A Customized Text Sanitization Mechanism with Differential Privacy

Huimin Chen[1]*, **Fengran Mo**[2]*, **Yanhao Wang**[1], **Cen Chen**[1]†,
Jian-Yun Nie[2], **Chengyu Wang**[3], **Jamie Cui**[4]
[1]East China Normal University [2]Université de Montréal [3]Alibaba Group [4]Ant Group
saichen@stu.ecnu.edu.cn, fengran.mo@umontreal.ca
{yhwang,cenchen}@dase.ecnu.edu.cn, nie@iro.umontreal.ca
chywang2013@gmail.com, jamie.cui@outlook.com

Abstract

As privacy issues are receiving increasing attention within the Natural Language Processing (NLP) community, numerous methods have been proposed to sanitize texts subject to differential privacy. However, the state-of-the-art text sanitization mechanisms based on metric local differential privacy (MLDP) do not apply to non-metric semantic similarity measures and cannot achieve good trade-offs between privacy and utility. To address the above limitations, we propose a novel Customized Text (CusText) sanitization mechanism based on the original ϵ-differential privacy (DP) definition, which is compatible with any similarity measure. Furthermore, CusText assigns each input token a customized output set of tokens to provide more advanced privacy protection at the token level. Extensive experiments on several benchmark datasets show that CusText achieves a better trade-off between privacy and utility than existing mechanisms. The code is available at https://github.com/sai4july/CusText.

1 Introduction

In many Natural Language Processing (NLP) applications, input texts often contain sensitive information that can infer the identity of specific persons (Jegorova et al., 2021), leading to potential privacy leakage that impedes privacy-conscious users from releasing data to service providers (Carlini et al., 2019, 2021; Song and Raghunathan, 2020). Moreover, legal restrictions such as CCPA[1] and GDPR[2] may further limit the sharing of sensitive textual data. This makes NLP service providers difficult to collect training data unless the privacy concerns of data owners, including individuals and institutions, are well discoursed.

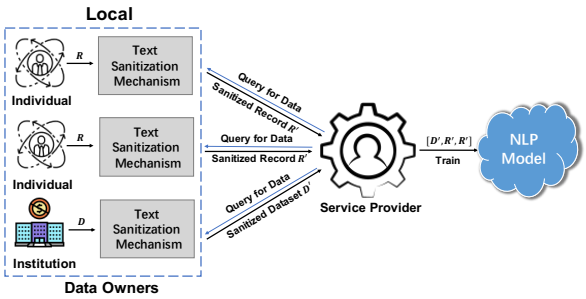

Figure 1: A privacy-preserving NLP workflow.

To address such privacy issues, great efforts (Lyu et al., 2020; Anil et al., 2022; Dupuy et al., 2022; Li et al., 2022; Mireshghallah et al., 2021) have been made to train language models (LMs) with *differential privacy* (Dwork et al., 2006) (DP), which has been regarded as the de facto standard for privacy-preserving computation. These approaches mainly focus on adding calibrated noise to gradients or text representations during the training phase so that sensitive user data cannot be inferred from trained LMs. Nevertheless, they require service providers to collect the original data for LM training. As such, data owners may still have privacy concerns when service providers are not fully trusted.

To solve the privacy problem from the root, a common paradigm is to let data owners sanitize their data *locally* before releasing them to the service provider, as shown in Figure 1. Generally, such privatization mechanisms (Feyisetan et al., 2019, 2020; Yue et al., 2021) generate a sanitized text document by replacing the original tokens (e.g., characters, words, or n-grams) in the original document sequentially with new tokens sampled from output token sets. Specifically, they adopt the Metric Local Differential Privacy (Chatzikokolakis et al., 2013) (MLDP, also known as d_χ-privacy), a relaxation of the original DP definition, to provide the privacy and utility guarantees simultaneously. On the one hand, MLDP inherits the idea of DP

*Equal contribution.
†Corresponding author.
[1]https://oag.ca.gov/privacy/ccpa
[2]https://data.europa.eu/eli/reg/2016/679/oj

Findings of the Association for Computational Linguistics: ACL 2023, pages 5747–5758
July 9-14, 2023

to ensure that the outputs of any adjacent input tokens are indistinguishable to protect the original tokens from being inferred. On the other hand, MLDP also preserves the utility of sanitized texts by assigning higher sampling probabilities to tokens that are semantically closer to the original ones. In these mechanisms, any metric distance (e.g., Euclidean distance) can be used to measure the semantic similarities between tokens.

However, the above text sanitization mechanisms suffer from two inherent limitations. First, since MLDP is specific for metric distances satisfying the triangle inequality, they do not apply to non-metric semantic similarity measures in NLP applications such as cosine similarity (Mrksic et al., 2016) and TF-IDF (Salton and Buckley, 1988). Second, they cannot achieve good privacy-utility trade-offs, i.e., either having high privacy costs with insufficient protections or resulting in low accuracy of models trained on sanitized data. We observe that the low accuracy arises as they treat each token in the text equally by assigning each input token with the same output set, which can be excessively large (e.g., the size of the output set is over 80,000). Such a huge output set leads to high costs for MLDP and thus impedes the model's utility when the privacy budget is tight.

To this end, we propose a novel Customized Text (CusText) sanitization mechanism that provides more advanced privacy protection at the token level. Specifically, to generalize CusText to all similarity measures, we turn to a mechanism that satisfies the original ϵ-Differential Privacy (ϵ-DP), i.e., Exponential Mechanism (EM) (McSherry and Talwar, 2007), to sample the output for each input token. Meanwhile, we inherit the merit of MLDP by designing an appropriate *scoring function* for EM to take into account the semantic similarities between tokens for sampling. Then, to achieve a better trade-off between privacy and utility, we design a mapping scheme to assign each input token a customized output set of a much smaller size for token-level privacy protection. Here, we can adjust a customized parameter K that determines the size of the output set for each input token for different utility-privacy trade-offs. Using the mapping scheme, we exclude most of the tokens that are semantically irrelevant to the input token from consideration and reduce the privacy costs caused by excessive output set sizes. As the privacy risks of some tokens, e.g., stopwords, are low in practice, we further propose an improved CusText+ mechanism that skips the stopwords in the sampling process to achieve higher utility without incurring greater privacy losses.

