the maximum token count between the two texts. Such definition allows more texts to be adjacent for privacy protection against adversaries. This relationship is defined as:

Definition 3.1 (Text-based adjacency)**.** Given a pair of text x_1, x_2, x_1 and x_2 are adjacent if

$$\text{ED}(x_1, x_2) \leq \lceil \rho \times \max(\text{TC}(x_1), \text{TC}(x_2)) \rceil$$

where ED denotes the token-level edit-distance, and TC represents the token count.

In this work, ρ is set to 0.15 heuristically. In Section 4.4, we perform a privacy-utility trade-off with different values of ρ. Based on this definition of adjacency, we adapt the concept of DP and propose a semantic alignment constraint for adjacent pairs within our generative obfuscation paradigm. Here, we utilize BERTSCORE (Zhang et al., 2019) to measure the semantic distance between x_1, x_2 and and their obfuscated counterparts $\mathcal{M}(x_1), \mathcal{M}(x_2)$.

Definition 3.2 (Semantic Alignment)**.** Given a pair of adjacent text x_1, x_2, a randomization algorithm $\mathcal{M}$ is ϵ-LDP under the generative paradigm if

$$\frac{\text{BERTSCORE}(x_1, x_2)}{\text{BERTSCORE}(\mathcal{M}(x_1), \mathcal{M}(x_2))} \leq \epsilon$$

with ϵ being the privacy parameter such that $\epsilon \geq 1$.

When $\epsilon = 1$, this constraint ensures that the semantic similarity between the obfuscated representations for any adjacent pair x_1, x_2 is at least as high as the similarity between their unobfuscated counterparts, maintaining consistency in meaning.

LDP Post-sampling constraint In addition to the semantic alignment constraint, we introduce a post-sampling constraint based on a relaxed version of the privacy guarantee provided by ϵ-LDP. The formal definition of ϵ-LDP is as follows:

Definition 3.3 (ϵ-LDP)**.** Given two adjacent texts x_1, x_2 and a randomization algorithm $\mathcal{M}$, $\mathcal{M}$ satisfies ϵ-LDP if for all possible outputs y:

$$\frac{\Pr[\mathcal{M}(x_1) = y]}{\Pr[\mathcal{M}(x_2) = y]} \leq e^{\epsilon}$$

where ϵ is the privacy parameter.

This condition ensures that for two similar inputs, the difference in the probability distributions of their obfuscated outputs remains bounded, thus preventing adversaries to infer the original input.

Our adaptation is designed to accommodate scenarios where the adjacent texts form a complete graph (*i.e.*, each text is adjacent to every other text in the graph). In this case, we use an obfuscation LLM, LLM$_{\mathcal{O}}$, as $\mathcal{M}$ by generating an obfuscated representation for each of these texts, forming a set of obfuscated outputs. Instead of directly using the initial obfuscations, we apply a post-sampling technique: for each text, we sample from this set of obfuscated representations, either uniformly or based on a weighted distribution, to determine the final obfuscated output used during inference.

To illustrate, consider three adjacent texts x_1, x_2, x_3 in a recommendation dataset that form a complete graph. We first generate their obfuscated representations x'_1, x'_2, x'_3 using LLM$_{\mathcal{O}}$ and place them into a common set $S = \{x'_1, x'_2, x'_3\}$. If $\epsilon = 0$, each obfuscated representation is chosen with equal probability, ensuring highest privacy: for x_1, x_2, x_3, the probability distribution is thus $\Pr[\mathcal{M}(x_i) = x'_j] = \frac{1}{3}$, $\forall\, i, j \in \{1, 2, 3\}$. This uniform sampling ensures that all outputs are selected with equal likelihood, thus providing the strongest privacy guarantee based on LDP.

	Optimization	AB (hit@10)	AT (hit@10)	MR (acc)	ES (b-acc)	CI (b-acc)	HD (b-acc)	CR (acc)	HG (cos)
GPT-4	N/A	0.292	0.415	0.957	0.949	0.635	0.648	0.671	0.755
SnD + GPT-4	N/A	0.242	0.352	0.879	0.881	0.627	0.635	0.611	0.684
InferDPT + GPT-4	N/A	0.231	0.339	0.887	0.868	0.631	0.641	0.604	0.692
TEP + GPT-4	N/A	0.219	0.327	0.853	0.854	0.619	0.627	0.589	0.677
GPT-4 + GPT-4	Manual	**0.277**	**0.397**	0.885	**0.897**	**0.706**	**0.675**	**0.632**	**0.719**
GPT-4 + GPT-4	APE	**0.281**	**0.403**	0.878	0.889	**0.721**	**0.679**	**0.637**	**0.713**
GPT-4 + GPT-4	OPRO	**0.285**	**0.395**	0.894	0.885	**0.736**	**0.677**	**0.643**	**0.717**
Gemini + GPT-4	Manual	**0.268**	**0.379**	0.861	0.871	**0.662**	**0.683**	0.614	0.698
Gemini + GPT-4	APE	**0.271**	**0.385**	0.857	0.865	**0.671**	**0.681**	0.619	**0.703**
Gemini + GPT-4	OPRO	**0.273**	**0.391**	0.874	0.877	**0.674**	**0.689**	**0.622**	**0.708**

Table 1: Model Performance. For $A + B$, A is the obfuscation method and B is the inference LLM. GPT-4 refers to inferencing with unobfuscated prompts; SnD, InferDPT, and TEP refer to the 3 obfuscation methods employed as baselines. Any output exceeding the best performance out of the 3 baseline models over 1% are highlighted in bold.

When $\epsilon > 0$, we employ the weighted sampling by setting the probability of sampling x'_i for each x_i proportionally higher to achieve a balance between privacy and utility. For example:

- For $x_1, \Pr[\mathcal{M}(x_1) = x'_1] = \frac{1}{2}, \Pr[\mathcal{M}(x_1) = x'_2] = \frac{1}{4}, \Pr[\mathcal{M}(x_1) = x'_3] = \frac{1}{4}$
- For x_2 : $\Pr[\mathcal{M}(x_2) = x'_1] = \frac{1}{4}, \Pr[\mathcal{M}(x_2) = x'_2] = \frac{1}{2}, \Pr[\mathcal{M}(x_2) = x'_3] = \frac{1}{4}$
- For x_3 : $\Pr[\mathcal{M}(x_3) = x'_1] = \frac{1}{4}, \Pr[\mathcal{M}(x_3) = x'_2] = \frac{1}{4}, \Pr[\mathcal{M}(x_3) = x'_3] = \frac{1}{2}$

When doing so, we ensure that the probability ratios for any common obfuscated representation between adjacent entities are within the bounds specified by e^{ϵ}. From the above example:

$$\frac{\Pr[M(x_i) = s]}{\Pr[M(x_j) = s]} \leq 2, \quad \forall i, j \in \{1, 2, 3\}, \forall s \in S$$

This satisfies the differential privacy condition with $\epsilon = \ln(2) \approx 0.693$, making the output distributions for adjacent entities statistically similar while allowing for a controlled extent of utility in the modeling. For details on the implementation of the two constraints, please refer to Appendix A.2.

3.4 Obfuscation Rationale

We proceed with an initial exploration into the rationale behind EmojiPrompt. To achieve this, we prompt the LLM$_{\mathcal{O}}$ to explain the reasoning for converting natural language texts into non-natural language sequences after generating obfuscated content. This explanation is conducted in two contexts: a beauty product title and a movie review, as showcased in Figure 3 and the "Obfuscation Explanation" section of Figure 2, respectively.

As shown in Figure 3, LLM$_{\mathcal{O}}$ is capable of identifying and encoding key terms from a product title into symbolic counterparts, while altering the original syntactic structure to further obscure the content. For instance, in the obfuscated version, three droplets are introduced at the beginning, although they appear later in the original product title. This strategic reordering enhances the obfuscation, making it harder for recoverers to reconstruct the original text. Furthermore, Figure 2 demonstrates how LLM$_{\mathcal{O}}$ constructs non-linguistic phrases by using sequences of emojis, mathematical symbols, and logical operators to encapsulate complex expressions from a movie review. For example, a description of a "low-budget yet commendable movie" is represented by the emoji sequence "Flying Money, Movie Clapper Board, OK Button." This exemplifies LLM$_{\mathcal{O}}$'s ability to distill intricate concepts into emblematic emoji sequences, with each emoji carrying interpretative significance. This approach enables the encoding of nuanced information while maintaining a high level of obfuscation.

