Cape: Context-Aware Prompt Perturbation Mechanism with Differential Privacy

Haoqi Wu [1] **Wei Dai** [1] **Li Wang** [1] **Qiang Yan** [1]

Abstract

Large Language Models (LLMs) have gained significant popularity due to their remarkable capabilities in text understanding and generation. However, despite their widespread deployment in inference services such as ChatGPT, concerns about the potential leakage of sensitive user data have arisen. Existing solutions primarily rely on privacy-enhancing technologies to mitigate such risks, facing the trade-off among efficiency, privacy, and utility. To narrow this gap, we propose `Cape`, a context-aware prompt perturbation mechanism based on differential privacy, to enable efficient inference with an improved privacy-utility trade-off. Concretely, we introduce a hybrid utility function that better captures the token similarity. Additionally, we propose a bucketized sampling mechanism to handle large sampling space, which might lead to long-tail phenomenons. Extensive experiments across multiple datasets, along with ablation studies, demonstrate that `Cape` achieves a better privacy-utility trade-off compared to prior state-of-the-art works.

1. Introduction

Recent advancements in large language models (Vaswani et al., 2017) have revolutionized fields such as natural language processing (Devlin et al., 2019; Radford et al., 2019), driving their widespread adoption in machine learning (ML) inference services. However, despite their growing popularity, data privacy remains a critical concern. In ML services like ChatGPT (Brown et al., 2020), the server provides an API that processes client queries and returns generated responses. This paradigm (Figure 1) requires clients to transmit their queries—commonly referred to as prompts in the context of language models—to the server in plaintext, posing significant risks to user privacy, as the prompts can contain sensitive user information or confidential business guidelines (Yu et al., 2024). For example, a user might request ChatGPT to summarize or analyze a confidential business report (Forbes, 2023), potentially resulting in severe privacy breaches. This issue is further exacerbated by the growing stringency of modern data privacy regulations.

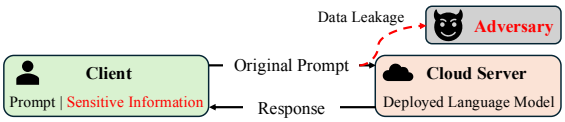


Figure 1: Illustration of existing LLM inference services.

Existing solutions (Ma et al., 2023; Dong et al., 2023; Wu et al., 2024; Hou et al., 2023; Li et al., 2024) utilize cryptographic techniques like secure multi-party computation (MPC) (Yao, 1986) and homomorphic encryption (HE) (Gentry, 2009) to offer provable security. However, the huge computation and communication overhead hinders their application in real world scenarios [1]. Another line of works utilize the client-server hybrid execution paradigm, where client performs some lightweight computations and sends shallow layer embeddings to the server. However, prior work (Song & Raghunathan, 2020) has shown that such information can be leveraged to reconstruct the original text effectively. Subsequent works (Zhou et al., 2023; Du et al., 2023) bridge this gap by adding noise to intermediate embeddings to break the linkage between the noisy embeddings and original tokens. However, a major limitation is their reliance on a white-box inference setup, requiring shallow layers to be deployed on the client side, which demands extensive model modifications, making it seldom practical.

A promising approach is to utilize differential privacy (DP) (Dwork, 2006), widely adopted as the *de facto* standard for protecting user privacy, to verbatim perturb tokens in the prompts. It is highly efficient and requires no modifications to the back-end model. SANTEXT (Yue et al., 2021) and CUSTEXT (Chen et al., 2022) focus on text classification tasks. InferDPT (Tong et al., 2023) takes the first

[1]TikTok. Correspondence to: Haoqi Wu <haoqi.1997@tiktok.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

[1]Take the SOTA 3-party computing work Ditto (Wu et al., 2024) as an example, the average runtime for generating one token with a sequence length of 128 on Bert-base model requires about 30s.

step towards the open-ended text generation tasks that manages to extract more accurate generation from the perturbed response from the server with the aid of a local LLM. To enhance utility, these works replace the original token with a random token from 'truncated' adjacency lists, to reduce the large sampling space in NLP (e.g., BERT models have a vocabulary of size 30,522). However, this practice achieves higher utility at the sacrifice of privacy, making it vulnerable to k-nearest neighbor attacks (Song & Raghunathan, 2020). Besides, these works typically measure semantic similarity via token-level embedding distances. Such approach overlooks contextual information, often leading to reduced contextual semantic coherence.

A Motivating Example: As shown in Figure 2, 'enjoyable' and 'unenjoyable' appear close in embedding space despite conveying opposite meanings. Incorporating contextual information as a constraint can mitigate such ambiguities.

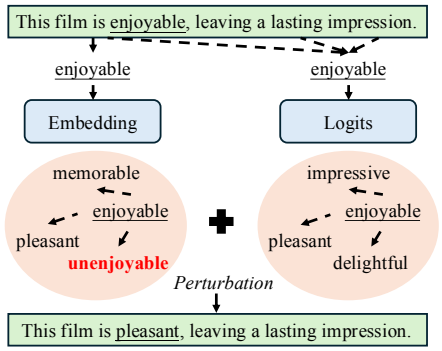


Figure 2: Intuition: Drawback of embedding distance alone.

Therefore, *Can we perform efficient, context-aware prompt perturbation with better privacy-utility trade-off?*

As an answer to the above question, we develop `Cape`, a *context-aware* and *bucketized* DP mechanism tailored to enhance semantic similarity measurement and optimize the sampling process in large sampling spaces for NLP tasks.

- **Context-Aware Prompt Perturbation**. We introduce a hybrid utility function that integrates both *contextual information* and *token embedding distance* (e.g., Euclidean distance) that achieves improved utility. Such utility function is essential to the subsequent DP-based perturbation (formulated as *private selection* problems), to ensure rigorous privacy guarantees.
- **Bucketized Sampling Mechanism**. We propose a customized exponential mechanism to tackle the challenge of large sampling spaces in NLP domain. Specifically, we utilize equal-width bucketing to restrict the sampling probability of low-utility items, achieving an improved privacy-utility trade-off.
- **Empirical Evaluations.** We conduct extensive experiments on both text classification and text generation tasks across three datasets. The experiments confirm that `Cape` achieves a better privacy-utility trade-off and provides stronger defense against existing privacy attacks. Additionally, ablation studies on the hyperparameters in utility function and bucketized sampling further validate their effectiveness in our approach.

2. Related Work

Existing works typically employ privacy enhancing technologies to enable private inference, where exist a trade-off among efficiency, utility, and privacy. Cryptographic-based solutions provide provable security and nearly lossless inference yet at the sacrifice of efficiency. The state-of-the-art works Ditto (Wu et al., 2024) and CipherGPT (Hou et al., 2023) offers comparable accuracy to plaintext computation. However, they require at least 30s to infer a token on Bert-base model, which is not ready for practical scenarios.

Table 1: Comparison of different methods. ✓ indicates the framework supports a feature, ●, ◐ and ○ refer to high-, medium- and low-performance, respectively.

Method	Black-box Inference	Privacy	Efficiency	Utility
CipherGPT (Hou et al., 2023)	✓	●	○	●
Ditto (Wu et al., 2024)	✓	●	○	●
TextObfuscator (Zhou et al., 2023)	✗	◐	◐	◐
DP-Forward (Du et al., 2023)	✗	◐	◐	◐
SANTEXT (Yue et al., 2021)	✓	✗	◐	●	◐
CUSTEXT (Chen et al., 2022)	✓	✗	◐	●	◐
InferDPT (Tong et al., 2023)	✓	✓	◐	●	◐
`Cape`	✓	✓	◐	●	◐

For DP-based solutions, although they yield better efficiency but faces the trade-off between privacy and utility. Some works add DP noise based on token embedding distance over the discrete vocabulary set. SANTEXT (Yue et al., 2021) utilizes metric-LDP (Alvim et al., 2018) and provides privacy protection over the entire vocabulary. Subsequently, CUSTEXT (Chen et al., 2022) and InferDPT (Tong et al., 2023) prunes the sampling space to a 'truncated' adjacency list (either static or dynamic), to achieve better utility at the sacrifice of privacy. Furthermore, SANTEXT and CUSTEXT mainly focus on training data privacy and text classification tasks. Our work introduces a context-aware utility function and optimized sampling mechanism tailored for large sampling space that strikes better trade-off. Other works like TextObfuscator (Zhou et al., 2023) and DP-Forward (Du et al., 2023) adopt a client-server hybrid paradigm to offload some part of the model to the client. Consequently, they can add noise to the continuous intermediate embeddings, which shall incur smaller errors compared to sampling over a discrete domain. However, they require white-box access to the model and fine-tuning to align the embedding space, which could be hard to put into practice. In comparison, our work can be applied to black-box scenario, requiring minimal modifications to existing paradigm. Recently, there

are also some works that offer less privacy protection but potentially better utility for minimizing disclosure risks (Dou et al., 2024; Staab et al., 2024). These works use LLMs to detect and anonymize sensitive attributes in an innovative way, which we perceive as complementary to our approach for improving utility.

3. Preliminaries

3.1. Transformer-based Inference

Transformer models generally consist of three parts: 1) the `Embedding` module that maps discrete tokens to continuous word embeddings; 2) a stack of Transformer `Block`; 3) the last `Head` module that maps the hidden embeddings to task-specific *logit*. The logits $y_t = \mathcal{M}(x_t|Ctx)$ are used to deduce the sampling probabilities for next token x_t, where Ctx denotes the preceding text (and following text).