Finally, we conduct extensive experiments on three benchmark datasets to demonstrate that CusText achieves better privacy-utility trade-offs than the state-of-the-art text sanitization mechanisms in (Feyisetan et al., 2020; Yue et al., 2021). More particularly, with the same privacy parameter ϵ, the models trained on texts sanitized by CusText have significantly higher accuracy rates than those sanitized by SANTEXT (Yue et al., 2021). Furthermore, when the utilities of models are comparable, CusText provides better protection against two token inference attacks than SANTEXT.

2 Related Work

There have been numerous studies on the vulnerability of deep learning models (Carlini et al., 2019; Song and Raghunathan, 2020), including language models (Carlini et al., 2021; Zhao and Chen, 2022) (LMs), against privacy attacks. In particular, such attacks can recover sensitive user attributes or raw texts from trained models. Therefore, incorporating privacy mechanisms with rigorous guarantees is vital to protect LMs from privacy attacks.

A few attempts at applying anonymization techniques for i.i.d. data (Li et al., 2007; Machanavajjhala et al., 2007) fail to provide strong privacy protection for textual data (Zhao and Chen, 2022). Then, many efforts (Lyu et al., 2020; Anil et al., 2022; Dupuy et al., 2022; Hessel and Schofield, 2021; Li et al., 2022; Mireshghallah et al., 2021) have been made to preserve the utility of LMs on textual data with provable differential privacy (DP) guarantees. Following the application of DP in deep learning (Abadi et al., 2016), they mainly focus on adding calibrated noise to gradients or text representations during the training phase for both utility and privacy. However, they need a trustworthy server to collect original texts from data owners for model training and thus cannot be applied to the scenario without trusted servers.

To address privacy issues from the root, different (customized) local differential privacy (Duchi et al., 2013; Chatzikokolakis et al., 2013) (LDP) mechanisms have been proposed to allow data owners to sanitize their data locally before releasing them to the server. Due to the high dimensionality and complicated features of textual data, compared with

statistical analytics on i.i.d. data with LDP (Murakami and Kawamoto, 2019; Nie et al., 2019), it is much more challenging to achieve good utility-privacy trade-offs for LMs with LDP. To improve the model utility, existing methods (Feyisetan et al., 2020; Qu et al., 2021; Yue et al., 2021) rely on a relaxed notion of metric local differential privacy (Chatzikokolakis et al., 2013) (MLDP, also known as d_χ-privacy) for text sanitization. However, they either achieve reasonable accuracy only at a very low privacy protection level (e.g., with a privacy parameter $\epsilon > 10$) or become unusable (around 50% accuracy rate for the benchmark binary classification tasks) with appropriate privacy guarantees (e.g., $\epsilon = 2$). Thus, there remains much room for improvement in terms of utility-privacy trade-off for differentially private text sanitization, which is the goal of this work.

3 Preliminaries

Before introducing our CusText mechanism, we briefly review the key concepts, including ϵ-DP and exponential mechanism (EM).

Definition 1 (ϵ-differential privacy (Dwork et al., 2006)). *For a given privacy parameter $\epsilon \geq 0$, all pairs of adjacent inputs $x, x' \in \mathcal{X}$, and every possible output $y \in \mathcal{Y}$, a randomized mechanism $\mathcal{M}$ is ϵ-differentially private (DP) if it holds that*

$$\frac{\Pr[\mathcal{M}(x) = y]}{\Pr[\mathcal{M}(x') = y]} \leq e^{\epsilon}. \tag{1}$$

By definition, a smaller value of ϵ corresponds to a higher level of privacy protection. Conceptually, the notion of ϵ-DP means that an unlimited adversary cannot distinguish the two probabilistic ensembles with sufficiently small ϵ because the probabilities of adjacent tokens producing the same output token y are similar. In the context of NLP, we consider any pair of input tokens that share the same output set $\mathcal{Y}$ to be adjacent to each other. In the rest of this paper, we follow the above definition of adjacent inputs for ϵ-DP. Next, we define the Exponential Mechanism (EM) commonly used for differentially private item selection from a discrete domain, which naturally fits NLP applications due to the discrete nature of textual data.

Definition 2 (Exponential Mechanism (McSherry and Talwar, 2007)). *For a given scoring function $u : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, an exponential mechanism (EM) $\mathcal{M}(\mathcal{X}, u, \mathcal{Y})$ satisfies ϵ-differential privacy if it samples an output token $y \in \mathcal{Y}$ to perturb the input token $x \in \mathcal{X}$ with probability proportional to $e^{\frac{\epsilon \cdot u(x,y)}{2\Delta u}}$, where $u(x, y)$ denotes the score of output token y for input token x. In addition, we use $\Delta u := \max_{y \in \mathcal{Y}} \max_{x, x' \in \mathcal{X}} |u(x, y) - u(x', y)|$ to denote the sensitivity of u for EM.*

From Definition 2, we can see that smaller sensitivity makes it harder for adversaries to distinguish the original token from its adjacent tokens. In practice, for simplicity, we can normalize the scoring function u to scale its sensitivity Δu to a specific real number (e.g., 1). As such, the sampling probability of each output token y for input token x is only related to $u(x, y)$, as ϵ and Δu are known beforehand, and a larger $u(x, y)$ indicates a higher sampling probability.