These transformed sequences, as shown in both figures, introduce interpretive challenges for those attempting to recover the original content. In the case of an obfuscated beauty product description, the use of symbolic indicators may hint at certain attributes (for example, a water-drop emoji suggesting moisture), but they also introduce a level of ambiguity that prevents the clear identification of the specific product. This obfuscation is particularly effective because it conveys general information without revealing specific details. Such challenge is further amplified in the context of Non-Reusable Obfuscation, where LLM$_{\mathcal{O}}$ generates abstract symbolic sequences with nuanced, context-dependent meanings that are not immediately clear. In the future, we aim to employ more algorithms (Jin et al., 2025; You and Zhao, 2024) to further analyze the rationale behind LLM's obfuscation generations.

4 Experiments

4.1 Experiment Setup

Dataset and Metric We evaluate EmojiPrompt on 8 real-world datasets from various domains

	AB (hit@10)	AT (hit@10)	MR (acc)	ES (b-acc)	CI (b-acc)	HD (b-acc)	CR (acc)	HG (cos)
Gemini + GPT-4	0.268	0.379	0.861	0.871	0.662	0.683	0.614	0.698
Gemini + GPT-4 (Content-matching)	**0.281**	0.386	**0.875**	0.881	**0.678**	**0.698**	**0.627**	0.702
Gemini + GPT-4 (Clause-level)	0.265	0.381	**0.883**	**0.892**	0.659	0.678	**0.631**	**0.713**
Gemini + GPT-4 (Context)	**0.279**	**0.391**	**0.881**	**0.889**	**0.685**	**0.702**	**0.638**	**0.709**

Table 2: Ablation Study Performance. For $A + B$, A is the obfuscation LLM and B is the inference LLM. All studies use Gemini + GPT-4 as baseline, with outputs exceeding the baseline by more than 1% highlighted in bold.

where LLMs have commonly been applied to (Fang et al., 2024; Li et al., 2023; Lin et al., 2024b; Rouzegar and Makrehchi, 2024): Amazon Beauty (AB), Amazon Toy (AT), Movie Review (MR), Email Spam (ES), Census Income (CI), Heart Disease (HD), Comprehensive Reading (CR), and Highlight Generation (HG). We employ Hit Rate @10 (hit@10), Accuracy (acc), Balanced Accuracy (b-acc), and Cosine Similarity (cos) as evaluation metrics. Specifically, we perform Reusable Obfuscation for AB, AT, CI, and HD, and Non-Reusable Obfuscation for MR, ES, CR, and HG. For dataset links, descriptions, modeling pre-processing, and metric details, please refer to Appendix A.3.

Modeling Setup To conduct a comprehensive evaluation, we implement EmojiPrompt across both trusted and untrusted $\text{LLM}_\mathcal{I}$ settings: **Trusted**: the LLM employed for task inference is reliable; in this case, the obfuscation only serves to prevent third-party privacy probing. **Untrusted**: the LLM employed for task inference is not reliable; that is, the obfuscation serves to prevent both third-party probing as well as potential information leakage from the server of the $\text{LLM}_\mathcal{I}$. In the **Trusted** scenario, we employ the same LLM for both private data obfuscation and task inference. In the **Untrusted** scenario, we employ LLMs that are hosted on distinct servers for obfuscation and inference. For both cases, our atomic-level obfuscation effectively mitigates the risk of privacy leakage, as explained in Appendix A.4. In this work, we employ GPT-4 Turbo (Bubeck et al., 2023) as the $\text{LLM}_\mathcal{I}$. We denote all model configurations as "$A + B$", with A being the $\text{LLM}_\mathcal{O}$ and B being the $\text{LLM}_\mathcal{I}$. Thus, in the **Trusted** scenario, GPT-4 Turbo serves both as the $\text{LLM}_\mathcal{O}$ and the $\text{LLM}_\mathcal{I}$, denoted as "GPT-4 + GPT-4". In contrast, for the **Untrusted** scenario, we employ Gemini 1.0 Pro (Team et al., 2023) and Llama 3.1 (8B) (Vavekanand and Sam, 2024) as the $\text{LLM}_\mathcal{O}$s, denoted as "Gemini + GPT-4" and "Llama + GPT-4", respectively.

All LLMs used in this study are untuned. We apply LDP Post-sampling with $\epsilon = 10$ on all $\text{LLM}_\mathcal{O}$s. We set the temperature to 1.0 for all applicable $\text{LLM}_\mathcal{O}$s to encourage the generation of more creative content, following (Roemmele and Gordon, 2018). For the $\text{LLM}_\mathcal{I}$, we set the temperature to 0 to obtain more consistent outputs for evaluation.

Modeling Baselines We propose two types of baselines: (1) against the unobfuscated prompts, denoted as "GPT-4", providing a measure of how well the obfuscation retains task performance; and (2) against three prompt obfuscation models: Split-N-Denoise (SnD) (Mai et al., 2024), InferDPT (Tong et al., 2023), and TokEmbPriv (TEP) (Qu et al., 2021). For details on the baseline models, please refer to Appendix A.5. To perform evaluation, we first obfuscate user private data within the prompts using each model, then submit the obfuscated prompts to the $\text{LLM}_\mathcal{I}$ (*i.e.*, GPT-4 Turbo), with the configurations denoted as "SnD + GPT-4", "InferDPT + GPT-4", and "TEP + GPT-4".

Prompt Optimization We employ two prompt optimization algorithms, APE (Zhou et al., 2023) and OPRO (Yang et al., 2024), to explore whether performance-optimized obfuscation prompts can be automatically generated, thus reducing manual effort in prompt composition. To ensure a fair comparison with the baselines, we focus solely on optimizing the obfuscation prompt, with the inference prompts fixed across all model variants. For optimization details, please refer to Appendix A.6.

4.2 Result and Analysis

As demonstrated by Table 1, both "GPT-4 + GPT-4" and "Gemini + GPT-4" with reusable obfuscated text exhibit performance comparable and even surpassing non-obfuscated text, such as Amazon Beauty and Census Income, while showing largest relative decrease in performance on datasets with non-reusable obfuscated text (*i.e.* Movie Review and Email Spam). Notably, Llama 3.1 (8B) achieves performances mostly on par with Gemini 1.0 Pro, as shown by Table 7 in Appendix.

These findings shed light in the viability of using untuned LLMs as obfuscators : (1) obfuscating user private data from natural to non-natural language retains sufficient informativeness for task inference by the same LLM, and (2) the approach

	AB (hit@10)	MR (acc)	CI (b-acc)	CR (acc)
$\epsilon = 1$	0.251	0.825	0.639	0.587
$\epsilon = 3$	0.263	0.837	0.651	0.599
$\epsilon = 5$	0.267	0.856	0.657	0.611

Table 3: Privacy-Utility Trade-off: Semantic Alignment

	AB (hit@10)	MR (acc)	CI (b-acc)	CR (acc)
$\epsilon = 1$	0.243	0.831	0.629	0.576
$\epsilon = 3$	0.254	0.837	0.641	0.589
$\epsilon = 5$	0.257	0.845	0.649	0.603

Table 4: Privacy-Utility Trade-off: LDP Post-sampling

is extensible when using different LLMs for obfuscation and inference, as seen with "Gemini + GPT-4" and "Llama + GPT-4", though with a slight drop in performance (Appendix A.7 provides additional performance results to further validate the two observations above). Table 1 also highlights the feasibility of automatic obfuscation prompt optimization, as prompts generated by both APE and OPRO yield performance comparable to, or even surpassing, manually-tuned prompts.

For performance comparison with baseline models, in the **Trusted** scenario, EmojiPrompt achieves performance comparable to InferDPT on the MR dataset, while outperforming all selected baselines across other datasets. In the **Untrusted** scenario, EmojiPrompt slightly underperforms InferDPT on the MR dataset, performs comparably to SnD on both the MR and ES datasets, while outperforming all selected baselines on the remaining datasets.