3.2. Differential Privacy

Differential privacy (DP) defines the *worst-case* privacy guarantee. The maximum output probability difference between any two neighbor inputs is bounded by e^ϵ, where ϵ denotes the privacy budget. In this paper, we focus on *local* DP (Kasiviswanathan et al., 2008), which operates on individual data points x and allows clients to locally perturb their data before sending it to the server using the DP mechanism $\mathcal{R}$. Therefore, an adversary cannot effectively infer the original input x based on the outputs.

Definition 3.1 (ϵ-Local Differential Privacy (Kasiviswanathan et al., 2008))**.** Given a privacy budget $\epsilon > 0$, a randomized algorithm $\mathcal{R} : \mathcal{X} \to \mathcal{Y}$ satisfies ϵ-LDP if, for each possible output $y \in \mathcal{Y}$, and all adjacent $x_1, x_2 \in \mathcal{X}$, the following holds:

$$\frac{\mathbb{P}[\mathcal{R}(x_1) = y]}{\mathbb{P}[\mathcal{R}(x_2) = y]} \leq e^\epsilon \tag{1}$$

In the NLP domain, we tokenize the prompt into tokens, and the sampling spaces $\mathcal{X}$ and $\mathcal{Y}$ are defined as the token vocabulary $\mathcal{V}$. Conceptually, under ϵ-LDP, any adjacent tokens x_1 and x_2 can be randomized to the same output token y with indistinguishable probability.

3.3. Private Selection

The task of private selection is to design a randomized algorithm $\mathcal{R}$ that returns some y_i that approximately maximizes the utility $u(x, y_i)$ while satisfying differential privacy.

Definition 3.2 (ϵ-Exponential Mechanism (EM) (McSherry & Talwar, 2007))**.** Given a privacy budget $\epsilon > 0$, a randomized Exponential mechanism $\mathcal{R}_{EM} : \mathcal{X} \to \mathcal{Y}$ satisfies ϵ-LDP if, for each possible output $y_i \in \mathcal{Y}$, and input $x \in \mathcal{X}$:

$$\mathbb{P}[y_i|x] \propto \frac{\epsilon}{2\triangle} \cdot u(x, y_i) \tag{2}$$

where $u(x, y_i)$ denotes the utility function that measures the utility score for selecting y_i, and $\triangle$ denotes the sensitivity, which is computed as $\triangle_u = \max_{x,x',y} |u(x, y) - u(x', y)|$.

3.4. Threat Model

In our scenario, the client $\mathcal{C}$ sends the prompt x to cloud server $\mathcal{S}$ and receives generation results in return: $r \leftarrow \mathcal{M}(x)$. Here, $x = \{t_1, t_2, ..., t_n\}$, where $t_i \in \mathcal{V}$ denotes the tokens in the vocabulary. The prompt x may contain sensitive tokens such as emails and genders, and directly uploading such information to $\mathcal{S}$ compromises the privacy of clients. Table 2 lists the main used notations in this work.

Table 2: Notations used in this paper.

Symbol	Description
$\mathcal{S}$	the server that provides inference service
$\mathcal{C}$	the client that inputs prompt
$\mathcal{M}$	the language model
$\mathcal{V}$	the vocabulary
t	word token $t \in \mathcal{V}$
$x, \hat{x}$	the original prompt $x = \{t_1, t_2, ..., t_n\}$ and perturbed prompt
$\epsilon, \triangle$	privacy budget and sensitivity, respectively
$u(\cdot, \cdot)$	utility function
$\mathcal{R}$	the private selection mechanism
$d_{euc}(t, t')$	Euclidean distance between tokens t and t'
L_r	the logits for candidate token t_r given context L_r
$D(t_i, t_r)$	the distance between i-th token t_i and candidate token t_r
N_b	the number of buckets
λ_L, λ_D	importance factors of contextual logits and token distance

Goal. Our goal is to perturb the original prompt using DP as $\hat{x} = \mathcal{R}(x)$, ensuring that not only the perturbed $\hat{x}$ provides *rigorous privacy guarantee* (i.e., an adversary can not effectively infer original prompt) but also maintains an *acceptable utility* (i.e., task-specific performance is retained).

Adversary. We consider the LLM inference service provider $\mathcal{S}$ as the potential adversary. We do not consider man-in-the-middle attacks and assume the communication channel is secure. $\mathcal{S}$ is assumed to be *semi-honest*, meaning the adversary follows the execution flow for inference but may attempt to extract sensitive information by collecting and analyzing the messages received from the client. $\mathcal{S}$ has access to the perturbed prompt $\hat{x}$, and the DP mechanism $\mathcal{R}$ is publicly known. Moreover, $\mathcal{S}$ possesses unlimited computational power, allowing it to launch attacks, such as the mask token inference attack (Yue et al., 2021).

4. Design

4.1. Context-aware Prompt Perturbation

`Cape` acts as a safeguard layer, allowing the client to locally perturb the prompts and hide sensitive data from the server.

As illustrated in Figure 3 (in green background), the overall procedure is as follows: Given an input prompt containing a sequence of sensitive tokens $x = \{t_1, t_2, ..., t_n\}$, and a

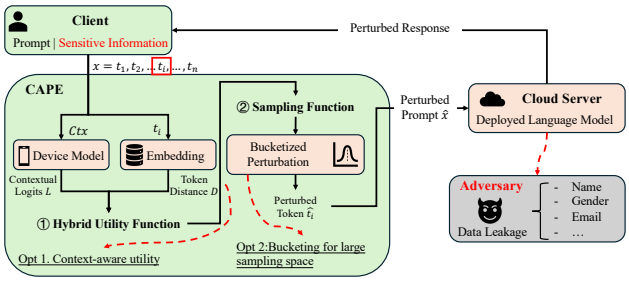


Figure 3: High-level workflow of Cape.

differential private mechanism $\mathcal{R}$ (i.e., Cape), we verbatim replace the original token t_i to a randomized output token $\hat{t}_i = t_r \in \mathcal{V}$. All these randomized tokens $\hat{t}_i$ constitute the perturbed prompt $\hat{x} = \{\hat{t}_i = \mathcal{R}(t_i)\}$, where $i \in [1, n]$. Then, the perturbed prompt $\hat{x}$ is sent to $\mathcal{S}$ for subsequent text generation. Cape consists of two major components:

- **Hybrid Utility Function:** By hybrid, we consider both token distance D and contextual logits L when computing the utility scores for candidate sampling tokens. Prior works (Yue et al., 2021; Chen et al., 2022) merely focus on the token distance alone. We resort to a client-side small device model to capture the contextual information and use Euclidean distance as the default measurement for token distance.
- **Bucketized Sampling Function:** We adopt Exponential mechanism, the *de facto* standard in private selection problems, to sample replacement tokens. Notably, the large sampling space in NLP (e.g., the vocabulary in Bert-base models is 30,522) could lead to long-tail phenomenon. To tackle this problem, we propose a bucketized variant to refine the sampling probability distribution, especially when the budget ϵ is small.

4.1.1. Utility Function

In private selection, the utility function is essential to the sampling performance, which should satisfy two features: 1) **Monotonicity:** Candidates with higher utility scores are more likely to be sampled, which guarantees the utility of the mechanism; 2) **Boundedness:** The output space of the utility function should be bounded to keep the sensitivity $\triangle$ controllable, ensuring rigorous privacy protection.

We hereby design a hybrid utility function for the i-th token t_i in a prompt as:

$$u(t_i, t_r) = L_r^{\lambda_L} \cdot D(t_i, t_r)^{\lambda_D} \tag{3}$$

where t_i, t_r denote the original and candidate tokens, $L_r = \mathcal{M}_c(t_r|Ctx)$ denotes the logits for t_r given context Ctx and a client-side model $\mathcal{M}_c$. Ctx can be both preceding and following information (e.g., $\{t_1, ..., t_{i-1}, t_{i+1}, ..., t_n\}$ in Bert) or preceding information alone (e.g., $\{t_1, t_2, ..., t_{i-1}\}$ in GPT). $D(t_i, t_r)$ denotes the token embedding distance between t_i and t_r. Additionally, we introduce importance factors λ_L and λ_D for logit and distance, respectively. Theoretically, increasing λ_L enhances *contextual coherence*, while emphasizing λ_D improves *semantic similarity*.

To satisfy the monotonicity, since a larger logit indicates higher contextual relevance (i.e., positive correlation) while a larger distance indicates lower semantic similarity (i.e., negative correlation), we use the form $u \propto L \cdot \mathbf{exp}(-D)$ to obtain the utility. By default, we use Euclidean distance over the embedding space to measure the token distance D. The contribution of token distance can be formulated as:

$$\vec{D}_{raw} = \{d_{euc}(t_i, t_r), 1 \le r \le |\mathcal{V}|\} \tag{4}$$

$$\vec{D}_{norm} = \frac{\vec{D} - D_{min}}{D_{max} - D_{min}} \tag{5}$$

$$\vec{D} = \mathbf{exp}(-\vec{D}_{norm}) \tag{6}$$

Regarding the boundedness, since $\vec{D}_{norm} \in [0, 1]$, we have $\vec{D} \le 1$. However, the logit L, which distributes over real value space $\mathbb{R}$, is dependent on the model and input. To limit the contribution of logit to the utility score, we can clip the logit to a given bound B as follows:

$$\vec{L} = \mathbf{clip}(\vec{L}_{raw}, B) \tag{7}$$

To determine a suitable bound B, we analyze the logit distribution using a few calibration samples. The bound is then selected based on the maximum value. We provide a histogram of both logits and distance in Appendix B.7.