In an NLP task, we suppose that each document $D = \langle R_i \rangle_{i=1}^{m}$ contains m records and each record $R = \langle t_j \rangle_{j=1}^{n}$ contains n tokens. We formulate our text sanitization task as follows: Given an input document D containing sensitive information, a set $\mathcal{X}$ of all possible input tokens, a set $\mathcal{Y}$ of all possible output tokens, and a differentially private mechanism $\mathcal{M}$ (e.g., EM in this work), it performs the mechanism $\mathcal{M}$ on each input token $t_j \in D$ to replace it with an output token t'_j from $\mathcal{Y}$ if $t_j \in \mathcal{X}$. All the tokens after replacement form the sanitized document, i.e., $D' = \langle R'_i \rangle_{i=1}^{m}$ and $R' = \langle t'_j \rangle_{j=1}^{n}$.

Following the prior work on text sanitization (Qu et al., 2021; Feyisetan et al., 2020; Yue et al., 2021), we consider a *semi-honest threat model* under the LDP setting where data owners (e.g., individuals or institutions) only submit their sanitized documents to the service provider. Malicious service providers may try to infer sensitive information from their received data. We assume that adversaries only have access to sanitized texts, and all algorithms and mechanisms are publicly known. Moreover, adversaries have unlimited computation resources.

4 The CusText Mechanism

An overview of our customized text (CusText) sanitization mechanism is presented in Figure 2. In general, it replaces each token in the original text document with a new token to achieve the privacy guarantee. It consists of two components: (1) a mapping function $f_{\mathsf{map}} : \mathcal{X} \to \{\mathcal{Y}' \subseteq \mathcal{Y}\}$ that determines the output set $\mathcal{Y}'_j$ for each input token $x_j \in \mathcal{X}$ based on semantic relevance; (2) a sampling function[3] $f_{\mathsf{sample}} : \mathcal{X}' \to \mathcal{Y}'$ based on the exponential mechanism to sample a new token from

[3]For any $\mathcal{Y}' \subseteq \mathcal{Y}$, $\mathcal{X}' = \{x \in \mathcal{X} \mid f_{\mathsf{map}}(x) = \mathcal{Y}'\}$.

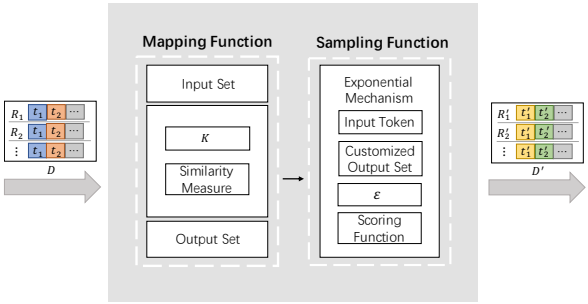

Figure 2: An overview of the CusText method.

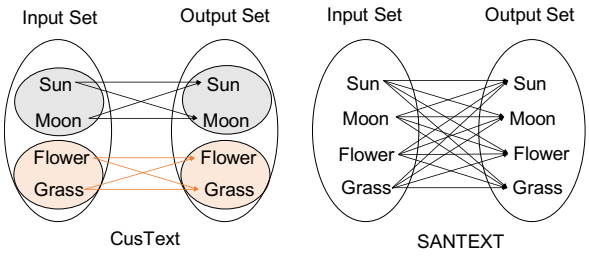

Figure 3: A comparison of the mapping schemes of SANTEXT and CusText.

an output set to sanitize the input token. Specifically, our CusText mechanism first obtains the output set $\mathcal{Y}'_j$ for each $t_j \in D$ according to f_{map}, i.e., $\mathcal{Y}'_j = f_{\mathsf{map}}(t_j)$, then samples an output token t'_j from $\mathcal{Y}'_j$ according to f_{sample}, i.e., $t'_j = f_{\mathsf{sample}}(t_j)$. Finally, after applying CusText on each input token t_j in D, the sanitized document D' is formed by all output tokens.

4.1 Mapping Function

In our CusText mechanism, the mapping function $f_{\mathsf{map}} : \mathcal{X} \rightarrow \{\mathcal{Y}' \subseteq \mathcal{Y}\}$ decides the output set for each input token. If a bunch of input tokens in $\mathcal{X}$ are mapped to the same output set $\mathcal{Y}'$, we say that they belong to the same input set $\mathcal{X}' \subseteq \mathcal{X}$ and are adjacent to each other. For the SANTEXT mechanism (Yue et al., 2021), the function $f_{\mathsf{map}} : \mathcal{X} \rightarrow \mathcal{Y}$ simply maps every input token $x \in \mathcal{X}$ to all tokens in the output set $\mathcal{Y}$. Since the size of the output set is excessively large in SANTEXT, the chances that the output token is semantically irrelevant to the original token become higher if the privacy budget is tight, thus leading to poor model utility. To overcome the above problem, CusText customizes the output set of each input token. A comparison of the mapping schemes of CusText and SANTEXT is shown in Figure 3. Before introducing how to construct f_{map}, we first discuss the requirements for mapping generation.