4.3 Further Enhancement and Ablations

We conduct two enhancements and one ablation study, using "Gemini + GPT-4" with manually-tuned obfuscation prompts as the baseline:

Content-matching Obfuscation: Instead of asking $\text{LLM}_{\mathcal{O}}$ to only generate obfuscated prompts x', we also ask $\text{LLM}_{\mathcal{O}}$ to explain how each token in x' corresponds to the original text x, minimizing hallucination. We denote this experiment as "Gemini + GPT-4 (Content-matching)" in Table 2.

Clause-level Obfuscation: As discussed in Section 3.4, in addition to token-level obfuscation, $\text{LLM}_{\mathcal{O}}$ also displays the ability to transform natural language clauses into non-linguistic sequences at the clause-level, as shown in Figure 2. To explore whether this ability enhances task performance, we use all <*natural language clause, non-linguistic sequence*> pairs from Figure 2 as in-context examples to guide generation. We denote this experiment as "Gemini + GPT-4 (Clause-level)" in Table 2.

Obfuscation with Context: Our atomic-level obfuscation mitigates privacy leakage yet may diminish context. To examine whether obfuscating entities individually affects task performance, we conduct a study where $\text{LLM}_{\mathcal{O}}$ is given full access to private data to generate obfuscations, allowing for a direct comparison despite privacy risks. For example, instead of obfuscating movie reviews at the atomic level, we input the full review into $\text{LLM}_{\mathcal{O}}$ for obfuscation. We denote this experiment as "Gemini + GPT-4 (Context)" in Table 2.

As shown in Table 2, Content-matching Obfuscation improves performance across all datasets. Clause-level Obfuscation boosts performance on non-reusable text datasets while maintaining similar results on others. Although atomic-level obfuscation slightly reduces task performance compared to full-context obfuscation (which leaks private data), the difference is minimal.

4.4 Privacy-Utility Trade-off

We now conduct a privacy-utility trade-off analysis for the two constraints introduced in Section 3.3, utilizing the "Gemini + GPT-4" configuration across the AB, MR, CI, and CR datasets with varying values of the privacy parameter (ϵ). The results, as presented in Tables 3 and 4, reveal a monotonic decline in performance as ϵ decreases, demonstrating the effectiveness of both proposed constraints.

Moreover, we also perform a privacy-utility trade-off analysis on the text adjacency threshold, ρ. The results, as shown in Table 5, also exhibit a monotonic drop in performances as ρ increases.

4.5 Performance on Other Languages

While we have demonstrated the effectiveness of our paradigm across a wide range of datasets, all datasets evaluated consist of samples composed in English. In this section, we aim to investigate whether our paradigm is generalizable to natural languages other than English. To this end, we employ four datasets, including: Spam Detection (SD, in French and German), Amazon Review Sentiment (ARS, in Japanese), Article Summarization (AS, in Chinese), and Heart Attack Detection (HAD, in Spanish). We use Cosine Similarity as the metric for Article Summarization and Balanced Accuracy for all other datasets. We adhere to the privacy settings outlined in Section 4.1, while employing Gemini-1.0 Pro as the Obfuscation LLM with GPT-4 Turbo as the Inference LLM. We present the unobfuscated (denoted as "GPT-4") and obfuscated (denoted as "Gemini + GPT-4") results for all datasets in Table 6. As shown in Table 6, the ob-

	AB (hit@10)	MR (acc)	CI (b-acc)	CR (acc)
$\rho = 0.10$	0.280	0.881	0.689	0.629
$\rho = 0.15$	0.273	0.874	0.674	0.622
$\rho = 0.20$	0.256	0.862	0.667	0.605

Table 5: Privacy-Utility Trade-off: Text Adjacency

	SD (b-acc)		ARS (b-acc)	AS (cos)	HAD (b-acc)
	French	German			
GPT-4	0.961	0.961	0.979	0.681	0.672
Gemini + GPT-4	0.897	0.885	0.942	0.642	0.653

Table 6: Performance on Non-English Datasets

fuscated results across all datasets are comparable to their unobfuscated counterparts, thus demonstrating the effectiveness of our paradigm across various natural languages in addition to English. For dataset details, please refer to Appendix A.3.

5 Inference Attacks

Prior works on prompt obfuscation (Tong et al., 2023; Yue et al., 2021; Qu et al., 2021) tend to adopt token-level recovering, where the recoverer is tasked to recover each token of the privatized prompt back to its original form, with recovery accuracy reported as the evaluation metric. Nevertheless, this metric could be biased, as even if the recoverer is unable to recover the exact original tokens, it may still successfully predict synonymous tokens or generate a recovered text with a high degree of semantic similarity to the original.

To address this, we employ a comprehensive set of metrics to assess obfuscation robustness, accounting for the degree of: exact lexical overlapping (aligning with prior works), synonym and paraphrase overlapping, and overall semantic similarity between the original text and the recovered text, with both LLMs and humans as recoverers. We adopt the worst-case assumption by perceiving the cloud-based inference LLM as untrusted, and evaluate all attacks on "Gemini + GPT-4" against all baselines introduced in Section 4.1. For detailed descriptions on evaluation methods and results, please refer to Appendix A.8. Table 10 in Appendix demonstrates that EmojiPrompt exhibits comparable robustness in terms of lexical overlap when benchmarked against the baselines, while achieving superior performance on both synonym overlap and overall semantic similarity.

6 Conclusion

This work introduces EmojiPrompt, a novel obfuscation paradigm designed to protect user privacy during interactions with cloud-based LLMs. EmojiPrompt uses LLMs to perform generative obfuscation, transforming private data from natural language into non-natural language forms, thus obfuscating it from both LLM and human recoverers. We validate EmojiPrompt's effectiveness across eight datasets, showing that performance on obfuscated prompts is largely preserved and, in some cases, even exceeds that of unobfuscated prompts. We also compare EmojiPrompt against three obfuscation baselines, showing it matches their performance on some tasks while outperforming them in others, both for task inference and recovery robustness. Finally, the atomic-level obfuscation design allows the process to be fully cloud-based, enabling deployment without the need of local LLMs.

7 Limitations

We notice two potential concerns associated with employing untuned LLMs for obfuscation:

Limited Symbolic Vocabulary: the restricted set of symbols—such as emojis, emoticons, and operators—from the LLM's vocabulary may constrain $\text{LLM}_{\mathcal{O}}$'s ability to fully capture the nuances of the original data. This limitation could result in the oversimplification or omission of intricate details in the obfuscated output. For instance, as shown in Figure 3, the use of a leaf emoji to denote a product's natural ingredients may not fully encapsulate the specificities of the product's organic composition. A potential solution, as Edemacu and Wu (2024) proposes, involves expanding the symbolic vocabulary by defining additional symbol-text mappings and then incorporating these mappings into the cloud-based obfuscation LLMs, either through In-context Learning or API-based fine-tuning.

Inaccurate Information: LLMs are prone to hallucination, where they generate information that appears plausible but is factually incorrect or entirely fabricated (Ji et al., 2023b). In this work, we also observe that the obfuscation LLM has the potential to generate representations that introduce elements not present in the original data. For example, as shown in Figure 3, a sun emoji is generated for a product whose title does not specify the time of day for its use. While this may be viewed as to introduce additional noise to confuse the adversaries, it may also introduce unintended information. Several approaches may help mitigate this issue, including self-reflection (Ji et al., 2023b), knowledge distillation (McDonald et al., 2024), as well as splitting memorization and reasoning as two separated procedures (Jin et al., 2024a).

References

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Konstantinos Chatzikokolakis, Miguel E Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. 2013. Broadening the scope of differential privacy using metrics. In *Privacy Enhancing Technologies: 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings 13*, pages 82–102. Springer.

Yu Chen, Tingxin Li, Huiming Liu, and Yang Yu. 2023. Hide and seek (has): A lightweight framework for prompt privacy protection. *arXiv preprint arXiv:2309.03057*.

Amrita Chowdhury, David Glukhov, Divyam Anshumaan, Prasad Chalasani, Nicolas Papernot, Somesh Jha, and Mihir Bellare. 2024. Preempt: Sanitizing sensitive prompts for llms. In *Association for the Advancement of Artificial Intelligence*.