4.1.2. Sampling Algorithm

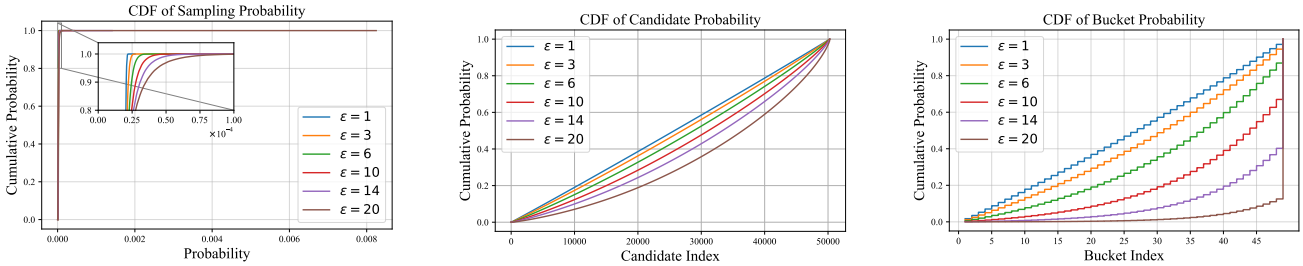
With the utility function defined, we proceed to the sampling algorithm, with Exponential mechanism (EM) as the basis.

Long-Tail Phenomenon. We first note that there exists long-tail phenomenon when the sampling space is extremely large (i.e., a large vocabulary $\mathcal{V}$ in NLP). In standard EM, the sampling probability for t_r can be simply formulated as:

$$\mathbb{P}[\mathcal{R}(t_i) = t_r] = \frac{\exp(\frac{\epsilon}{2\triangle} u(t_i, t_r))}{\sum_{j=1}^{|\mathcal{V}|} \exp(\frac{\epsilon}{2\triangle} u(t_i, t_j))} \tag{8}$$

When $\mathcal{V}$ is relatively large, many low-utility candidate tokens have individually negligible probabilities, but collectively their sum is significant. According to the cumulative sampling probability distribution in Figure 4a and 4b, the majority of probabilities are significantly lower than 0.0001, yet they account for a substantial portion of the CDF.

Theoretically, let p_s and p_l denote the smallest and largest probabilities, respectively, with their maximum difference bounded by e^ϵ (i.e., $p_l \le p_s \cdot e^\epsilon$). N denotes the vocabulary size. The probability sum of the top-k tokens ($\mathcal{K}$) relative to



(a) Sampling CDF. (b) Candidate Probability. (c) Bucket Probability.

Figure 4: Cumulative distribution function of sampling probabilities for 'book' in 'This is a good book.' upon GPT2.

the remaining ones can be expressed as:

$$P_1 = \sum_{i \in \mathcal{K}} p_i \leq k \cdot p_l, P_2 = \sum_{i \notin \mathcal{K}} p_i \geq (N-k) \cdot p_s \quad (9)$$

$$\frac{P_1}{P_2} \leq \frac{k \cdot p_l}{(N-k) \cdot p_s} \leq \frac{k \cdot e^{\epsilon}}{(N-k)} \approx \frac{k \cdot e^{\epsilon}}{N} \quad (10)$$

If we set $\epsilon = 6$, $k = 10$ and $N = 50000$, we have $\frac{P_1}{P_2} \leq \frac{403 \cdot 10}{50000} \approx 0.08$. That is, a great portion of low-utility candidates takes over 90% percent of probability mass. In practice, the empirical cumulative probability for $k = 10$ is less than 1%, i.e., $P_2 \geq 99\%$. This observation contradicts the intuitive expectation that candidate tokens closely resembling the original token would have higher probabilities.

Algorithm 1 Equal width Bucketing: **Bucket**

Input: Utility score vector $\vec{u}$, the number of buckets N_b
Output: A set of buckets B with different tokens.

```
Initialize B ← ∅
ū_s = Sort(ū)
u2b = Bucketize(ū_s, N_b)
for i ← 1 to N_b do
    b_i ← {t_j | u2b_j == i}
    ū_{b,i} ← ū{b_i}
    if len(v) > 0 then
        B ← B|(mean(ū_{b,i}), b_i)
return B
```

Bucketing Strategy. A straightforward solution is to use a larger ϵ, which, however, results in a low level of formal privacy. To address this, we propose to bucketize the sampling space to refine the sampling probability distribution. Specifically, we adopt equal-width bucketing, which partitions the sampling space into uniformly sized N_b buckets based on the numerical utility ranges. The key idea is to bound the probability sum of all candidates in the same bucket (i.e., within a certain utility interval), proportional to $\exp(\frac{\epsilon}{2\triangle}\mathbf{mean}(b_i))$, where b_i denotes the token set that maps to the i-th bucket, $\vec{u}_{b,i}$ denotes the score vectors for these tokens and $\mathbf{mean}(b_i) = \sum_{k=1}^{|b_i|} b_{i,k}/|b_i|$. As shown in Algorithm 1, we sort the utility scores and assign each candidate to a unique bucket of width $(u_{max} - u_{min})/N_b$. Notably, if a bucket contains no candidates, it is skipped-a scenario that may occur when N_b is large. For each bucket, we use the average utility score of its candidates as its representative score. Then, as shown in Algorithm 2, we use EM to sample a bucket, followed by a uniform sampling from the chosen bucket to obtain the replacement token. The bucketized probability is given by:

$$\mathbb{P}[\mathcal{R}(t) = t_r] \propto \frac{\exp(\frac{\epsilon}{2\triangle}\mathbf{mean}(b_i))}{|b_i|}, \text{ if } t_r \in b_i \quad (11)$$

where $\triangle = \max_{t_r} \max_{b \sim b'} |b - b'|$, b, b' denote utilities of different buckets. As shown in Figure 4c, `Cape` effectively mitigates the long-tail phenomenon, where low-utility candidates now constitute a smaller cumulative probability. The proof to Theorem 4.1 is provided in Appendix C.

Theorem 4.1. *The bucketized Exponential Mechanism satisfies $(\epsilon + \epsilon')$-differential privacy, where $\epsilon' = \ln(\max_{i,j} \frac{|b_i|}{|b_j|})$.*

4.2. Define Non-sensitive Tokens

To further enhance the utility, we utilize a pre-defined set of context-independent non-sensitive tokens. This set primarily consists of 179 NLTK stopwords (e.g., 'the', 'a') and 32 punctuations (e.g., ',', '.'), which, while lacking significant context-sensitive information, are essential for text coherence and readability. We retain these non-sensitive tokens while perturbing those sensitive ones in the prompt.

5. Experiments

Datasets & Metrics. We consider two types of text tasks: 1) text classification; and 2) open-ended text generation. For the former, we follow prior works (Yue et al., 2021; Chen et al., 2022) to use two GLUE datasets with privacy implications. 1) **SST-2:** This contains sentiment annotations for movie reviews, which is used to perform sentiment classification (positive or negative); 2) **QNLI:** This is a dataset containing sentence pairs for binary classification (entailment/not entailment). We use *accuracy* as the metric.

Algorithm 2 `Cape` Mechanism: $\mathcal{R}$

Input: Prompt $x = \{t_1, t_2, ..., t_n\}$, device model $\mathcal{M}$, embedding table $\vec{e}$, importance factors λ_L, λ_D, bucket number N_b, and budget $\epsilon > 0$
Output: Perturbed prompt $\hat{x} \leftarrow \mathcal{R}(x)$

Initialize $\hat{x} \leftarrow \emptyset$
for $i \leftarrow 1$ **to** n **do**
  $\vec{u}(t_i) = \textbf{Utility}(\mathcal{M}, Ctx, \vec{e}, \lambda_L, \lambda_D)$
  $\vec{u}_b(t_i), \vec{b}(t_i) \leftarrow \textbf{Bucket}(\vec{u}(t_i), N_b)$
  /* Sample a bucket using EM */
  $b_r \leftarrow \textbf{EM}(\vec{u}_b(t_i), \vec{b}(t_i), \epsilon)$
  /* Randomly choose a token from b_r */
  $t_r \sim \textbf{Uniform}(b_r)$
  $\hat{x} \leftarrow \hat{x}|t_r$
return $\hat{x}$

For the latter, we follow (Tong et al., 2023) to use **Wikitext-103-v1**, a large-scale dataset derived from Wikipedia articles for language modeling tasks. We use *coherence*, and *alignment* as the evaluation metrics (detailed in Section 5.1.2).

For privacy evaluation, we consider the following two attacks. We calculate the attack success rate asr on sensitive tokens and the privacy score as $1 - asr$.

KNN Attack (Song & Raghunathan, 2020). The adversary computes the embedding distance between the perturbed token and all other tokens in the vocabulary, and then selects the Top-k tokens with the smallest distances, where we set $k = 10$. The **Top-10 accuracy** is defined as the proportion of cases where the original token appears within Top-10.

Masked Token Inference (MTI) Attack (Yue et al., 2021). The adversary employs a BERT model, which captures both preceding and following contextual information, to predict the masked token. We replicate this attack by sequentially replacing the perturbed token $\hat{t}$ with the special token [MASK]. The **Rouge-L F1 score** is used to evaluate the similarity between predicted and original prompt.

Baselines. We compare `Cape` with SANTEXT (Yue et al., 2021) and CUSTEXT (Chen et al., 2022) on their default setting (based on GloVe embedding). We also adopt InferDPT (Tong et al., 2023) as the baseline (based on text-embedding-ada-002 (OpenAI, 2023)), which is the first work that supports open-ended text generation. For `Cape`, we utilize either the BERT (Devlin et al., 2019) model or the GPT-2 (Radford et al., 2019) model to capture contextual information, along with their respective embeddings to compute token distances. By default, we set $\lambda_L = 0.5$, $\lambda_D = 1.0$ and $N_b = 50$. We run inference on the original data as non-private baseline. The detailed vocabulary configuration is provided in Appendix A. To ensure a fair comparison, we faithfully implement all these mechanisms in Python and re-run all the experiments.