Algorithm 1 Token Mapping Generation

Input: Customization parameter K, input set $\mathcal{X}$, output set $\mathcal{Y} = \mathcal{X}$, similarity measure d
Output: Mapping Function f_{map}

1: **while** $|\mathcal{X}| \geq K$ **do**
2: Pick an arbitrary token x from $\mathcal{X}$
3: Initialize an output set $\mathcal{Y}' = \{x\}$ for x
4: **for all** $y \in \mathcal{Y} \setminus \{x\}$ **do**
5: Compute the similarity $d(x, y)$ of x and y
6: Add the top-$(K-1)$ tokens that are semantically closest to x to $\mathcal{Y}'$ based on $d(\cdot, \cdot)$
7: **for all** $x' \in \mathcal{Y}'$ **do**
8: Assign the output set of x' as $\mathcal{Y}'$
9: Update $\mathcal{X} \leftarrow \mathcal{X} \setminus \mathcal{Y}'$ and $\mathcal{Y} \leftarrow \mathcal{Y} \setminus \mathcal{Y}'$
10: Perform Lines 2–9 for the remaining tokens in $\mathcal{X}$ and $\mathcal{Y}$ with customization parameter $K' = |\mathcal{X}|$
11: **return** f_{map}

Mapping Strategy. According to the sizes of $\mathcal{X}'$ and $\mathcal{Y}'$ as indicated by the mapping function f_{map}, we can categorize the token mappings into four types: 1-to-1, N-to-1, 1-to-N, and N-to-M, where 1, N, and M denote the size of the input/output token sets and $N, M > 1$. Theoretically, CusText can provide ϵ-differential privacy protection to all input tokens only if the mappings of all input tokens in the set $\mathcal{X}$ are N-to-M or N-to-1 mappings so that every input token in $\mathcal{X}$ has at least one adjacent token. This is because the goal of applying ϵ-DP is to make any two adjacent tokens indistinguishable so that the input token cannot be effectively inferred. Moreover, following prior work (Feyisetan et al., 2020; Yue et al., 2021), we consider that $\mathcal{X}$ is equal to $\mathcal{Y}$ (i.e., $\mathcal{X} = \mathcal{Y}$) in CusText, as they both correspond to the vocabulary of a specific language. Also, any input token x is always included in its output set because it must be the closest to itself. Next, we describe our mapping generation that can satisfy all the above requirements.

Mapping Function Generation. The generation of the mapping function $f_{\mathsf{map}} : \mathcal{X} \rightarrow \{\mathcal{Y}' \subseteq \mathcal{Y}\}$ is to assign the customized output set for each input token based on semantic relevance. The semantic relevance can be defined by any similarity measure $d : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$. In practice, we use the Euclidean distance or cosine similarity on the vector representations of tokens, such as Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and Counter-Fitting (Mrksic et al., 2016) as the similarity measure. Then, we fix the sizes of all output sets to K. Specifically, we pick an arbitrary unmapped token $x \in \mathcal{X}$, find the K tokens semantically closest to x, generate an K-to-K mapping from all the K tokens to themselves, and remove the mapped

tokens from $\mathcal{X}$ and $\mathcal{Y}$ at each round until either all tokens are mapped or fewer than K tokens remain unmapped. In the latter case, the remaining tokens will constitute a K'-to-K' mapping where $K' \in [1, K)$. The pseudocode of generating the mapping function f_{map} is presented in Algorithm 1.

4.2 Sampling Function

Based on the mapping function $f_{\mathsf{map}} : \mathcal{X} \to \{\mathcal{Y}' \subseteq \mathcal{Y}\}$, a sampling function $f_{\mathsf{sample}} : \mathcal{X}' \to \mathcal{Y}'$ is designed to sample the output token for each input token. CusText adopts the exponential mechanism (McSherry and Talwar, 2007) (EM) for sampling. We need to design an appropriate scoring function for EM to strike a good utility-privacy trade-off. We obey the following two rules when designing the scoring function $u : \mathcal{X}' \times \mathcal{Y}' \to \mathbb{R}$.

1. The score of each pair of input and output tokens should be bounded, i.e., $\forall x \in \mathcal{X}'$, $\forall y \in \mathcal{Y}'$, $u(x, y) < B$, so that the sensitivity Δu of u is bounded for satisfying ϵ-DP.

2. The higher the semantic similarity between a pair of input and output tokens is, the higher the score is, i.e., $\forall x \in \mathcal{X}'$, $\forall y, y' \in \mathcal{Y}'$, if y is semantically closer to x than y', $u(x, y) > u(x, y')$. This ensures the candidates semantically closer to x have higher probabilities of being sampled, which inherits the advantage of d_χ-privacy (Chatzikokolakis et al., 2013).

For the scoring function, we are based on the same similarity function as used in the mapping scheme, e.g., Euclidean distance or cosine similarity on the vector representations of tokens (Mikolov et al., 2013; Pennington et al., 2014; Mrksic et al., 2016). Generally, according to the correlation between scores and semantic closeness, all the similarity measures can be categorized into two types, i.e., *negative correlation* and *positive correlation*. For instance, Euclidean distance and cosine similarity are *negative* and *positive correlation* measures, respectively, as a smaller Euclidean distance and a larger cosine value between two vectors imply higher semantic closeness of their corresponding tokens. Next, we will design scoring functions for both types of similarity measures.

Scoring Function for Negative Correlation Measures. We take Euclidean distance as an example to design the scoring function $u : \mathcal{X}' \times \mathcal{Y}' \to \mathbb{R}$. For any input set $\mathcal{X}'$ and its corresponding output set $\mathcal{Y}'$, we first compute the Euclidean distance $d(x, y)$ between each $x \in \mathcal{X}'$ and $y \in \mathcal{Y}'$. Specifically, we have $d(x, y) = \|\Phi(x) - \Phi(y)\|_2$, where $\Phi(x)$ and $\Phi(y)$ are the vector representations of x and y, respectively. Then, we normalize the distances of all pairs of tokens to the range $[0, 1]$ as $d'(x, y) = \frac{d(x,y)-d_{min}}{d_{max}-d_{min}}$, where $d_{min} = \min_{x \in \mathcal{X}', y \in \mathcal{Y}'} d(x, y)$ and $d_{max} = \max_{x \in \mathcal{X}', y \in \mathcal{Y}'} d(x, y)$. Finally, we transform the normalized distance $d'(x, y)$ into the score of output token y for input token x as $u(x, y) = -d'(x, y)$. After the above transformation, a more similar pair x, y of input and output tokens has a higher score $u(x, y)$. Finally, by repeating the above steps on all disjoint partitions of adjacent tokens with the same $\mathcal{X}'$ and $\mathcal{Y}'$, we have obtained the scoring functions for all tokens.