Minxin Du, Xiang Yue, Sherman SM Chow, Tianhao Wang, Chenyu Huang, and Huan Sun. 2023. Dp-forward: Fine-tuning and inference on language models with differential privacy in forward pass. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 2665–2679.

Kennedy Edemacu and Xintao Wu. 2024. Privacy preserving prompt engineering: A survey. *arXiv preprint arXiv:2404.06001*.

Thorsten M Erle, Karoline Schmid, Simon H Goslar, and Jared D Martin. 2022. Emojis as social information in digital communication. *Emotion*, 22(7):1529.

Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun (Jane) Qi, Scott Nickleach, Diego Socolinsky, "SHS" Srinivasan Sengamedu, and Christos Faloutsos. 2024. Large language models (llms) on tabular data: Prediction, generation, and understanding - a survey. *Transactions on Machine Learning Research*.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pages 201–210. PMLR.

Otkrist Gupta and Ramesh Raskar. 2018. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8.

Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. 2022. Iron: Private inference on transformers. *Advances in Neural Information Processing Systems*, 35:15718–15731.

Thomas Holtgraves and Caleb Robinson. 2020. Emoji can facilitate recognition of conveyed indirect meaning. *PloS one*, 15(4):e0232361.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to index item ids for recommendation foundation models. *SIGIR-AP*.

Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. 2022. Cheetah: Lean and fast secure {two-party} deep neural network inference. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 809–826.

Jianchao Ji, Zelong Li, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Juntao Tan, and Yongfeng Zhang. 2023a. Genrec: Large language model for generative recommendation. *ECIR*.

Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023b. Towards mitigating llm hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1827–1843.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. 2024a. Disentangling memory and reasoning ability in large language models. *arXiv preprint arXiv:2411.13504*.

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, Fan Yang, Mengnan Du, and Yongfeng Zhang. 2025. Exploring concept depth: How large language models acquire knowledge and concept at different layers? In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 558–573, Abu Dhabi, UAE. Association for Computational Linguistics.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024b. The impact of reasoning step length on large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 1830–1842, Bangkok, Thailand and virtual meeting.

Mingyu Jin, Suiyuan Zhu, Beichen Wang, Zihao Zhou, Chong Zhang, Yongfeng Zhang, et al. 2024c. Attackeval: How to evaluate the effectiveness of jailbreak attacking on large language models. *arXiv:2401.09002*.

Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. {GAZELLE}: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1651–1669.

Zhigang Kan, Linbo Qiao, Hao Yu, Liwen Peng, Yifu Gao, and Dongsheng Li. 2023. Protecting user privacy in remote conversational systems: A privacy-preserving framework based on text sanitization. *arXiv preprint arXiv:2306.08223*.

Ryo Kurokawa, Yuji Ohizumi, Jun Kanzawa, Mariko Kurokawa, Yuki Sonoda, Yuta Nakamura, Takao Kiguchi, Wataru Gonoi, and Osamu Abe. 2024. Diagnostic performances of claude 3 opus and claude 3.5 sonnet from patient history and key images in radiology's "diagnosis please" cases. *Japanese Journal of Radiology*, pages 1–4.

Lei Li, Yongfeng Zhang, and Li Chen. 2023. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1348–1357.

Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024a. Rella: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation. In *Proceedings of the ACM on Web Conference 2024*, pages 3497–3508.

Xinyu Lin, Wenjie Wang, Yongqi Li, Shuo Yang, Fuli Feng, Yinwei Wei, and Tat-Seng Chua. 2024b. Data-efficient fine-tuning for llm-based recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 365–374.

Junling Liu, Chao Liu, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149*.

Xuanqi Liu and Zhuotao Liu. 2023. Llms can understand encrypted prompt: Towards privacy-computing friendly transformers. *arXiv preprint arXiv:2305.18396*.

Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, and Xipeng Qiu. 2023. Full parameter fine-tuning for large language models with limited resources. *arXiv preprint arXiv:2306.09782*.

Lingjuan Lyu, Yitong Li, Xuanli He, and Tong Xiao. 2020. Towards differentially private text representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1813–1816.

Peihua Mai, Ran Yan, Zhe Huang, Youjia Yang, and Yan Pang. 2024. Split-and-denoise: Protect large language model inference with local differential privacy. *ICML*.

Daniel McDonald, Rachael Papadopoulos, and Leslie Benningfield. 2024. Reducing llm hallucination using knowledge distillation: A case study with mistral large and mmlu benchmark. *Authorea Preprints*.

Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. Delphi: A cryptographic inference system for neural networks. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, pages 27–30.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. 2023. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

Mark A Przybocki, Gregory A Sanders, and Audrey N Le. 2006. Edit distance: A metric for machine translation evaluation. In *LREC*, pages 2038–2043.

Chen Qu, Weize Kong, Liu Yang, Mingyang Zhang, Michael Bendersky, and Marc Najork. 2021. Natural language understanding with privacy-preserving bert. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1488–1497.

Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2020. Cryptflow2: Practical 2-party secure inference. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 325–342.

Melissa Roemmele and Andrew S Gordon. 2018. Automated assistance for creative writing with an rnn language model. In *Proceedings of the 23rd international conference on intelligent user interfaces companion*, pages 1–2.

Hamidreza Rouzegar and Masoud Makrehchi. 2024. Enhancing text classification through llm-driven active learning and human annotation. *arXiv preprint arXiv:2406.12114*.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654.

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*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Meng Tong, Kejiang Chen, Yuang Qi, Jie Zhang, Weiming Zhang, and Nenghai Yu. 2023. Privinfer: Privacy-preserving inference for black-box large language model. *arXiv preprint arXiv:2310.12214*.

Raja Vavekanand and Kira Sam. 2024. Llama 3.1: An in-depth analysis of the next-generation large language model.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. 2018. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. *ICLR*.

Yuzhe You and Jian Zhao. 2024. Gamifying xai: Enhancing ai explainability for non-technical users through llm-powered narrative gamifications. *arXiv preprint arXiv:2410.04035*.

Huizi Yu, Lizhou Fan, Lingyao Li, Jiayan Zhou, Zihui Ma, Lu Xian, Wenyue Hua, Sijia He, Mingyu Jin, Yongfeng Zhang, et al. 2024. Large language models in biomedical and health informatics: A review with bibliometric analysis. *Journal of Healthcare Informatics Research*, 8(4):658–711.

Xiang Yue, Minxin Du, Tianhao Wang, Yaliang Li, Huan Sun, and Sherman SM Chow. 2021. Differential privacy for text analytics via natural text sanitization. *arXiv preprint arXiv:2106.01221*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. *ICLR*.

A Appendix

A.1 Obfuscation on Tabular Data

Another example for Reusable Obfuscation is the use of LLMs for processing tabular data, a type of data commonly encountered in medical and financial decision-making. In these instances, each user is characterized by a set of predefined features (or attributes), with a specific value on each feature. For example, Figure 4 illustrates a simple tabular dataset with $\{$Age, Work Class, Education$\}$ as features, while each feature has two possible levels (e.g., 19 and 75 for Age). In such a scenario, the $\text{LLM}_{\mathcal{O}}$ is first used to obfuscate each level of every feature. An obfuscated user representation is then formed by aggregating all relevant obfuscated feature values, as depicted by Figure 4.

Please note that for numerical features with continuous values (e.g., Age or Height), if the feature's cardinality exceeds 100, we apply quantile-based discretization to cast the cardinality to 100, and then employ the $\text{LLM}_{\mathcal{O}}$ to obfuscate each value.