Experimental setup. All the experiments are carried out on one Debian 11 machine equipped with one Intel Xeon Platinum 8260 CPU (6 cores and 2.40GHz), 16GB of RAM and 4 Nvidia Tesla-V100-SXM2-32GB GPUs.

5.1. Privacy-Utility Trade-off

To begin, we focus on the **empirical** privacy-utility trade-off of different perturbation mechanisms in both text classification and open-ended text generation tasks. We will study the effects of **formal** privacy ϵ in the following Section 5.2.

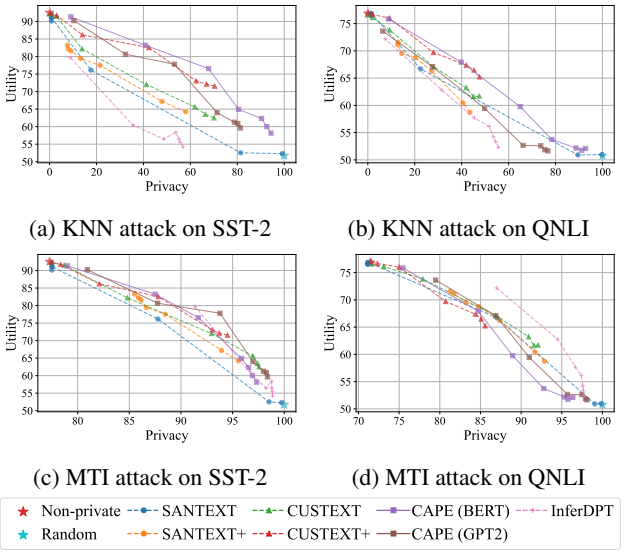
In terms of efficiency, after a setup phase that computes the token embedding distance, the perturbation of `Cape` only consumes about $0.1 \sim 0.15$ second for each input in average considering a small batch size of 2. The detailed efficiency evaluation is provided in Appendix B.5.

5.1.1. Text Classification

For text classification tasks, we leverage a pre-trained Qwen2-1.5B-Instruct model (Li et al., 2023) for zero-shot learning on the validation set of SST-2 and QNLI datasets. The instructions (i.e., system prompts) used to obtain classification results are provided in Appendix D. Notably, this inference-only paradigm presents a greater challenge in effectively preserving semantics, as the absence of re-training limits its ability to capture and refine noisy semantics.

Figure 5 illustrates the privacy-utility trade-off under KNN and MTI attacks with $\epsilon \in [1, 20]$, respectively. The closer the line is to the **upper right** corner, the better the trade-off. The detailed utility and privacy numbers are deferred to Appendix B.1 and B.4. The results across different settings confirm that `Cape` achieves better trade-off than baselines. `Cape` achieves comparable task utility to CUSTEXT and superior to that of SANTEXT and InferDPT owing to the hybrid utility function and bucketing optimization. Specifically, CUSTEXT achieves higher utility even when ϵ is small (a stringent formal privacy) owing to its usage of fixed and small adjacency lists. However, such practice leads to worse defense against attacks. Besides, `Cape` (BERT) yields better utility than `Cape` (GPT2) since Bert models captures both preceding and following contextual information, thus ensuring better coherence and semantic similarity.

There is one special case in Figure 5d that `Cape` (BERT) performs slightly worse than `Cape` (GPT2) and the baselines when ϵ is small under MTI attacks. This is mainly due to the fact that BERT used in MTI attack fail to recognize the sub-word tokens in GPT-based `Cape` (GPT2) and InferDPT, and word-level tokens in GloVe-based SANTEXT and CUSTEXT. This also explains the high privacy scores (over 70%) even when $\epsilon = 20$. Such vocabulary mismatch actually hinders the attack performance. Therefore, we mainly



(a) KNN attack on SST-2 (b) KNN attack on QNLI

(c) MTI attack on SST-2 (d) MTI attack on QNLI

Figure 5: Privacy-utility trade-offs in terms of privacy attacks vs. accuracy rates by varying $\epsilon \in [1, 20]$. Privacy is measured by privacy scores under empirical attacks.

evaluate KNN attacks in the following experiments.

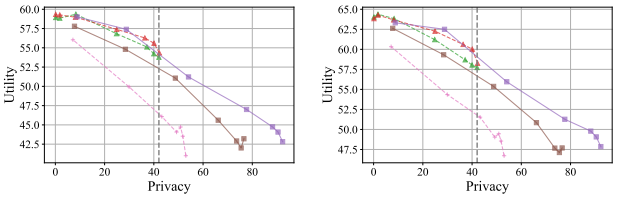
5.1.2. Open-ended Text Generation Tasks

Compared to text classification tasks, text generation is more sensitive to the perturbation. When some key information in the text is disturbed, the generated text may have poor coherence and alignment to the original prompt. We thereby follow InferDPT (Tong et al., 2023) to use a lightweight local model (Qwen2-1.5B-Instruct), denoted as *extraction* model $\mathcal{M}_e$, to refine the perturbed response generated by the server. When receiving the noisy generation $\hat{y}$ from $\mathcal{S}, \mathcal{C}$ uses the *extraction* model to de-noise the response as $y' \leftarrow \mathcal{M}_e(x, \hat{y})$. Additionally, we denote the expected response (i.e., non-private) as y. In the following experiments, we vary the perturbation mechanisms, while utilizing the same extraction model (default temperature of 0.5). The instructions used to generate noisy response and extract response are provided in Appendix D. We use *coherence* and *alignment* to measure the text generation utility. Both metrics are computed using cosine similarity that goes as:

$$CS(s_1, s_2) = \frac{Emb(s_1) \cdot Emb(s_2)}{|Emb(s_1)| \cdot |Emb(s_2)|} \tag{12}$$

where the sentence embedding $Emb(s)$ is computed using the method in SimCSE (Gao et al., 2021). **Coherence** measures the cosine similarity between original prompt x and the extracted response y' as $CS(x, y')$, and **Alignment** measures the similarity between the extracted response y' and expected response y as $CS(y, y')$.

Notably, we fail to evaluate SANTEXT because it requires a vocabulary size of approximately 221,642 over Wikitext,



(a) KNN attack vs. Coherence. (b) KNN attack vs. Alignment.

Figure 6: Privacy-utility trade-offs in terms of KNN attack by varying $\epsilon \in [1, 20]$ on Wikitext-103-v1 dataset. Privacy is measured by privacy scores under empirical attacks.

which causes OOM error when calculating sampling probabilities. As shown in Figure 6, `Cape` significantly outperforms InferDPT and achieves similar utility to CUSTEXT when ϵ is large. Whereas, CUSTEXT fails to provide reasonable empirical privacy. Even with $\epsilon = 1$, it achieves a privacy score of only around 40%. The results exhibit the superior performance of `Cape`. The detailed utility numbers are deferred to Appendix B.2.

5.2. Influence of Privacy Budget

In this section, we evaluate each method in terms of formal privacy, with $\epsilon \in [1, 20]$. Notably, we examine the effect of varying ϵ on each method. We do not attempt to align the privacy levels across these methods, as they provide distinct forms of *actual* formal privacy, which could introduce ambiguities. For instance, the ϵ-metric-LDP-based SANTEXT effectively adheres to ϵ'-DP, where $\epsilon' = \epsilon \cdot d_{max}$, with $d_{max} \sim 14.86$ (as derived from GloVe embeddings, shown in Table 4). In InferDPT, the random adjacency list is generated using Laplace noise with a default budget of approximately 9, resulting in an actual privacy budget of $\epsilon' \sim \epsilon + 9$. Similarly, our method operates under $\epsilon' = \epsilon + \ln(\max_{i,j} \frac{|b_i|}{|b_j|}) \sim \epsilon + 8$. An exception is CUSTEXT, which employs a static adjacency list of default size 20, making it hard to quantify the actual privacy budget.

5.2.1. Utility Evaluation

We compute the **Rouge-L F1 score** between the original and perturbed prompts on SST-2 dataset, which serves as an indicator of sentence-level semantic similarity. We also provide some perturbation examples in Appendix E.

The results are presented in Table 3. In general, the scores of all methods increase with the privacy budget ϵ. Among them, SANTEXT+ and CUSTEXT+ achieve high scores even when ϵ is small. This is because in SANTEXT+, a proportion of tokens (top $w\%$ most frequent tokens, even not stopwords) are treated as non-sensitive. While in CUS-

Table 3: Sentence-level similarity of various methods.

Mechanism		Rouge-L (F1) ↑						
		$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 6$	$\epsilon = 10$	$\epsilon = 14$	$\epsilon = 20$
SANTEXT		0.87	13.43	71.31	99.36	99.45	99.45	99.45
SANTEXT+		52.56	57.77	72.98	76.37	77.46	77.66	77.56
CUSTEXT		14.50	17.87	22.74	47.27	81.44	95.54	99.48
CUSTEXT+		50.66	52.90	55.47	69.41	88.46	97.18	99.71
InferDPT		13.00	14.09	14.92	16.48	23.20	38.68	68.11
Cape	Bert	38.38	39.20	40.86	46.85	60.22	76.49	92.03
	GPT2	37.60	38.19	40.08	44.55	56.46	73.46	90.65

TEXT, each word has a pre-defined token adjacency list, which results in a high probability that a word will either remain unchanged or be perturbed to a synonym[2], yielding higher similarity. This also accounts for their weak privacy defense against attacks. Regarding SANTEXT, InferDPT and Cape, Cape achieves the best performance. Specifically, when we align the actual $\epsilon \sim 14$, corresponding to SANTEXT ($\epsilon \sim 1$), InferDPT and Cape ($\epsilon \sim 6$), the score of Cape is significantly larger than the other two methods.