Algorithm 2 Document Sanitization

Input: Original document $D = \langle R_i \rangle_{i=1}^m$, sampling function f_{sample}, stopword list T
Output: Sanitized document D'

```
Initialize the sanitized document D' = ∅
for all record R ∈ D do
    Initialize the sanitized record R' = ∅
    for all token x ∈ R do
        if 'CusText+' is used and x ∈ T then
            Append x to R'
        else
            x' ← f_sample(x) and append x to R'
    Add R' to D'
return D'
```

Scoring Function for Positive Correlation Measures. We take cosine similarity as another example to design the scoring function u. For any input set $\mathcal{X}'$ and its corresponding output set $\mathcal{Y}'$, we also compute the cosine similarity $\cos(x, y)$ between each $x \in \mathcal{X}'$ and $y \in \mathcal{Y}'$, where $\cos(x, y) = \frac{\langle \Phi(x), \Phi(y) \rangle}{\|\Phi(x)\| \cdot \|\Phi(y)\|}$ and $\Phi(x)$ and $\Phi(y)$ are the vector representations of x and y. Then, the normalization procedure is the same as that for Euclidean distance, but we use the normalized distance, instead of its additive inverse, in the score function, i.e., $u(x, y) = \frac{d(x,y)-d_{min}}{d_{max}-d_{min}}$. Finally, we repeat the above steps on all disjoint partitions of adjacent tokens to obtain all scoring functions.

Sampling Procedure. After acquiring the scoring function u for each input token x, the sampling function f_{sample} is used to generate the sanitized token x' for x based on the exponential mechanism $\mathcal{M}(\{x\}, u, \mathcal{Y}')$ with a privacy parameter $\epsilon > 0$. The pseudocode of sanitizing a document based on f_{sample} is provided in Algorithm 2. Theoretically, it guarantees that f_{sample} satisfies ϵ-DP. For any input set $\mathcal{X}'$ and its corresponding output set $\mathcal{Y}'$,

the sensitivity Δu between any two adjacent input tokens $x, x' \in \mathcal{X}'$ is bound by 1 according to the design of the scoring function u, i.e.,

$$\Delta u = \max_{y \in \mathcal{Y}'} \max_{x, x' \in \mathcal{X}'} |u(x, y) - u(x', y)| = 1$$

Given a privacy parameter $\epsilon > 0$, the probability of obtaining an output token $y \in \mathcal{Y}'$ for an input token $x \in \mathcal{X}'$ is as follows:

$$\Pr[f_{\mathsf{sample}}(x) = y] = \frac{\exp(\frac{\epsilon u(x,y)}{2\Delta u})}{\sum_{y' \in \mathcal{Y}'} \exp(\frac{\epsilon u(x,y')}{2\Delta u})}$$

We can prove that the sampling function f_{sample} satisfies ϵ-DP because, for any two input tokens $x, x' \in \mathcal{X}'$ and output token $y \in \mathcal{Y}'$, it holds that

$$\frac{\Pr[f_{\mathsf{sample}}(x) = y]}{\Pr[f_{\mathsf{sample}}(x') = y]} = \frac{\frac{\exp(\frac{\epsilon u(x,y)}{2\Delta u})}{\sum_{y' \in \mathcal{Y}'} \exp(\frac{\epsilon u(x,y')}{2\Delta u})}}{\frac{\exp(\frac{\epsilon u(x',y)}{2\Delta u})}{\sum_{y' \in \mathcal{Y}'} \exp(\frac{\epsilon u(x',y')}{2\Delta u})}}$$

$$= e^{\frac{\epsilon \cdot (u(x,y) - u(x',y))}{2\Delta u}} \cdot \left(\frac{\sum_{y' \in \mathcal{Y}'} \exp(\frac{\epsilon u(x',y')}{2\Delta u})}{\sum_{y' \in \mathcal{Y}'} \exp(\frac{\epsilon u(x,y')}{2\Delta u})} \right)$$

$$\leq e^{\frac{\epsilon}{2}} \cdot e^{\frac{\epsilon}{2}} \cdot \left(\frac{\sum_{y' \in \mathcal{Y}'} \exp(\frac{\epsilon u(x,y')}{2\Delta u})}{\sum_{y' \in \mathcal{Y}'} \exp(\frac{\epsilon u(x,y')}{2\Delta u})} \right) = e^{\epsilon}.$$

4.3 The CusText+ Mechanism

Since not all tokens contain sensitive information, our CusText mechanism that replaces all tokens might be over-protective. Therefore, we can retain non-sensitive original tokens with low privacy risk (e.g., stopwords) to improve the utility of the sanitized text. In practice, we have a predefined list of stopwords T (e.g., the collection of stopwords in the NLTK library), check whether each token x is included in T, and keep x in the sanitized document if $x \in T$ or replace x with $x' = f_{\mathsf{sample}}(x)$ otherwise. The above procedure is called the CusText+ mechanism and is also described in Algorithm 2.