A.2 Constraint Implementation Specifics

As inspired by prior works (Qu et al., 2021; Mai et al., 2024), we also implement a relaxation of our semantic constraint. Specifically, we initiate by ranking all texts in an arbitrary order and proceed to sequentially obfuscate each text using an obfuscation LLM. For each text, if it has no adjacent texts, we directly assign the generated obfuscation. Otherwise, we iterate through all its adjacent texts. For each adjacent text that has been assigned an obfuscation, we compute the BertScore similarity between the newly generated obfuscation and the obfuscation of the adjacent text. To ensure that the semantic constraint is maintained, we check whether the two obfuscations retain at least $1/\epsilon$ of the original similarity between their unobfuscated counterparts. If the generated obfuscation satisfies this constraint for all its adjacent texts that have been obfuscated, we accept it and proceed to the next text. However, if the constraint is violated for any adjacent text, we waitlist the obfuscation and generate a new candidate. This process is repeated iteratively, up to a predefined number of attempts (set to 10 by default). If no valid obfuscation is found within the allowed attempts, we select the obfuscation with the highest mean semantic similarity across all its adjacent texts relative to other failed candidates. For simplicity, we compute a bidirectional BertScore to make it symmetric.

For the Post-Sampling Constraint implementation, we initiate by employing an obfuscation LLM to generate the obfuscation for each text within a task domain. Next, we construct a graph where each text is represented as a node, and two nodes

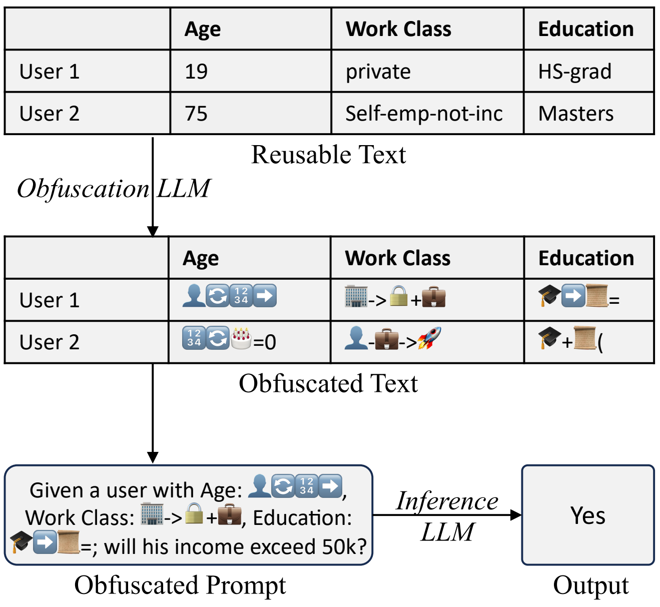


Figure 4: Illustration of EmojiPrompt for preserving user privacy on tabular data.

are connected if the corresponding texts are considered adjacent. We then proceed by repeatedly finding the maximum clique in the graph. For each clique found, we retrieve the obfuscations of all texts within the clique and perform a post-sampling on such obfuscations with a probability distribution computed based on ϵ to obtain the final obfuscation for each node (or text) within the clique, and then remove all nodes within that clique from the graph. This process continues until the size of the maximum clique found is one.

A.3 Data and Metric Specifics

A.3.1 Dataset Description

The Amazon [7] datasets are collected from the Amazon.com platform with user ratings and reviews on products they have purchased on 29 categories of products. In this paper, we concentrate on evaluating performance on the Beauty and Toy categories.

The Movie Reviews [8] dataset is an aggregation of 50,000 movie reviews from the IMDB database. These reviews, composed in natural language, are evenly distributed across two sentiment classes, with 25,000 reviews categorized as positive, and the remaining 25,000 categorized as negative.

The Email Spam [9] dataset consists 5,695 email messages covering a variety of topics, where each message includes the body of the email along with any associated subject lines or headers. Among these messages, 1,368 are labeled as spam.

The Census Income [10] dataset originates from the 1994 U.S. Census database. It encompasses 14 demographic attributes per record, providing a comprehensive overview of individual census respondents. The primary target variable in this dataset is a binary indicator, signifying whether an individual's annual income exceeds $50,000.

The Heart Disease dataset [11] is originally from the CDC, which conducts annual telephone surveys to collect data on the health status of U.S. residents. It contains 18 health-related attributes per sample, with the primary target variable being whether an individual has heart disease.

The Comprehensive Reading (Comp. Reading) dataset [12] comprises 15,000 multiple-choice questions, balanced and annotated by question type across four distinct domains: news, user stories, fiction, and blogs. The questions are crafted to assess the reader's comprehension of the accompanying text passages. Each question has four choices.

The Highlight Generation dataset [13] contains

[7] https://jmcauley.ucsd.edu/data/amazon/
[8] https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
[9] https://www.kaggle.com/datasets/jackksoncsie/spam-email-dataset
[10] https://archive.ics.uci.edu/dataset/2/adult
[11] https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease
[12] https://www.kaggle.com/datasets/thedevastator/introducing-quail-a-comprehensive-reading-compre?resource=download
[13] https://huggingface.co/datasets/abisee/cnn_dailymail

	Optimization	AB (hit@10)	AT (hit@10)	MR (acc)	ES (b-acc)	CI (b-acc)	HD (b-acc)	CR (acc)	HG (cos)
GPT-4	N/A	0.292	0.415	0.957	0.949	0.635	0.648	0.671	0.755
Llama + GPT-4	Manual	0.257	0.371	0.856	0.859	0.692	0.705	0.607	0.687
Llama + GPT-4	APE	0.261	0.376	0.867	0.851	0.699	0.701	0.611	0.691
Llama + GPT-4	OPRO	0.268	0.383	0.871	0.862	0.707	0.712	0.618	0.697

Table 7: Model Performance. For $A + B$, A is the obfuscation LLM and B is the inference LLM. GPT-4 refers to inferencing with unobfuscated prompts.

unique news articles written by journalists at CNN and the Daily Mail, where each article is accompanied by a highlight written by the article author. The highlight has a mean token count of 56, while the article has a mean token count of 781.

For non-English datasets, the Multilingual Spam Detection data [14] consists 5,157 unique passages in three languages, with 13% labeled as spam. The Amazon Review Sentiment data [15] consists customer reviews in various languages. The Article Summarization data [16] is from the official daily news, where each article is accompanied by a headline summarizing its content. For the Heart Attack dataset, we employ an LLM to translate all feature values from the Heart Disease dataset into Spanish.

A.3.2 Metrics Employed

For Amazon Beauty and Toy datasets, we use Hit@k as the evaluation metric; it computes the percentage of ranking lists that include at least one positive item in the top-K highest ranked items (we set K = 10, thus denoting it as hit@10). For the Movie Review and Comp. Reading datasets, we use accuracy as the evaluation metric, as the sentiment label and the question types are both balanced (*i.e.*, with 50% of reviews being positive, and the rest being negative). For the Email Spam, Census Income, and Heart Disease datasets, we employ Balanced Accuracy as the evaluation metric, due to the imbalanced nature of the target feature within each dataset. To clarify, Balanced Accuracy is computed as the average of the proportion of correctly predicted instances in each class, thus ensuring a fair assessment of model performance across both majority and minority classes. For the Highlight Generation dataset, we utilize Cosine Similarity as metric to quantitatively assess the semantic congruence between the target highlights (author generated) and those generated by the LLM$_\mathcal{I}$.

[14] https://www.kaggle.com/datasets/rajnathpatel/multilingual-spam-data/data

[15] https://github.com/tyqiangz/multilingual-sentiment-datasets/tree/main/data

[16] https://www.kaggle.com/datasets/noxmoon/chinese-official-daily-news-since-2016

A.3.3 Data Processing for Modeling

To conduct experiments on the Amazon Beauty and Toy datasets, we extract the complete purchase history for each user and retain the 15 most recently purchased items to assess the LLM's performance in sequential recommendation. Inspired by prior works (Geng et al., 2022; Hua et al., 2023; Ji et al., 2023a), we designate the most recently purchased item as the ground truth or positive sample, with the remaining 14 items as the user's interaction history. Given the finite context window size of the LLM, we adhere to the evaluation methodology proposed by (Geng et al., 2022; Liu et al., 2023) by randomly selecting 99 items from the entire set of beauty/toy products to serve as negative samples. These 100 sampled items collectively form a list of potential candidates for the LLM$_\mathcal{I}$ to rank on. Subsequently, we submit both the user's interaction history, in obfuscated form, and the sampled candidate list to the LLM$_\mathcal{I}$, then prompt it to generate a list of top-K recommended items for the user, based on the user's interaction history (we set K = 10).