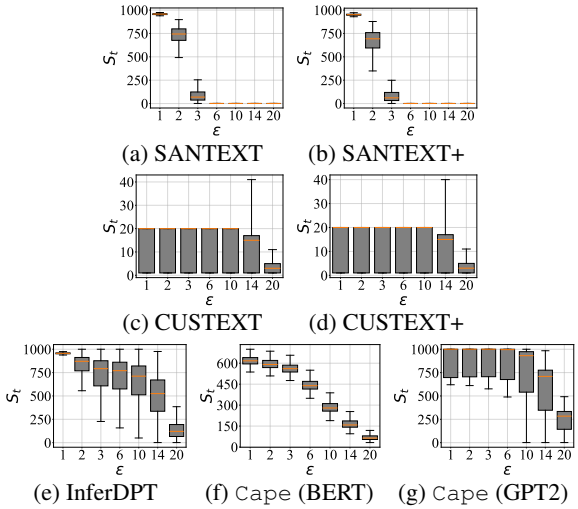
5.2.2. Privacy Evaluation

We evaluate the empirical privacy using **effective mapping set size** S_t. This is calculated by counting the number of distinct tokens mapped from t after perturbation as $S_t = \{t' | t' = \mathcal{R}(t)\}$. Intuitively, the mapping set should encompass the entire vocabulary. A larger S_t implies greater entropy in the mapping distribution, thereby indicating higher empirical privacy. We also provide two other metrics: *retention ratio* N_t and *defense against attacks* in Appendix B.3∼B.4 due to page limitation.

To evaluate S_t, we run perturbation methods 1,000 times to each token in the vocabulary on SST-2 dataset with $\epsilon \in [1, 20]$. The results are illustrated in Figure 7. For the metric-LDP-based SANTEXT, S_t decreases rapidly as ϵ increases, dropping to approximately 100 when $\epsilon = 3$. In contrast, CUSTEXT maintains S_t at around 20 in most cases, even when $\epsilon = 1$. This consistency stems from CUSTEXT's design, which restricts the sampling space for each token to a fixed size (20 by default). However, this approach also implies that under a KNN attack with Top-20 accuracy, CUSTEXT's privacy score would be nearly zero.

For InferDPT and Cape, both methods exhibit a proportional decrease in S_t as ϵ increases due to their use of EM over the entire vocabulary. Owing to our bucketing strategy, Cape exhibits lower entropy than InferDPT. Notably, in Cape, the BERT variant outperforms the GPT-2 variant by producing a smaller and smoother S_t distribution. This can be attributed to BERT's use of a denser embedding space. Moreover, BERT is better at capturing both preceding and following context. This distinction also explains why Cape (BERT) surpasses Cape (GPT-2) in the above experiments.

[2]The proportion of synonyms exceeds 40% even when $\epsilon = 2$, as shown in Table III of the paper (Tong et al., 2023).



(a) SANTEXT (b) SANTEXT+ (c) CUSTEXT (d) CUSTEXT+ (e) InferDPT (f) Cape (BERT) (g) Cape (GPT2)

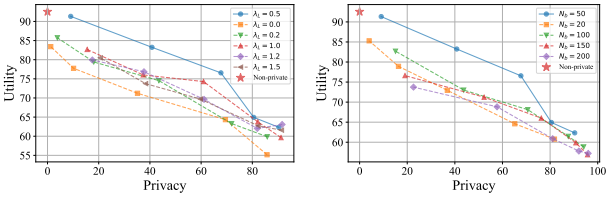
Figure 7: Mapping set size S_t with $\epsilon \in [1, 20]$ on SST-2.

5.3. Ablation Studies

We additionally evaluate Cape under various parameter configurations on SST-2 dataset. From now on, we adopt the Bert-based Cape for its superiority. We here vary utility importance factor and bucket number to perform an extensive comparison. Additionally, we evaluate the feasibility of using **model distillation** on client-side device model in Appendix B.6. We choose KNN attack as the privacy metric and evaluate the classification accuracy on SST-2 dataset.

5.3.1. Varying Importance Factor

To evaluate the effect of incorporating contextual logits, we run experiments on varying logit importance factor $\lambda_L \in [0.2, 0.5, 1.0, 1.2, 1.5]$, with fixed distance importance factor λ_D as 1.0 in the hybrid utility function.



(a) $\lambda_L \in [0, 1.5]$ (b) $N_b \in [20, 200]$

Figure 8: Ablation results under various parameter configurations with $\epsilon \in [1, 20]$.

As shown in Figure 8a, a utility function that relies solely on token embedding distance (i.e., $\lambda_L = 0.0$) demonstrates the poorest performance, highlighting the importance of incorporating contextual logits for designing effective utility

functions. However, assigning excessive weight to logits (e.g., $\lambda_L = 1.5$) results in diminished performance. This occurs because a token that better fits the context may not always be semantically similar to the original token.

5.3.2. Varying Bucket Number

We use bucketing to handle the dilemma of large sampling space. To test the effect of bucket number N_b, we conduct experiments on varying $N_b \in \{20, 50, 100, 150, 200\}$.

The experiment results are illustrated in Figure 8b. Notably, $N_b = 50$ achieves the best performance, while the two extremes, $N_b \in \{20, 200\}$, yield the worst results. As N_b increases, resulting in a larger sampling space, the empirical privacy level improves. For instance, when $\epsilon = 1.0$, the privacy score for $N_b = 200$ is approximately 10% higher than that for $N_b = 50$. However, this comes at the cost of utility, which declines rapidly. This is because larger N_b makes it more likely to select a bucket with low utility, causing significant utility degradation. Balancing the privacy-utility trade-off, $N_b = 50$ emerges as an optimal choice.

6. Conclusion

In this paper, we propose a context-aware prompt perturbation mechanism to strike a better balance in privacy-utility trade-off in LLM inference services. By perturbing user prompts using differential privacy, we provide quantifiable privacy protection, with moderate utility degradation. In the future, we plan to explore more fine-grained privacy mechanisms tailored to different groups of tokens, such as application-specific semantic categories or user-specific personalization scenarios.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. Our work aims to address critical privacy concerns surrounding user prompts in widely adopted LLM-based inference services. To safeguard sensitive user information during inference, we leverage differential privacy, which provides quantifiable privacy protection controlled by a privacy budget. However, we also acknowledge the inevitable utility degradation caused by the introduction of random noise to perturb the original prompts. A key contribution of this work lies in the development of a novel hybrid utility function that integrates contextual information with token embedding distances, alongside a bucketized sampling algorithm designed specifically for the NLP domain's large sampling space. These advancements have the potential to make private LLM inference more practical and scalable for real-world applications.

References

Alvim, M. S., Chatzikokolakis, K., Palamidessi, C., and Pazii, A. Invited paper: Local differential privacy on metric spaces: Optimizing the trade-off with utility. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, pp. 262–267. IEEE Computer Society, 2018. doi: 10.1109/CSF.2018.00026. URL https://doi.org/10.1109/CSF.2018.00026.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

Chen, H., Mo, F., Chen, C., Cui, J., and Nie, J. A customised text privatisation mechanism with differential privacy. *CoRR*, abs/2207.01193, 2022. doi: 10.48550/ARXIV.2207.01193. URL https://doi.org/10.48550/arXiv.2207.01193.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Dong, Y., Lu, W., Zheng, Y., Wu, H., Zhao, D., Tan, J., Huang, Z., Hong, C., Wei, T., and Chen, W. PUMA: secure inference of llama-7b in five minutes. *CoRR*, abs/2307.12533, 2023. doi: 10.48550/arXiv.2307.12533. URL https://doi.org/10.48550/arXiv.2307.12533.

Dou, Y., Krsek, I., Naous, T., Kabra, A., Das, S., Ritter, A., and Xu, W. Reducing privacy risks in online self-disclosures with language models. In Ku, L., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 13732–13754. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.741. URL https://doi.org/10.18653/v1/2024.acl-long.741.

Du, M., Yue, X., Chow, S. S. M., Wang, T., Huang, C., and Sun, H. Dp-forward: Fine-tuning and inference on language models with differential privacy in forward pass. In Meng, W., Jensen, C. D., Cremers, C., and Kirda, E. (eds.), *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pp. 2665–2679. ACM, 2023. doi: 10.1145/3576915.3616592. URL https://doi.org/10.1145/3576915.3616592.

Dwork, C. *Differential Privacy*, pp. 1–12. Jan 2006. doi: 10.1007/11787006_1. URL http://dx.doi.org/10.1007/11787006_1.

Forbes. Samsung bans chatgpt among employees after sensitive code leak, 2023. URL https://www.forbes.com/sites/siladityaray/2023/05/02/samsung-bans-chatgpt-and-other-chatbots-for-employees-after-sensitive-code-leak.

Gao, T., Yao, X., and Chen, D. SimCSE: Simple contrastive learning of sentence embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Gentry, C. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, USA, 2009. URL https://searchworks.stanford.edu/view/8493082.

Hou, X., Liu, J., Li, J., Li, Y., Lu, W., Hong, C., and Ren, K. Ciphergpt: Secure two-party GPT inference. *IACR Cryptol. ePrint Arch.*, pp. 1147, 2023. URL https://eprint.iacr.org/2023/1147.

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. D. What can we learn privately? In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pp. 531–540. IEEE Computer Society, 2008. doi: 10.1109/FOCS.2008.27. URL https://doi.org/10.1109/FOCS.2008.27.