5 Experiments

5.1 Experimental Setup

Following (Feyisetan et al., 2020; Yue et al., 2021), we choose two datasets from the GLUE benchmark (Wang et al., 2019) and one medical dataset MedSTS (Wang et al., 2020), which all contain sensitive information, in our experiments. Detailed descriptions of the three datasets are as follows:

- **SST-2** is a popular movie reviews dataset with 67k training samples and 1.8k test samples for sentiment classification, where *accuracy* is used as the evaluation metric.
- **MedSTS** is a medical dataset with 1,642 training samples and 412 test samples for semantic similarity computation, where *Pearson correlation coefficient* is used for evaluation.
- **QNLI** is a sentence dataset with 105k training samples and 5.2k test samples for sentence-pair classification, where *accuracy* is used as the evaluation metric.

In the experiments, we compare CusText with two existing text sanitization mechanisms, i.e., FBDD (Feyisetan et al., 2020) and SANTEXT (Yue et al., 2021). In the training phase, we perform each mechanism to sanitize the training data and then use the sanitized documents to fine-tune the pre-trained model. In the evaluation phase, we sanitize the test data by the same mechanism as used for training. When producing the sanitized documents, both the input set $\mathcal{X}$ and output set $\mathcal{Y}$ are assigned to the vocabulary of Counter-Fitting (Mrksic et al., 2016) (of size 65,713), and out-of-vocabulary (OOV) tokens except numbers are retained. For a fair comparison, we adopt the same vocabulary in GloVe (Pennington et al., 2014) as in Counter-Fitting. The Euclidean distance and cosine similarity are used as the similarity measures for GloVe and Counter-Fitting, respectively. We use the stopword list in NLTK for CusText+. For each downstream task, we set the maximum sequence length to 128 and the training epoch to 3. On the SST2 and QNLI datasets, we set the batch size to 64 and the learning rate to 2×10^{-5} using *bert-base-uncased*[4] as the pre-trained model. On the MedSTS dataset, we set the batch size to 8 and the learning rate to 5×10^{-5} using *ClinicalBERT* (Alsentzer et al., 2019) as the pre-trained model. Other hyperparameters are the same as those used in the default Transformer model (Wolf et al., 2020). All experiments were conducted on a server with two Intel Xeon Silver 4210R 2.40GHz CPUs and one NVIDIA Tesla V100 SXM2 (32GB).

5.2 Experimental Results

Comparison of Different Mechanisms for Text Sanitization. In this experiment, we fix the customization parameter K to 20 in CusText and CusText+ and vary the privacy parameter $\epsilon = 1, 2, 3$

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

Mechanisms	SST2			MedSTS			QNLI		
	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$
Random	0.5014			0.0382			0.5037		
FBDD	0.5022	0.5041	0.5032	0.0321	0.0368	0.0411	0.5021	0.5152	0.5368
SANTEXT	0.5014	0.4827	0.5091	0.0850	0.1673	0.1124	0.5304	0.5302	0.5357
CusText	0.6985	0.7172	0.7029	0.4957	0.5112	0.5242	0.6926	0.6884	0.7133
SANTEXT+	0.7211	0.7446	0.7260	0.4143	0.4271	0.5423	0.7607	0.7636	0.7493
CusText+	0.7501	0.7452	0.7683	0.6172	0.6316	0.6213	0.7528	0.7602	0.7740
Original	0.9050			0.7598			0.9096		

Table 1: Utility comparison of different sanitization mechanisms at similar privacy levels.

for DP. The evaluation of the effect of K on the performance of CusText will be presented later. Furthermore, we choose GloVe as the token embedding in CusText and CusText+ for a fair comparison since FBDD, SANTEXT, and SANTEXT+ cannot apply the Counter-Fitting embedding. This is because they only work with metric distances (e.g., Euclidean distance in GloVe) due to the inherent limitation of MLDP and thus cannot handle the non-metric cosine similarity in Counter-Fitting. Finally, because a mechanism will be ϵ-DP if it is ϵ'-MLDP (Chatzikokolakis et al., 2013), where $\epsilon = \epsilon' \cdot d_{max}$ and $d_{max} = \max_{x \in \mathcal{X}, y \in \mathcal{Y}} d(x, y)$, we re-scale the privacy parameter ϵ in FBDD, SANTEXT, and SANTEXT+ with d_{max} to align their privacy levels to be similar to our mechanisms.

Table 1 presents the utilities of different text sanitization mechanisms with ϵ-DP ($\epsilon = 1, 2, 3$) on three datasets. The results demonstrate the huge advantages of CusText compared with two existing mechanisms, i.e., FBDD and SANTEXT, which achieves over 20% improvements in accuracy on the SST-2 and QNLI datasets and more than 50% improvement in Pearson correlation coefficient on the MedSTS dataset. Compared with SANTEXT and CusText, their improved versions, i.e., SANTEXT+ and CusText+, exhibit significantly better performance because they keep some original tokens to preserve original semantics. Generally, the results indicate the superior performance of CusText by showing that using a customized, smaller output set for each input token can lead to better utilities at similar (theoretical) privacy levels.

Privacy-Utility Trade-off. Subsequently, we compare SANTEXT and CusText in terms of privacy-utility trade-offs. As shown in (Yue et al., 2021) as well as our previous results, FBDD has lower performance than SANTEXT and CusText and thus is not compared in the remaining experiments anymore. To alleviate the effects of different DP definitions in SANTEXT and CusText, we do not use

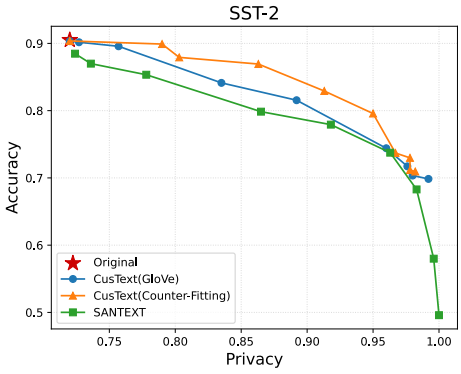

Figure 4: Privacy-utility trade-offs in terms of success rates of mask token inference attacks vs. accuracy rates by varying the privacy parameter $\epsilon \in [0.01, 50]$ on the SST-2 dataset. Here, "Original" denotes the result on unsanitized data.

the privacy parameter ϵ, which corresponds to the worst possible privacy leakage but may not reveal the privacy protection level in practice. Alternatively, we adopt two privacy attacks to evaluate the privacy protection levels: One is the *Mask Token Inference Attack* in (Yue et al., 2021), and the other is *Query Attack* proposed in this work.