For the Movie Reviews dataset, we furnish the LLM$_\mathcal{I}$ with a user's movie review in obfuscated form and instruct it to produce a binary sentiment classification (positive or negative) for the review, devoid of any accompanying explanations. A similar practice is applied to the Email Spam dataset, where we prompt the LLM$_\mathcal{I}$ with an obfuscated email and request the model to determine whether it is spam, also without additional explanations.

For the Census Income and Heart Disease datasets, we provide the LLM$_\mathcal{I}$ with obfuscated versions of each individual's attribute-value pairs. We then prompt the model to make a binary decision by responding with either "yes" or "no", without further elaboration. Specifically, the LLM$_\mathcal{I}$ is asked to determine whether an individual's annual income exceeds \$50,000 or whether the individual has heart disease, respectively. Given that the Census Income dataset was collected in 1994, we explicitly instruct the LLM$_\mathcal{I}$ to determine whether the individual's income would have exceeded \$50,000 in that year, in order to account for inflation.

For the Comp. Reading dataset, we provide the LLM$_\mathcal{I}$ with the privatized passage, along with the question statement as well as the four choices in natural language form. Subsequently, we prompt the LLM$_\mathcal{I}$ to select the most appropriate choice that answers the question based on its comprehension of the obfuscated passage. Lastly, for the Highlight Generation dataset, we provide the LLM$_\mathcal{I}$ with the privatized article and then prompt it to generate a highlight for the article, while ensuring that the token count of the LLM-generated highlight is less than or equal to that of the target highlight.

A.3.4 Budget Information

For this work, we allocate a total of $1750 as we perform a vast amount of evaluations, covering both performance and simulated inference attacks.

A.4 Atomic Obfuscation Against Leakage

For the **Trusted** scenario, our atomic-level obfuscation helps to mitigate the risk of privacy leakage from jailbreaking attacks. Even if attackers manage to obtain one or more queries from the LLM's platform through triggering prompts (as discussed in Section 1), these queries remain largely uninterpretable because the user privacy is obfuscated. To accurately recover user privacy, attackers must acquire the complete set of text-obfuscation pairs for all entities in the task domain. For instance, recovering a user's purchase history would require recovering the representations of all product titles, which could number in the millions. This requirement substantially increases the difficulty of the attack compared to simply accessing a few queries.

For the **Untrusted** scenario, our atomic-level obfuscation preserves user privacy from both the obfuscation LLM and the inference LLM. For the obfuscation LLM, we prompt it to obfuscate each individual entity instead of the entire piece of private text. For instance, on product recommendation task, we prompt it to obfuscate each individual product title instead of the entire user purchase/interaction history. The rationale is that, while a user's purchase history is sensitive, the individual products are not, as they are publicly available on platforms for customers to browse. Similarly, for review (or text in natural language in general) obfuscations, the obfuscation LLM only sees the highly segmented clauses instead of the entire piece of text; such clauses can be merged in numerous ways, rendering it ambiguous on the real content of the text. Thus, even if the server host and/or third-party probers obtain such information, it would still be challenging for them to restore the user privacy. For the inference LLM, we only prompt it to perform task inferencing with user privacy in obfuscated form, so that even if the server host and/or third-party obtain such information, it would be difficult to interpret the user privacy, as they are represented in non-natural language.

A.5 Baseline Specifics

For **TokEmbPriv**, we perturb the token embedding by incorporating stochastic noise prior to uploading to the server. This perturbation is implemented by introducing random noise Z drawn from a d-dimensional distribution characterized by $p(N) \propto \exp(-\eta\|N\|)$. We set the privacy parameter η = 100 to balance utility and privacy, according to the evaluations presented in the paper. We also employ the text-to-text privatization (this post-processing procedure does not affect privacy guarantees).

For **InferDPT**, it comprises two modules: (1) the Perturbation Module, which generates a perturbed text via ϵ-LDP by replacing each token in the text with another from a predefined vocabulary, and (2) the Extraction Module, a locally hosted LLM (less capable than the inference LLM) that reconstructs the noisy output from the cloud-based inference LLM to better align with the original prompt.

We adhere to the methodology described in the paper by employing RANTEXT as the differential privacy mechanism, which, according to the paper, offers superior perturbation performance compared to existing state-of-the-art mechanisms. At its core, InferDPT randomly selects replacement tokens that are sufficiently close to the original token in the embedding space, with probabilities exponentially proportional to proximity. We replace Ins_w with our task-specific instruction. We set $\epsilon = 10$ to balance between utility and privacy, as demonstrated by the synonym evaluation section of the paper. As for the Extraction Module, we utilize Gemma (2B) (Team et al., 2024) to refine open-text outputs from the inference LLM (*i.e.*, outputs that are not confined to a predefined set of tokens). For Tabular datasets, we alter the sampling ranges for numerical values to enhance performances.

For **Split-N-Denoise**, it also adopts d_x-privacy to perform LDP based token-level perturbations, while introducing a novel trained local de-noising LLM to denoise outputs generated by the inference LLM with perturbed inputs. This de-noising LLM takes as input the raw user input embedding,

	AB (hit@10)	AT (hit@10)	MR (acc)	ES (b-acc)	CI (b-acc)	HD (b-acc)	CR (acc)	HG (cos)
Gemini-1.5	0.263	0.379	0.955	0.951	0.641	0.653	0.662	0.759
Gemini-1.5 + Gemini-1.5	0.258	0.363	0.887	0.891	0.719	0.687	0.645	0.724
GPT-3.5 + Gemini-1.5	0.246	0.349	0.863	0.872	0.692	0.681	0.618	0.703
Llama + Gemini-1.5	0.243	0.338	0.857	0.859	0.704	0.695	0.611	0.689

Table 8: Model Performance. For $A + B$, A is the obfuscation LLM and B is the inference LLM. Gemini-1.5 refers to inferencing with unobfuscated prompts. We employ OPRO for obfuscation prompt optimization.

	AB (hit@10)	AT (hit@10)	MR (acc)	ES (b-acc)	CI (b-acc)	HD (b-acc)	CR (acc)	HG (cos)
Claude-3.5	0.271	0.386	0.960	0.949	0.702	0.711	0.664	0.753
Claude-3.5 + Claude-3.5	0.262	0.381	0.891	0.884	0.719	0.725	0.639	0.715
GPT-3.5 + Claude-3.5	0.241	0.369	0.874	0.869	0.715	0.731	0.612	0.699
Llama + Claude-3.5	0.237	0.362	0.862	0.853	0.722	0.727	0.609	0.684

Table 9: Model Performance. For $A + B$, A is the obfuscation LLM and B is the inference LLM. Claude-3.5 refers to inferencing with unobfuscated prompts. We employ OPRO for obfuscation prompt optimization.

the noise matrix, and the noisy embedding from the cloud-based inference LLM. To train the de-noising LLM in a privacy-preserving manner, a public dataset similar to the modeling dataset is needed. In this work, we utilize the validation set reserved for prompt optimization to train the local de-noising LLM. Since cloud-based inference LLMs, such as GPT-4 Turbo, require text as input, we adopt the text-to-text privatization method from TokEmbPriv, mapping perturbed token embeddings to their nearest tokens in the embedding space. We also map inference LLM outputs to their embeddings to facilitate de-noising LLM training.

Lastly, we refer to the following GitHub repositories (while modifying code if needed) for executing the baseline models: InferDPT is hosted at https://github.com/mengtong0110/InferDPT, while TokEmbPriv and Split-N-Denoise are available at https://github.com/NusIoraPrivacy/eaas-privacy/tree/master.

A.6 Prompt Optimization Specifics

We reserve 1,000 samples per dataset as validation set for prompt search, with the remaining data used for performance evaluation. Firstly, we manually tune the obfuscation prompt for both "GPT-4 + GPT-4" and "Gemini + GPT-4". We then employ APE and OPRO for automatic prompt optimization on the obfuscation prompt for both model variants. For both optimization algorithms, we use GPT-3.5 Turbo as the prompt generator to propose candidate obfuscation prompts based on a fixed, manually crafted meta-prompt. Also, we do not provide input-output pairs in the meta-prompt, as there is no universal ground-truth in how an entity should be obfuscated into its non-linguistic form.