Li, Z., Zhang, X., Zhang, Y., Long, D., Xie, P., and Zhang, M. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

Li, Z., Yang, K., Tan, J., Lu, W., Wu, H., Wang, X., Yu, Y., Zhao, D., Zheng, Y., Guo, M., and Leng, J. Nimbus: Secure and efficient two-party inference for transformers. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/264a9b3ce46abdf572dcfe0401141989-Abstract-Conference.html.

Ma, J., Zheng, Y., Feng, J., Zhao, D., Wu, H., Fang, W., Tan, J., Yu, C., Zhang, B., and Wang, L. Secretflow-spu: A performant and user-friendly framework for privacy-preserving machine learning. In Lawall, J. and Williams, D. (eds.), *Proceedings of the 2023 USENIX Annual Technical Conference, USENIX ATC 2023, Boston, MA, USA, July 10-12, 2023*, pp. 17–33. USENIX Association, 2023. URL https://www.usenix.org/conference/atc23/presentation/ma.

McSherry, F. and Talwar, K. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pp. 94–103. IEEE Computer Society, 2007. doi: 10.1109/FOCS.2007.41. URL https://doi.org/10.1109/FOCS.2007.41.

OpenAI. New and improved embedding model., 2023. URL https://openai.com/blog/new-andimproved-embedding-model.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL http://arxiv.org/abs/1910.01108.

Song, C. and Raghunathan, A. Information leakage in embedding models. In Ligatti, J., Ou, X., Katz, J., and Vigna, G. (eds.), *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pp. 377–390. ACM, 2020. doi: 10.1145/3372297.3417270. URL https://doi.org/10.1145/3372297.3417270.

Staab, R., Vero, M., Balunovic, M., and Vechev, M. T. Large language models are advanced anonymizers. *CoRR*, abs/2402.13846, 2024. doi: 10.48550/ARXIV.2402.13846. URL https://doi.org/10.48550/arXiv.2402.13846.

Tong, M., Chen, K., Qi, Y., Zhang, J., Zhang, W., and Yu, N. Privinfer: Privacy-preserving inference for black-box large language model. *CoRR*, abs/2310.12214, 2023. doi: 10.48550/ARXIV.2310.12214. URL https://doi.org/10.48550/arXiv.2310.12214.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Wu, H., Fang, W., Zheng, Y., Ma, J., Tan, J., and Wang, L. Ditto: Quantization-aware secure inference of transformers upon MPC. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=ZzXNCQGzqT.

Yao, A. C. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pp. 162–167. IEEE Computer Society, 1986. doi: 10.1109/SFCS.1986.25. URL https://doi.org/10.1109/SFCS.1986.25.

Yu, D., Kairouz, P., Oh, S., and Xu, Z. Privacy-preserving instructions for aligning large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=mUT1biz09t.

Yue, X., Du, M., Wang, T., Li, Y., Sun, H., and Chow, S. S. M. Differential privacy for text analytics via natural text sanitization. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pp. 3853–3866. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.FINDINGS-ACL.337. URL https://doi.org/10.18653/v1/2021.findings-acl.337.

Zhou, X., Lu, Y., Ma, R., Gui, T., Wang, Y., Ding, Y., Zhang, Y., Zhang, Q., and Huang, X. Textobfuscator: Making pre-trained language model a privacy protector via obfuscating word representations. In Rogers, A., Boyd-Graber, J. L., and Okazaki, N. (eds.), *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 5459–5473. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.337. URL https://doi.org/10.18653/v1/2023.findings-acl.337.

A. Vocabulary Configuration

The vocabulary configuration for the baselines and Cape are provided in Table 4. Notably, SANTEXT leverages a subset of GloVe, specifically tailored to the dataset's vocabulary, excluding tokens that do not appear in the dataset. CUSTEXT similarly trims the GloVe vocabulary to align with a predefined vocabulary subset [3]. InferDPT also prunes the cl100k_base to initial 11,000 English tokens as the vocabulary. Specifically, the embeddings in CUSTEXT are normalized, resulting in smaller Euclidean distances.

Table 4: Vocabulary configurations. $\mathcal{V}_{sample}$ denotes the actual sampling space. K and $\hat{K}$ denote the size of (random) adjacency list in CUSTEXT and InferDPT, respectively.

Methods	GloVe ($d = 300$)			Bert ($d = 768$)			GPT ($d \in \{768, 1536\}$)		
	$\lvert\mathcal{V}\rvert$	d_{max}	$\lvert\mathcal{V}_{sample}\rvert$	$\lvert\mathcal{V}\rvert$	d_{max}	$\lvert\mathcal{V}_{sample}\rvert$	$\lvert\mathcal{V}\rvert$	d_{max}	$\lvert\mathcal{V}_{sample}\rvert$
SANTEXT	2,196,017	14.86	14,283 (SST-2) 87,751 (QNLI) 221,642 (Wikitext-103-v1)	-	-	-	-	-	-
CUSTEXT	76,855	1.66	K	-	-	-	-	-	-
InferDPT	-	-	-	-	-	-	100,256 (11,000)	1.44	$\hat{K}$
Cape	-	-	-	30,522	2.89	30,522	50,257	7.58	50,257

B. Detailed Experiment Results

B.1. Utility for text classification tasks

Table 5 presents the utility of different methods, measured by inference accuracy of text classification on SST-2 and QNLI datasets. Notably, as mentioned in Section 5.2, we directly use the definition of privacy budget ϵ in their papers without alignment to our work in case of ambiguities. We here focus on the influence of ϵ.

Table 5: Utility comparison of various perturbation mechanisms at similar privacy levels on SST-2 and QNLI datasets.

Methods		SST-2							QNLI						
		$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 6$	$\epsilon = 10$	$\epsilon = 14$	$\epsilon = 20$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 6$	$\epsilon = 10$	$\epsilon = 14$	$\epsilon = 20$
Random		51.72							50.78						
SANTEXT		52.29	52.52	76.15	91.06	92.09	91.17	90.14	50.98	50.91	66.68	76.90	76.55	76.55	76.66
SANTEXT+		64.22	67.20	77.52	79.47	81.63	82.17	83.27	58.72	60.44	66.26	68.74	69.52	71.13	71.59
CUSTEXT		62.50	63.53	65.60	72.01	82.21	91.51	92.20	61.71	61.61	63.23	67.07	73.81	76.11	76.57
CUSTEXT+		71.56	72.13	73.05	82.56	86.19	91.74	92.43	65.22	66.48	67.38	69.69	76.00	76.66	77.01
InferDPT		54.19	55.52	56.54	58.38	56.42	60.44	79.70	52.27	53.33	54.19	56.13	57.67	62.80	72.18
Cape	Bert	58.14	60.01	62.33	64.93	76.56	83.24	91.33	52.10	51.77	52.17	53.74	59.76	67.93	75.84
	GPT2	59.63	60.89	61.27	63.99	77.81	80.67	90.25	51.66	51.86	52.59	52.66	59.45	67.09	73.62
Non-private		92.54							76.88						

B.2. Utility for open-ended text generation tasks

The detailed utility numbers for open-ended text generation tasks are illustrated in Table 6. We measure the utility of the noisy generation from two aspects: *coherence* to the original prompt, and *alignment* to the expected generation.

B.3. Privacy: Retention Ratio

Retention ratio N_t is calculated by counting the frequency of retention of token t after the perturbation. This is a straightforward metric for evaluating privacy protection. A higher N_t indicates that more tokens remain unperturbed, reflecting a lower level of privacy protection. As the privacy budget ϵ increases, all methods exhibit higher N_t. Among

[3] Limited vocabulary: https://github.com/nmrksic/counter-fitting/blob/master/linguistic_constraints/vocabulary.txt

Table 6: Coherence & Similarity on Open-ended Text Generation.

Methods	Coherence ↑							Alignment ↑						
	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 6$	$\epsilon = 10$	$\epsilon = 14$	$\epsilon = 20$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 6$	$\epsilon = 10$	$\epsilon = 14$	$\epsilon = 20$
CUSTEXT	53.78	54.23	55.11	56.83	59.36	58.85	58.93	57.73	58.04	58.69	61.21	63.83	64.38	64.03
CUSTEXT+	54.34	55.59	56.28	57.39	58.97	59.27	59.34	58.26	60.02	60.61	62.26	63.67	64.27	63.83
InferDPT	41.00	43.53	44.73	44.07	46.07	49.93	56.04	46.72	48.55	49.46	49.04	51.53	54.35	60.33
Cape Bert	42.82	44.07	44.74	47.00	51.22	55.40	58.02	47.86	49.07	49.79	51.27	55.95	59.50	62.31
Cape GPT	43.20	42.01	42.91	45.61	51.07	54.81	57.80	47.69	47.12	47.67	50.85	55.37	59.32	62.62
Non-private	59.78							65.05						

these methods, Cape consistently achieves the lowest N_t. SANTEXT demonstrates a much faster increase in N_t due to its usage of metric-LDP. Additionally, since SANTEXT+ classifies a portion of the most frequent tokens as non-sensitive, and InferDPT operates with a truncated vocabulary of size 11,000, both methods exhibit high N_t even when $\epsilon = 1$. CUSTEXT also produces higher N_t due to its limited sampling space for each token. Under a fixed ϵ, the N_t of CUSTEXT is nearly double that of Cape.