We first present the results for mask token inference attacks. To recover raw texts from sanitized texts, an adversary can use the pre-trained BERT model to help infer the original tokens since it is trained via masked language modeling. It replaces each token with a special token "[MASK]" in the sanitized text sequentially, inputs the masked text to BERT, and acquires the predicted output of "[MASK]" as the original token. Then, we consider the attack successful if the output token is the same as the input. Finally, we compute the success rate among all attacks, denoted as r_{mask}, and define the privacy protection level as $1 - r_{mask}$.

Figure 4 illustrates the privacy-utility trade-offs of CusText (based on GloVe and Counter-Fitting, respectively) and SANTEXT (based on GloVe) by varying the value of ϵ on the SST-2 dataset. The

Token	SANTEXT			CusText (GloVe)				CusText (Counter-Fitting)			
	$\epsilon' = 1$	$\epsilon' = 2$	$\epsilon' = 3$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 8$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 8$
she	2350	35	4	1000	200	80	5	5500	1000	320	4
car	1300	14	1	1220	250	90	6	420000	90000	31000	3200
alice	1550	20	3	1190	240	100	6	1700	360	120	9
happy	3200	55	4	1490	290	110	8	320000	55000	21500	1500
Accuracy	0.4959	0.5799	0.7958	0.6985	0.7172	0.7029	0.8155	0.7117	0.7370	0.7298	0.7957

Table 2: Results for query attacks on four selected tokens in the SST-2 dataset.

results confirm that CusText achieves better utility-privacy trade-offs than SANTEXT and remains a relatively good utility (accuracy at around 0.7) when the privacy level approaches 1 (over 0.98). In comparison, SANTEXT degenerates to a random classifier (accuracy at around 0.5). Meanwhile, the results also imply that Counter-Fitting works better with CusText than GloVe. The higher performance of Counter-Fitting can be attributed to its better representations of synonyms.

We then describe the results for query attacks. Since the input token is contained in its corresponding output set and always has the highest score, the probability that it is sampled by f_{sample} is also the highest among all output tokens. An adversary can determine the input token by querying the data owner for the sanitized document multiple times, as the input token will have the highest frequency among all output tokens after a sufficiently large number of queries. Thus, we use the smallest number N of queries an adversary needs to infer the input token at a confidence level of 95% as a new measure of the privacy protection level. Here, the larger the value of N is, the higher the level of privacy protection is. In the experiment, we obtain the value of N by using the Monte Carlo method (Gentle, 2009) to sample the output tokens until the confidence level of determining the input token from the output distribution reaches 95%.

Table 2 further confirms that CusText achieves better privacy-utility trade-offs than SANTEXT. Although SANTEXT achieves a good utility when $\epsilon' = 3$ (i.e., with 3-MLDP), it almost provides no privacy protection as input tokens can be inferred by performing only a few queries. CusText (with either GloVe or Counter-Fitting) remains relatively good privacy protection levels when $\epsilon = 3$ (i.e., with 3-DP) while achieving high utilities. Generally, Counter-Fitting also outperforms GloVe for CusText. But the privacy protections for different tokens vary very much for Counter-Fitting: "she" and "alice" are more vulnerable than "car" and "happy". This is because "she" and "alice" are mapped with semantically less relevant tokens than themselves in the mapping function generation.

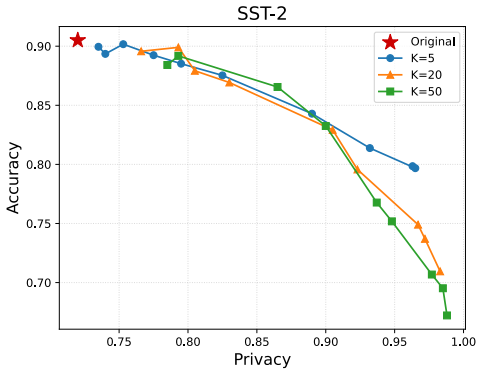

Figure 5: Privacy-utility trade-offs of CusText with different customization parameters K by varying the privacy parameter $\epsilon \in [0.001, 50]$ on the SST-2 dataset.

Effect of K on CusText. To test the effect of K on CusText in practice, we study the privacy-utility trade-offs with different customization parameters $K = 5, 20, 50$ on the SST-2 dataset. We choose the mask token inference attack as the privacy metric since its performance is more semantically related. Then, we use Counter-Fitting for its better performance than GloVe, as depicted previously.

The results for different K's are presented in Figure 5. We observe that the performance of CusText is generally stable for different K's. But it achieves slightly better utilities when K is smaller at relatively higher privacy protection levels (> 0.9). This is because, on the one hand, the semantic similarity of output tokens to the input token will be higher when K is smaller. However, on the other hand, a smaller K will also make it easier to infer the input token, thus lowering the privacy protection levels (e.g., for $K = 5$, it does not exceed 0.96 even when ϵ has been decreased to 0.001).