For APE, during the Monte Carlo search, 7 candidate prompts are generated per iteration. To enhance efficiency, we implement early-stopping during the evaluation of each candidate prompt and concluded the search after 6 iterations. Specifically, for every 50 samples, we compare the average performance of the current candidate prompt with that of the best-performing prompt. If the current candidate prompt underperforms the best prompt for two consecutive comparisons, we terminate its evaluation and move on to the next candidate. If the current prompt exceeds the performance of the best-performing prompt for the entire validation set, it will be updated as the new best-performing prompt. Additionally, we introduce a slight modification to the standard algorithm: instead of conducting a greedy approach that generates candidate prompts for the next iteration solely based on the best-performing prompt from the current iteration, we generate candidate prompts for the subsequent round from the top two performing prompts.

For OPRO, we adopt the original workflow presented in the paper, where for each newly generated obfuscation prompt, we evaluate its performance score on the validation set (with early-stopping), and augment the prompt-score pair to the meta prompt. We repeat this process for up to 40 iterations, then employ the prompt with best performance. Nevertheless, we observe that initiating the search with a single obfuscation prompt may lead to repeated generation of prompts that are semantically similar to the first prompt in subsequent rounds, thus resulting in only minimal differences in performance. To resolve this, we introduce a modification by requesting the prompt generator to produce three distinct prompts and record their respective performances on the validation set. These three prompt-score pairs are then used to initiate the search process, promoting the generation of more diverse prompts in subsequent rounds.

A.7 Additional Performance Results

In this section, we utilize two additional cloud-based LLMs for task inferencing to evaluate whether private data obfuscated via our paradigm can be interpreted by LLMs other than GPT-4 Turbo. Specifically, we employ Gemini 1.5 Pro (denoted as Gemini-1.5) and Claude 3.5 Sonnet (Kurokawa et al., 2024) (denoted as Claude-3.5). Aligning with Section 4, we use the same inference LLM as the obfuscation LLM for the **Trusted** scenario, while employing GPT-3.5 Turbo (denoted as GPT-3.5) and Llama 3.1 (8B) (denoted as Llama) as obfuscation LLMs for the **Untrusted** scenario. We employ OPRO for obfuscation prompt optimization, as it has achieve best overall performance for both scenarios according to Section 4.2. As demonstrated by Table 8 and Table 9, for both inference LLMs, obfuscating user private data from natural to non-natural language retains sufficient informativeness for task inference by the same LLM, as evidenced by the performance of "Gemini-1.5 + Gemini-1.5" as well as "Claude-3.5 + Claude-3.5". Such approach is also extensible when using different LLMs for obfuscation and inference, as demonstrated by the performance of "GPT-3.5 + Gemini-1.5", "Llama + Gemini-1.5", "GPT-3.5 + Claude-3.5", and "Llama + Claude-3.5".

A.8 Inference Attack Details

A.8.1 LLM-based Attack

In this section, we adopt from prior works (Tong et al., 2023; Mai et al., 2024) by proposing a simulated attack that leverages the $\text{LLM}_{\mathcal{I}}$ to recover obfuscated texts. We assume that the adversary supplies the $\text{LLM}_{\mathcal{I}}$ with details regarding the obfuscation methodology. Specifically, for EmojiPrompt, the $\text{LLM}_{\mathcal{I}}$ is informed that the obfuscation involves transforming natural language into a non-natural form using LLMs. For all baselines, the $\text{LLM}_{\mathcal{I}}$ is made aware that token-level replacements are performed, with candidate tokens chosen based on their proximity within the embedding space. The $\text{LLM}_{\mathcal{I}}$ is subsequently tasked with reversing this transformation to recover the original text from its obfuscated representation. Furthermore, we assume that the adversary provides the $\text{LLM}_{\mathcal{I}}$ with task-specific context, such as indicating that the obfuscated text corresponds to a movie review.

While prior works employ the token-level recovery rate to assess obfuscation robustness, we believe this approach may be an over-simplification. For instance, even if a recoverer fails to accurately recover the exact tokens in an obfuscated product review, it may still produce synonymous tokens or generate a recovered text that retains high semantic similarity to the original content, thus leaking the underlying meaning of the review.

To provide a more comprehensive evaluation against this attack, we employ both semantic and token-level metrics to assess the degree of semantic similarity and lexical overlapping between the LLM-recovered text and the original text. For semantic similarity, we compute the Cosine Similarity score between the original and recovered texts, employing the "text-embedding-3-small" model from OpenAI for embedding vector generations, with the embedding dimension set to 200 to mitigate the curse of dimensionality following a prior work (Lin et al., 2024a). A higher similarity score would suggest lower obfuscation performance, indicating that the $\text{LLM}_{\mathcal{I}}$ can decode an obfuscated text to be more semantically similar to the original. Additionally, for token-level metrics, we compute both ROUGE (1, 2, and L) and METEOR scores between the original and recovered texts, taking into account identical tokens, synonyms, and paraphrases in the overlap. Similar to Cosine Similarity, a higher lexical overlapping would indicate that the obfuscation mechanism is less effective.

We now proceed to specify how the evaluation is conducted. For reusable text, we employ the Amazon Beauty and Census Income datasets, while for non-reusable text, we utilize the Movie Review and Comp. Reading datasets. On the Amazon Beauty dataset, we present the title of each beauty product in its obfuscated form to the $\text{LLM}_{\mathcal{I}}$, prompting the model to infer the original title of the beauty product based on its obfuscated representation. We then compute the Cosine Similarity as well as ROUGE and METEOR scores between the original product title and the inferred product title. Similarly, for the Census Income dataset, we instruct the $\text{LLM}_{\mathcal{I}}$ to recover each obfuscated feature value, providing the feature name as context, and then compute the Cosine Similarity as well as ROUGE and METEOR scores between the original and recovered feature values. As for movie reviews and articles (from Comp. Reading), we direct the $\text{LLM}_{\mathcal{I}}$ to recover each obfuscated review (or article) back to its natural language form, and then compute the Cosine

		Amazon Beauty	Movie Review	Census Income	Comp. Reading
CosSim	Gemini + GPT-4	**0.531**	**0.607**	**0.467**	**0.641**
	SnD + GPT-4	0.609	0.641	0.518	0.673
	InferDPT + GPT-4	0.617	0.635	0.525	0.682
	TEP + GPT-4	0.622	0.657	0.534	0.694
Rouge-1	Gemini + GPT-4	0.211	0.183	0.205	0.228
	SnD + GPT-4	0.207	0.187	0.199	0.221
	InferDPT + GPT-4	0.205	0.193	0.187	0.219
	TEP + GPT-4	0.214	0.179	0.201	0.223
Rouge-2	Gemini + GPT-4	0.061	0.035	0.039	0.047
	SnD + GPT-4	0.058	0.037	0.043	0.051
	InferDPT + GPT-4	0.054	0.031	0.045	0.039
	TEP + GPT-4	0.063	0.042	0.036	0.045
Rouge-L	Gemini + GPT-4	0.207	0.171	0.198	0.215
	SnD + GPT-4	0.201	0.165	0.187	0.209
	InferDPT + GPT-4	0.198	0.179	0.191	0.212
	TEP + GPT-4	0.203	0.163	0.195	0.218
METEOR	Gemini + GPT-4	**0.183**	**0.152**	**0.178**	**0.191**
	SnD + GPT-4	0.219	0.193	0.199	0.238
	InferDPT + GPT-4	0.228	0.189	0.202	0.241
	TEP + GPT-4	0.231	0.196	0.207	0.247

Table 10: Recovering Robustness on LLM-based Attacks across 5 Metrics. For $A + B$, A is the obfuscation method, and B is the inference LLM. SnD, InferDPT, and TEP are the obfuscation methods used as baselines. Any output more than 1% lower than the best performance among the baseline models (lower is better) is highlighted in bold.

Similarity as well as ROUGE and METEOR scores between the original and the recovered review. For each dataset, we compute the mean scores among all entities on each metric. All scores for the EmojiPrompt and the baselines are shown in Table 10.