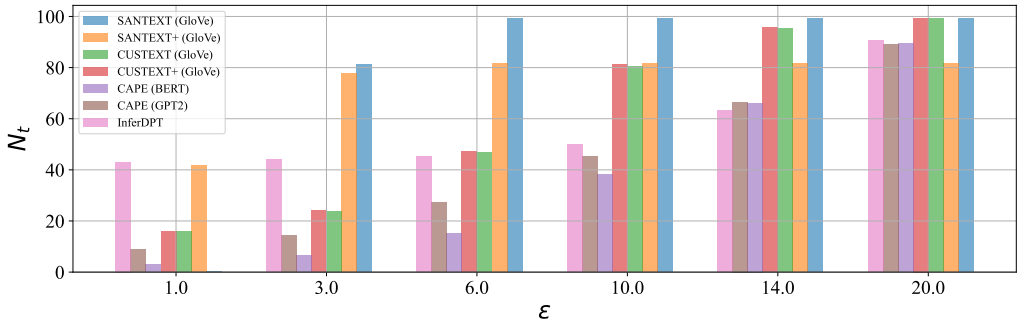
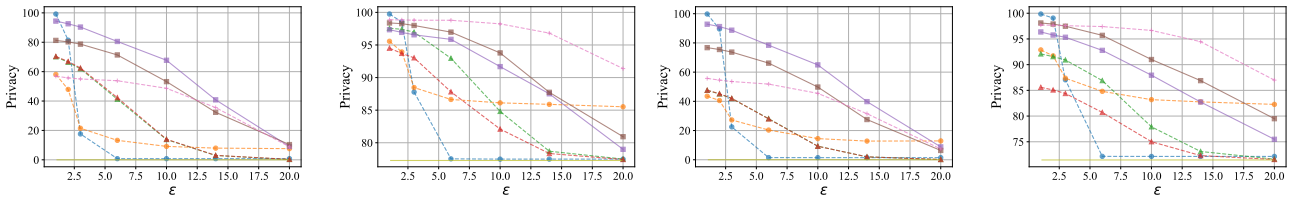


Figure 9: Retention ratio N_t by varying the privacy parameter $\epsilon \in [1, 20]$ on the SST-2 dataset.

B.4. Privacy: Defense scores against privacy attacks

We conduct some existing attacks to empirically demonstrate the privacy protection comparison with varying privacy budgets. As introduced in Section 5, we calculate the attack success rate asr on sensitive tokens and the privacy score as $1 - asr$.



(a) KNN attacks on SST-2. (b) MTI attacks on SST-2. (c) KNN attacks on QNLI. (d) MTI attacks on QNLI.

Figure 10: Privacy evaluation by varying the privacy parameter $\epsilon \in [1, 20]$.

B.5. Efficiency Evaluation

We evaluate the efficiency of various methods by measuring their runtime cost and memory consumption. A batch size of 2 is used, representing two simultaneous client-side queries. This is reasonable given the lack of high concurrency

requirements on the client. Each experiment is repeated five times to calculate the average runtime. The total runtime is divided into two phases: 1) **Setup** phase, which pre-computes token embedding distances, constructs adjacency lists and calculates sampling probabilities; 2) **Perturbation** phase, which performs the perturbation on a input user prompt.

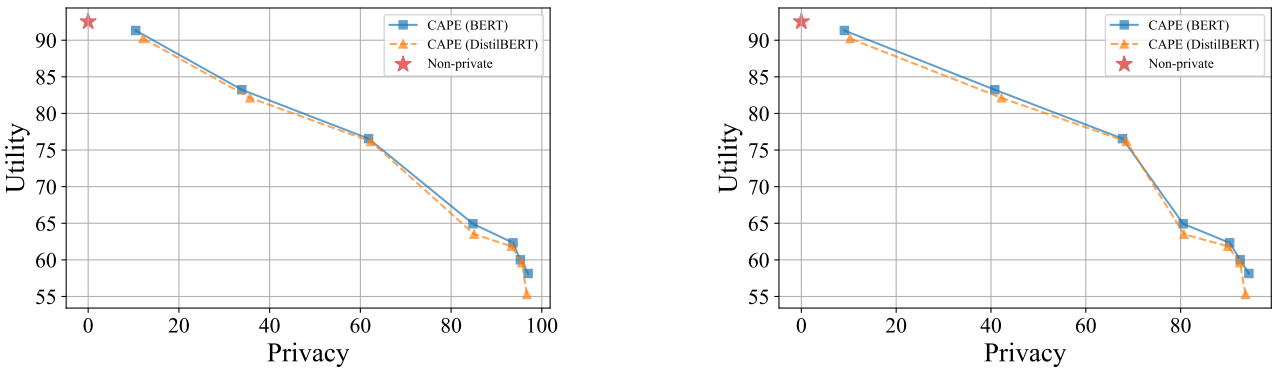
As shown in Table 7, Cape ensures prompt privacy with minimal time overhead. While Cape consumes more resources due to the use of a local model, its average runtime per input prompt remains under 0.2 seconds. SANTEXT and CUSTEXT exhibit the lowest runtime costs, as they precompute adjacency lists and sampling probabilities for each token. However, this comes at the expense of higher setup phase runtimes, particularly for CUSTEXT, where adjacency list construction significantly increases overhead. The GPU memory consumption of Cape is approximately $1 \sim 1.5$ GB. In Appendix B.6, we further explore the potential of model distillation to reduce this memory requirement.

Table 7: Efficiency evaluation by measuring the average runtime cost in each stage.

Methods		**Runtime**		**Memory**
		Setup	Perturbation	Perturbation
SANTEXT		325.14s	0.003s	0.81G
CUSTEXT		2929.75s	0.002s	0.16G
InferDPT		28.66s	0.05s	1.03G
Cape	Bert	37.91s	0.15s	1.37G
	GPT2	98.25s	0.10s	1.12G

B.6. Ablation: Distilled Local Model

In Cape, a client-side device model is used to obtain the contextual information to enhance the utility. The parameter size for the Bert-base and GPT2-base models evaluated above is about 400 MB. We note that for those resource-limited devices, we need to further decrease the resource consumption. Model distillation (Sanh et al., 2019) provides a promising direction to lower the model size. To verify its feasibility in Cape, we evaluated the performance on the widely-used distilbert model [4], of which the size is reduced by about 40%.



(a) KNN attack with Top-1 accuracy.

(b) KNN attack with Top-10 accuracy

Figure 11: Privacy-utility trade-offs in terms of success rates of KNN attacks vs. accuracy rates by varying the privacy parameter $\epsilon \in [1, 20]$ on the SST-2 dataset.

We evaluate the classification accuracy and KNN attack accuracy on the SST-2 dataset using both the bert-base and distilbert models. As illustrated in Figure 11, distilbert generally achieves performance comparable to the original bert model. Specifically, its classification accuracy is at most $\sim 1\%$ lower than that of bert under the same empirical privacy level. Detailed results in Table 8 indicate that the distilled model exhibits slightly lower accuracy and stronger privacy defense on

[4] https://huggingface.co/distilbert/distilbert-base-uncased.

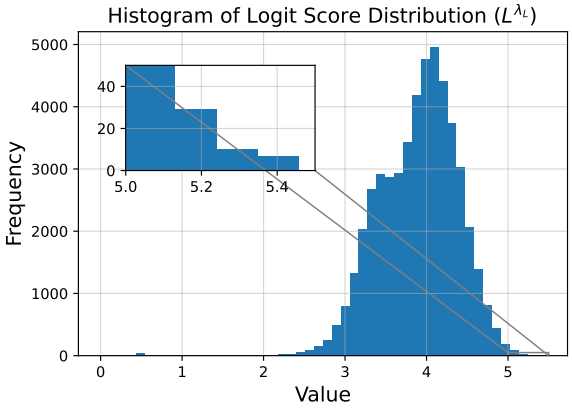
a fixed formal priavcy level (i.e., ϵ). This can be attributed to the reduced expressiveness of distilbert embeddings compared to the original model.

Table 8: Privacy-utility trade-offs in terms of success rates of KNN attacks vs. accuracy rates w/wo model distillation.

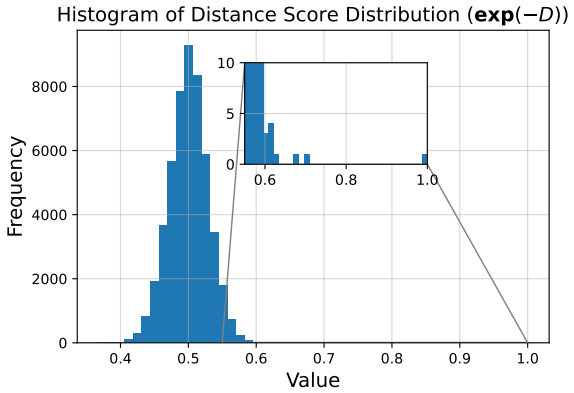
Methods		Accuracy ↑							Privacy (Top-10) ↑						
		$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 6$	$\epsilon = 10$	$\epsilon = 14$	$\epsilon = 20$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 6$	$\epsilon = 10$	$\epsilon = 14$	$\epsilon = 20$
Cape	Bert	58.14	60.01	62.33	64.93	76.56	83.24	91.33	94.40	92.56	90.34	80.54	67.74	40.80	9.07
	DistilBert	55.28	59.63	61.86	63.50	76.17	82.13	90.24	93.69	92.50	89.96	80.72	68.57	42.21	10.27
Non-private		92.54							100.0						

B.7. Histogram of logits and token embedding distance

We here run some statistics on the example prompt: 'This is a good book.' using GPT-2 model. We compute the logits for book with 'This is a good [MASK].' as the context. We compute the Euclidean distance between book and other tokens in the vocabulary $\mathcal{V}$. The histogram of the logits and distance are illustrated in Figure 12.



(a) Logit L^{λ_L}



(b) Distance $\exp(-D)^{\lambda_D}$

Figure 12: Histogram of different distance metrics for 'This is a good book.'. $\lambda_L = 0.5, \lambda_D = 1.0$.