6 Concluding Remarks

In this work, we study the problem of differentially private text sanitization. We propose a novel CusText mechanism consisting of a mapping scheme

to assign each input token a customized output set and sampling function generation methods based on the mapping scheme and exponential mechanism to reduce privacy costs while improving the utilities of sanitized texts. Extensive experiments demonstrate that CusText achieves better privacy-utility trade-offs than state-of-the-art text sanitization mechanisms. In the future, we will explore how to improve our mechanism by adaptively allocating privacy costs across tokens and find a better way to decide whether a token is sensitive than based on a pre-defined stopword list.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (under Grant numbers 62202170, 62202169) and Alibaba Group through the Alibaba Innovation Research Program.

Limitations

First, as indicated in Table 2, different tokens are not equally vulnerable to privacy attacks. As such, assigning every token with the same output size K and privacy parameter ϵ might not be an ideal choice. An improved method would be to adaptively allocate privacy costs across tokens so that all of them are adequately protected. Second, we adopt two simple strategies to decide whether a token is sensitive: assuming all tokens are sensitive or based on a pre-defined stopword list. However, the prior might be over-protective, but the latter can lead to privacy leakage since stopwords might help infer other sanitized tokens. Therefore, a more flexible and practical way to decide the sensitivity of tokens is required.

References

Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *CCS*, pages 308–318.

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78.

Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. 2022. Large-scale differentially private BERT. In *EMNLP (Findings)*, pages 6481–6491.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, pages 267–284.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. In *USENIX Security Symposium*, pages 2633–2650.

Konstantinos Chatzikokolakis, Miguel E. Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. 2013. Broadening the scope of differential privacy using metrics. In *Privacy Enhancing Technologies (PETS)*, pages 82–102.

John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2013. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438.

Christophe Dupuy, Radhika Arava, Rahul Gupta, and Anna Rumshisky. 2022. An efficient DP-SGD mechanism for large scale NLU models. In *ICASSP*, pages 4118–4122.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography (TCC)*, pages 265–284.

Oluwaseyi Feyisetan, Borja Balle, Thomas Drake, and Tom Diethe. 2020. Privacy- and utility-preserving textual analysis via calibrated multivariate perturbations. In *WSDM*, pages 178–186.

Oluwaseyi Feyisetan, Tom Diethe, and Thomas Drake. 2019. Leveraging hierarchical representations for preserving privacy and utility in text. In *ICDM*, pages 210–219.

James E. Gentle. 2009. Monte Carlo methods for statistical inference. In *Computational Statistics*, pages 417–433. Springer.

Jack Hessel and Alexandra Schofield. 2021. How effective is BERT without word ordering? Implications for language understanding and data privacy. In *ACL/IJCNLP (Short Papers)*, pages 204–211.

Marija Jegorova, Chaitanya Kaul, Charlie Mayor, Alison Q. O'Neil, Alexander Weir, Roderick Murray-Smith, and Sotirios A. Tsaftaris. 2021. Survey: Leakage and privacy at inference time. *arXiv:2107.01614*.

Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. 2007. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, pages 106–115.

Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. 2022. Large language models can be strong differentially private learners. In *ICLR*.

Lingjuan Lyu, Xuanli He, and Yitong Li. 2020. Differentially private representation for NLP: Formal guarantee and an empirical study on privacy and fairness. In *EMNLP (Findings)*, pages 2355–2365.

Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. 2007. *L*-diversity: Privacy beyond *k*-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3:1–3:52.

Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *FOCS*, pages 94–103.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv:1301.3781*.

Fatemehsadat Mireshghallah, Huseyin A. Inan, Marcello Hasegawa, Victor Rühle, Taylor Berg-Kirkpatrick, and Robert Sim. 2021. Privacy regularization: Joint privacy-utility optimization in LanguageModels. In *NAACL-HLT*, pages 3799–3807.

Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Lina Maria Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2016. Counter-fitting word vectors to linguistic constraints. In *NAACL-HLT*, pages 142–148.

Takao Murakami and Yusuke Kawamoto. 2019. Utility-optimized local differential privacy mechanisms for distribution estimation. In *USENIX Security Symposium*, pages 1877–1894.

Yiwen Nie, Wei Yang, Liusheng Huang, Xike Xie, Zhenhua Zhao, and Shaowei Wang. 2019. A utility-optimized framework for personalized private histogram estimation. *IEEE Trans. Knowl. Data Eng.*, 31(4):655–669.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Chen Qu, Weize Kong, Liu Yang, Mingyang Zhang, Michael Bendersky, and Marc Najork. 2021. Natural language understanding with privacy-preserving BERT. In *CIKM*, pages 1488–1497.

Gerard Salton and Chris Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.*, 24(5):513–523.

Congzheng Song and Ananth Raghunathan. 2020. Information leakage in embedding models. In *CCS*, pages 377–390.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Yanshan Wang, Naveed Afzal, Sunyang Fu, Liwei Wang, Feichen Shen, Majid Rastegar-Mojarad, and Hongfang Liu. 2020. MedSTS: a resource for clinical semantic textual similarity. *Lang. Resour. Eval.*, 54(1):57–72.

Thomas Wolf, Lysandre Debut, et al. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP (Demos)*, pages 38–45.

Xiang Yue, Minxin Du, Tianhao Wang, Yaliang Li, Huan Sun, and Sherman S. M. Chow. 2021. Differential privacy for text analytics via natural text sanitization. In *ACL/IJCNLP (Findings)*, pages 3853–3866.

Ying Zhao and Jinjun Chen. 2022. A survey on differential privacy for unstructured data content. *ACM Comput. Surv.*, 54(10s):207:1–207:28.

ACL 2023 Responsible NLP Checklist

A For every submission:

☐ A1. Did you describe the limitations of your work?
Left blank.

☐ A2. Did you discuss any potential risks of your work?
Left blank.

☐ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☐ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☐ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Left blank.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C ☐ Did you run computational experiments?

Left blank.

☐ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☐ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Left blank.

☐ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Left blank.

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Left blank.

D ☐ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Left blank.