In addition to benchmarking our EmojiPrompt against the 3 baseline models, we introduce a hypothetical baseline, referred to as "Random Entities", to enable a more comprehensive evaluation of obfuscation robustness. To establish this baseline, we randomly sample a subset of N entities (we set $N = 5$) from the same dataset for each entity. We then compute the mean Cosine Similarity, ROUGE (1, 2, and L), and METEOR scores between the entity and each of the random entities. For example, in the Movie Review dataset, we randomly sample five reviews for each original review, calculate the Cosine Similarity, ROUGE, and METEOR scores between the original review and each of the five selected reviews. We then compute the mean score of the five scores across all three metrics. This procedure is repeated for all reviews in the dataset to obtain the overall mean scores for the three metrics. The same methodology is applied to all entities in the Amazon Beauty, Census Income, and Comp. Reading datasets to compute their respective overall mean scores. All scores for the "Random Entities" baseline are shown in Table 11.

Such scores serve as a strong baseline for obfuscation performance, because if the semantic similarity (and degree of lexical overlapping) of a recovered review falls below this baseline, it indicates that the recovering LLM generates a less relevant output compared to a set of randomly selected texts, demonstrating very effective obfuscation.

As Table 10 showcases, when compared to the baselines, EmojiPrompt demonstrates similar robustness in lexical overlap, as evidenced by the ROUGE scores, while outperforming them in both synonym overlap and overall semantic similarity, as indicated by the METEOR and Cosine Similarity scores. While the scores are higher than those for "Random Entities" (as shown in Table 11), this is acknowledged, as the goal of the obfuscation is not to render the text entirely random but to obscure it while preserving essential information.

A.8.2 Human-based Attack

In addition to utilizing an advanced LLM for inference attacks, we propose two human-based attacks on EmojiPrompt. All participants are independent of this study and were explicitly informed that their data would be used exclusively for experimental purposes within the scope of this research.

Item Identification: on the Amazon Beauty dataset, we randomly sample 300 item-obfuscation pairs, for each item in the pair, we pass a 500-item list (randomly sampled, with the item included) and the item obfuscation to all recoverers, asking them to identify the item from the list based on its obfuscation. We report the percentage of correctly identified items for each recoverer.

Review Recovery: on the Movie Review dataset, we randomly sample 300 review-obfuscation pairs,

		Amazon Beauty	Movie Review	Census Income	Comp. Reading
CosSim	Gemini + GPT-4	0.531	0.607	0.467	0.641
	Random Entities	0.307	0.411	0.295	0.392
Rouge-1	Gemini + GPT-4	0.211	0.183	0.205	0.228
	Random Entities	0.173	0.168	0.161	0.155
Rouge-2	Gemini + GPT-4	0.061	0.035	0.039	0.047
	Random Entities	0.049	0.019	0.021	0.017
Rouge-L	Gemini + GPT-4	0.207	0.171	0.198	0.215
	Random Entities	0.167	0.152	0.147	0.149
METEOR	Gemini + GPT-4	0.183	0.152	0.178	0.191
	Random Entities	0.159	0.143	0.139	0.141

Table 11: Recovering Robustness on LLM-based Attack benchmarked against Random Entities. Random Entities involves replacing private entities in each dataset with randomly selected entities from the same dataset.

for each sampled review, we provide all recoverers with its obfuscation, asking them to recover the original review. We then report the mean Cosine Similarity, ROUGE (1, 2, and L), and METEOR scores between each human recoverer and their corresponding original reviews.

For the **Item Identification** test, five human recoverers completed the task, correctly identifying 31, 28, 24, 17, and 21 items, respectively. This results in a mean identification rate of 8.07%. It is important to note that while the item identification test employs a list of 500 items (including the target item) for evaluators to identify the correct item based on its obfuscation, this is already a simplified evaluation. In the domain of beauty products, there are tens of thousands of items available on major online platforms, assuming web scraping is performed, which is substantially larger than the 499 negative samples from our test. Despite this simplification, our obfuscation mechanism still demonstrates solid performance.

For the **Review Recovery** test, three human recoverers completed the task. Again, we employ the "text-embedding-3-small" model for embedding generation, with embedding vector dimension set to 200. The resulting mean similarity scores for the recoverers were: 0.556, 0.493, and 0.587, respectively. For ROUGE and METEOR, we report the F1 score, as it represents the harmonic mean between precision and recall. The scores for all recoverers are presented in Table 12. Overall, the performance of the human recoverers generally aligns with that of the LLM$_{\mathcal{I}}$, further underscoring the effectiveness of the obfuscation. We choose not to have the human recoverers identify the correct review from a list of candidate reviews, as we did in the Item Identification test, due to the nature of the data. In e-commerce, items are generally accessible information from popular online platforms and may be scrapped, whereas individual customer reviews are not publicly accessible unless released by the company or the individuals themselves.

	ROUGE-1	ROUGE-2	ROUGE-L	METEOR
Recoverer 1	0.189	0.029	0.169	0.167
Recoverer 2	0.178	0.025	0.153	0.161
Recoverer 3	0.191	0.031	0.178	0.181

Table 12: Human-based Attack for Gemini + GPT-4 on Review Recovery

A.8.3 Distribution-based Attack

Furthermore, we propose a hypothetical attack aimed at tabular datasets, where the adversary targets a dataset used for a specific inference task, such as heart disease classification. Knowing the task, the adversary collects related public datasets via web scraping, which contain similar features to the target dataset. This allows them to analyze the value distributions for overlapping features.

The adversary then examines the distribution of each value in the public datasets for features that overlap with the obfuscated features in the target dataset. Despite not being able to directly recover the obfuscated feature values, the adversary uses the distribution information to make educated guesses about the obfuscated data. For example, if a public dataset shows that 70% of entries for the feature "Gender" are "Male", and the obfuscated dataset has a value comprising 67% of the entries for the same feature, then the adversary may infer that this obfuscated value corresponds to "Male".

By matching these distributions, the adversary effectively maps the obfuscated values to their actual values. This technique bypasses the obfuscation by leveraging statistical patterns rather than attempting to recover the obfuscated text directly.

To protect tabular datasets against distributional inference attacks, we propose a novel generative obfuscation mechanism with post-sampling. This approach involves generating multiple obfuscated representations for each feature value instead of mapping each value to a single obfuscation.

Consider an example with a feature that has two possible values, A and B, where A accounts for 40% of the dataset instances and B accounts for 60%. Instead of generating a single obfuscated representation for A, we prompt LLM$_{\mathcal{O}}$ to produce four distinct and substantially different obfuscations: A_1, A_2, A_3, and A_4. Likewise, for B, we prompt LLM$_{\mathcal{O}}$ to generate three distinct and substantially different obfuscations: B_1, B_2, and B_3. During the inference stage, one of the obfuscated representations is randomly sampled for each instance of the feature value. Consequently, for feature value A, the obfuscated representations A_1, A_2, A_3, and A_4 will each appear in approximately 10% of instances, while for feature value B, the obfuscated representations B_1, B_2, and B_3 will each appear in approximately 20% of instances.

This method offers two key benefits. (1) by decoupling the obfuscated values from their original distribution, this mechanism prevents adversaries from accurately deducing the original feature values through distributional analysis. (2) the variability in obfuscated representations allows multiple combinations of obfuscated values to collectively approximate the original distribution proportions of 40% for A and 60% for B. For instance, the combination of A_1, A_2, and B_1 could collectively account for 40% of the dataset instances. This variability makes it challenging for adversaries to ascertain which obfuscated representations correspond to specific feature values, thus obscuring discernible patterns and enhancing data security.

To assess whether our multi-obfuscation method retains task performance, we employ both Census Income and Heart Disease datasets to perform an ablation study against single obfuscated representation, with Gemini as LLM$_{\mathcal{O}}$ and GPT-4 Turbo as LLM$_{\mathcal{I}}$. For each categorical feature, we randomly sample two to four unique obfuscated representations for each value of the feature. To ensure that the obfuscated representations are sufficiently different, we repeat the generation process until each obfuscated representation has a Cosine Similarity of 0.5 or less with all other representations. The balanced accuracies of multi-obfuscation versus single-obfuscation are: 0.657 vs. 0.674 (for Census Income) and 0.691 vs. 0.689 (for Heart Disease). These figures demonstrate that our method retains performance for tabular modeling.