C. Proofs of Theorems

Proof of Theorem 4.1. Consider the original token $t \in \mathcal{V}$, another token $t' \in \mathcal{V} \setminus \{t\}$ and a possible output token $y \in \mathcal{V}$. We use $B(t) = b_i(i \in [N_b])$ to denote the i-th bucket that token t belongs to, N_b is the bucket number. Additionally, we use $u_b(t, \cdot)$ to denote the bucket utility for $B(t)$ and $|B(t)|$ to denote the cardinality of bucket $B(t)$.

According to the sampling probabilities in Equation 11, we have:

$$
\begin{aligned}
&\frac{\mathbb{P}[\mathcal{R}(t) = y]}{\mathbb{P}[\mathcal{R}(t') = y]} \\
=&\Big(\frac{\exp(\frac{\epsilon}{2\triangle} u_b(t, y))}{\sum_{j=1}^{|\mathcal{V}|} \exp(\frac{\epsilon}{2\triangle} u_b(t, y_j))} \cdot \frac{1}{|B(y)|}\Big) / \Big(\frac{\exp(\frac{\epsilon}{2\triangle} u_b(t', y))}{\sum_{j=1}^{|\mathcal{V}|} \exp(\frac{\epsilon}{2\triangle} u_b(t', y_j))} \cdot \frac{1}{|B'(y)|}\Big) \\
=&\frac{\exp(\frac{\epsilon}{2\triangle} u_b(t, y))}{\exp(\frac{\epsilon}{2\triangle} u_b(t', y))} \cdot \frac{\sum_{j=1}^{|\mathcal{V}|} \exp(\frac{\epsilon}{2\triangle} u_b(t', y_j))}{\sum_{j=1}^{|\mathcal{V}|} \exp(\frac{\epsilon}{2\triangle} u_b(t, y_j))} \cdot \frac{|B'(y)|}{|B(y)|}
\end{aligned}
$$

Let $\max_{i,j}\frac{|b_i|}{|b_j|} = \mathbf{exp}(\epsilon')$, we have $\frac{|B'(y)|}{|B(y)|} \leq \mathbf{exp}(\epsilon')$. Consequently, we have:

$$
\begin{aligned}
&\frac{\mathbb{P}[\mathcal{R}(t)=y]}{\mathbb{P}[\mathcal{R}(t')=y]} \\
\leq &\frac{\mathbf{exp}(\frac{\epsilon}{2\triangle}u_b(t,y))}{\mathbf{exp}(\frac{\epsilon}{2\triangle}u_b(t',y))} \cdot \frac{\sum_{j=1}^{|\mathcal{V}|}\mathbf{exp}(\frac{\epsilon}{2\triangle}u_b(t',y_j))}{\sum_{j=1}^{|\mathcal{V}|}\mathbf{exp}(\frac{\epsilon}{2\triangle}u_b(t,y_j))} \cdot \mathbf{exp}(\epsilon') \\
= &\mathbf{exp}[\frac{\epsilon}{2}\cdot\frac{(u(t,y)-u(t',y))}{\triangle}] \cdot \frac{\sum_{j=1}^{|\mathcal{V}|}\mathbf{exp}(\frac{\epsilon}{2\triangle}u(t',y_j))}{\sum_{j=1}^{|\mathcal{V}|}\mathbf{exp}(\frac{\epsilon}{2\triangle}u(t,y_j))} \cdot \mathbf{exp}(\epsilon') \\
\leq &\mathbf{exp}(\frac{\epsilon}{2}) \cdot \frac{\sum_{j=1}^{|\mathcal{V}|}\mathbf{exp}(\frac{\epsilon}{2\triangle}u(t',y_j))}{\sum_{j=1}^{|\mathcal{V}|}\mathbf{exp}(\frac{\epsilon}{2\triangle}u(t,y_j))} \cdot \mathbf{exp}(\epsilon') \\
\leq &\mathbf{exp}(\frac{\epsilon}{2}) \cdot \mathbf{exp}(\frac{\epsilon}{2}) \cdot \mathbf{exp}(\epsilon') \\
= &\mathbf{exp}(\epsilon+\epsilon')
\end{aligned}
$$

The proof shown that our sampling mechanism satisfies $(\epsilon+\epsilon')$-DP. □

D. System Prompts

We provide the system prompts used in the text classification and open-ended text generation tasks in Table 9.

Table 9: The detailed prompts for text classification and open-ended text generation tasks.

Text Classification on SST-2: You are a helpful assistant. Classify if the sentence is positive or negative sentiment. Just gives the answer in positive, negative. —-"Content": {input}
Text Classification on QNLI: You are a helpful assistant trained to determine whether a given context sentence contains the information needed to answer the given question. If the context answers the question, respond "Yes". Otherwise, respond "No". —-"Question": {question} —-"Context": {sentence}
Text Generation on Wikitext-103-v1: Your task is to extend Prefix Text. —-"Prefix Text": {prompt} Provide only your Continuation. —-"Continuation":
Extraction of noisy response on Wikitext-103-v1: Your task is to extend the "Prefix Text". Use the "Perturbed Generation" as your primary writing material for your extension. Extract coherent and consistent text from the "Perturbed Generation" and integrate them into your continuation. Ensure a seamless alignment with the context established by the "Prefix Text". Provide only your "Extended Text". —-"Prefix Text": {prompt} —-"Perturbed Generation": {noisy_response} —-"Extended Text":

E. Perturbation Example

We provide some perturbation examples in Table 10. Given input prompts '**it 's a charming and often affecting journey .**' and '**it 's slow – very , very slow .**', we vary the perturbation methods and privacy budgets.

Toy Example. Consider the example prompt: 'it 's slow – very , very slow .'. The step-by-step perturbation process is as follows.

1. Initialize an empty token set as $T \leftarrow \phi$.
2. Tokenize the text prompt using tokenizer and obtain a list of tokens denoted as L.
3. For each token t in the list, calculate its utility score. Take 'slow' in 'it 's slow – very , very slow .' as an example.
   (a) Compute the contextual logits for the MASKED token as $L = \mathcal{M}$('it 's [MASKED] – very , very slow .').
   (b) Compute the embedding distance between 'slow' and other tokens in the vocabulary as $D = d_{euc}$('slow', $\mathcal{V}$).
   (c) Obtain the final utility score as $u = L^{\lambda_L} \cdot D^{\lambda_D}$.
   (d) Bucketize all the tokens in the vocabulary into N_b buckets according to the utility score
   (e) Sample a bucket using standard Exponential mechanism.
   (f) Uniformly sample a token $\hat{t}$ from the bucket sampled in the previous step.
   (g) Concatenate the newly sampled token to T as $T \leftarrow T|\hat{t}$.
4. Finally, we get the perturbed prompt T.

Table 10: Perturbation examples of varying privacy budgets ϵ for different methods.

Mechanism	ϵ	**Original Prompt:** **it 's a charming and often affecting journey .** **it 's slow – very , very slow .**	Rouge-L (F1)
SANTEXT	1	photo describes several brilliant nowhere practically partly sorts nietzsche clearly do chewing bolstered female reid see addictive emphasized	0.87
	2	nevertheless serviceable 2/3 reaction pretend especially daft journey . sucked addition laziness – increasingly catastrophic defies unwatchable injuries	13.43
	3	fan recognized a charming its often affecting journey . yes 's slow – very , intensity stacks .	71.31
	6	it 's a charming and often affecting journey . it 's slow – very , very slow .	99.36
SANTEXT+	1	it 's a charming and often committed journey . bible 's slow laconic very , very slow solely	52.56
	2	discussed 's combined charming banquet often majority encountered precedent it 's slow – very , very slow .	57.77
	3	indeed 's leaving charming bothered often operates journey f. it 's slow – very , very slow .	72.98
	6	it wanted puzzled enchanting and often affecting journey yes honestly s unfortunately attract however possibilities very slow .	76.37
CUSTEXT	1	this 's rather charming all may impair alone . kind 's try – something , this went .	14.50
	2	as 's rather charming others tend impair alone . kind 's try – something , this went .	17.87
	3	as 's be charming also often impair future . kind 's try – sort , this went .	22.74
	6	rather 's a charming also remain impair happiness . every 's slow – very , is try .	47.27
CUSTEXT+	1	it 's a lovely and exist consequence experiences . it 's runs – very , very slow .	50.66
	2	it 's a lovely and we consequence experiences . it 's runs – very , very slow .	52.90
	3	it 's a lovely and we altering experiences . it 's runs – very , very slow .	55.47
	6	it 's a fairytale and often affecting lifestyle . it 's pace – very , very slow .	69.41
InferDPT	1	Saw ' axis amar Translate Sie Incorrect invasion sizes . security ' instance exter – precision , Converter versus .	13.00
	6	plash ' sha outed Char TE prior affecting Joy . ILL ' ana neglect – extraordinary , intimate tox .	16.48
	10	Kit ' so Adventure charming ing according affecting analysis . Tit ' s l – verity , functional wrest .	23.20
	14	inferior ' Bike a rub and alike affecting journey . learning ' n slow – very , levant instead .	38.68
Cape	1	it's a royals andnard popularized progressing. dormitory it's libre - - very, very slower. '	38.38
	6	it's axon and quietlyately journey. will it's - - very, very civilized. of	46.85
	10	it's a charming and greatly intimidation her. it's slower - - very, very busy.	60.22
	14	it's a charming and highly painful journey. it's slow - - very, very slow.	76.49
Cape	1	it's aivably and typicallyEStream screened. Christensen it'sDragonMagazine after very, very 8.Keefe	37.60
	6	it's a really and her angry smart. all it's poor downed very, veryadium. roommate	44.55
	10	it's a telling and often affecting ride. it's a – very, very PM. where	56.46
	14	it's a w and often affecting journey. it's slow office very, very put. This	73.46