TextObfuscator: Making Pre-trained Language Model a Privacy Protector via Obfuscating Word Representations

Xin Zhou[1*], **Yi Lu**[5*†], **Ruotian Ma**[1], **Tao Gui**[2‡],
Yuran Wang[4], **Yong Ding**[4], **Yibo Zhang**[4], **Qi Zhang**[1], **Xuanjing Huang**[1, 3‡]
[1]School of Computer Science, Fudan University, Shanghai, China
[2] Institute of Modern Languages and Linguistics, Fudan University, Shanghai, China
[3] International Human Phenome Institutes, Shanghai, China
[4] Honor Device Co., Ltd
[5] School of Computer Science and Engineering, Northeastern University, Shenyang, China
{xzhou20, tgui, qz}@fudan.edu.cn, luyi18171297680@gmail.com

Abstract

In real-world applications, pre-trained language models are typically deployed on the cloud, allowing clients to upload data and perform compute-intensive inference remotely. To avoid sharing sensitive data directly with service providers, clients can upload numerical representations rather than plain text to the cloud. However, recent text reconstruction techniques have demonstrated that it is possible to transform representations into original words, suggesting that privacy risk remains. In this paper, we propose **TextObfuscator**, a novel framework for preserving inference privacy by applying random perturbations to clustered representations. The random perturbations make each word representation indistinguishable from surrounding functionally similar representations, thus obscuring word information while retaining the original word functionality. To achieve this, we utilize prototypes to learn clustered representations, where words of similar functionality are encouraged to be closer to the same prototype during training. Additionally, we design different methods to find prototypes for token-level and sentence-level tasks, which can improve performance by incorporating semantic and task information. Experimental results on token and sentence classification tasks show that TextObfuscator achieves improvement over compared methods without increasing inference cost.

1 Introduction

Pre-trained language models (PLMs) have achieved impressive performance on various NLP downstream tasks (Devlin et al., 2018; Brown et al., 2020; Qiu et al., 2020), but they also come with increased model size and significant computational requirements. In real-world applications, these large-scale models are often offered as an inference service (Altman, 2022). The service providers train PLMs for target tasks and deploy them on the cloud. Clients who lack high-computation resources can query these service with their input and obtains the desired responses (DALE, 2015).

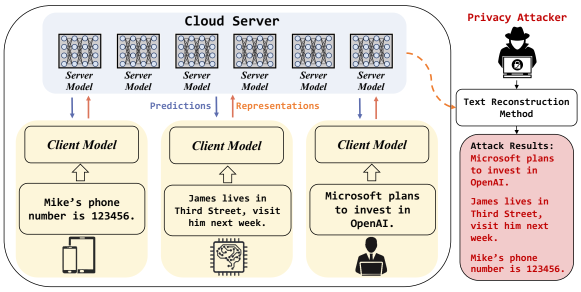
Cloud Server
Server Model
Server Model
Server Model
Server Model
Server Model
Server Model
Predictions
Representations
Client Model
Client Model
Client Model
Mike's phone number is 123456.
James lives in Third Street, visit him next week.
Microsoft plans to invest in OpenAI.
Privacy Attacker
Text Reconstruction Method
Attack Results:
Microsoft plans to invest in OpenAI.
James lives in Third Street, visit him next week.
Mike's phone number is 123456.

Figure 1: Illustration of the inference services and privacy risks. Although clients upload word representations instead of plain text to the cloud server, privacy attackers can still transform representations into original texts. Clients are still at privacy risk.

Unfortunately, current inference services are plagued by serious privacy concerns (Lehmkuhl et al., 2021). Client data may contain sensitive information such as names, addresses, and even trade secrets, sharing such information with service providers compromises the privacy of clients. To address privacy risks, a naive solution is for clients to generate shallow representations on their devices and upload numerical representations to the cloud for subsequent inference, as shown in Figure 1. However, recent text reconstruction methods (Song and Raghunathan, 2020; Pan et al., 2020) have shown that word representations can be easily transformed into raw texts, indicating privacy risk remains. Inference service without privacy

*Equal contribution.
†Work done during the internship at Fudan NLP Lab.
‡Corresponding authors.

guarantees is not only unacceptable for clients but also illegal for service providers[1].

Recent literature has proposed various methods to mitigate privacy leakage in representation. For example, Chen et al. (2022b) and Hao et al. (2022) have applied homomorphic encryption (Gentry, 2009) to transformer-based models, which enables computations to be performed on encrypted data. But homomorphic encryption often incurs significant **computation time and communication costs** (Gilad-Bachrach et al., 2016), making it impractical for real-world applications. Alternatively, several studies have adapted differential privacy (Lyu et al., 2020a; Hoory et al., 2021; Yue et al., 2021a) and adversarial training (Li et al., 2018; Coavoux et al., 2018; Plant et al., 2021) to reduce the privacy information contained in representations. However, in our scenario, the privacy information pertains to each word, and **reducing word information in the shallow layer can harm subsequent inference, thus degrading performance** (Jawahar et al., 2019), especially in token-level tasks.

In this paper, we propose TextObfuscator, a novel paradigm for privacy-preserving inference. The key idea of our method is to learn private representations that **obscure original word information while preserving original word functionality**. Specifically, we find prototypes for each word and encourage functionally similar words close to the same prototype during training. Subsequently, random perturbations are applied to these clustered representations, which yields two key benefits. Firstly, it obscures original word information as the perturbed representations are indistinguishable from those clustered around them, making it harder for privacy attackers to reconstruct original words, thus protecting privacy. Secondly, it maintains the original word functionality as the perturbed representations stay within the same functional clusters, leading to improved performance.

To learn clustered representations, we have designed different methods to find suitable prototypes for token and sentence classification. For token-level tasks, each word is assigned a label that serves as a prototype indicator (Snell et al., 2017). But for sentence-level tasks, there is no explicit prototype indicator for words. Therefore, the clustering algorithm is used for word assignment. Based on clustering results, we take the cluster centers as prototypes and assign semantically similar words to the same prototype. However, semantic-based clustering may lead to keywords from different classes being clustered together, hindering target tasks. For example, if "good" and "bad" play the same role, the sentiment of a sentence will be ambiguous. To address this, we utilize TF-IDF to identify keywords from different classes and use these keywords to re-divide the clustering results. Our codes are publicly available at `https://github.com/xzhou20/TextObfuscator`.

Our contribution can be summarized as follows:

- We propose TextObfuscator, a novel representation learning method for privacy-preserving inference by obfuscating representations.
- We propose to combine semantic and task information for finding prototypes, which leads to improved performance.
- We evaluate TextObfuscator on several NLP tasks including token and sentence classification, and demonstrate its effectiveness in protecting privacy while improving performance.

2 Preliminaries

2.1 Inference as Service

Suppose a client wants to query the inference service without leaking privacy, client can perform acceptable computations on their device (such as intelligent chips, smartphones, and personal computers) to obtain the word representations $\mathbf{H} = f_{\theta_c}(X)$, where f_{θ_c} is the client model released by service provider. Then numerical $\mathbf{H}$ instead of plain text X are uploaded to the cloud. The PLM f_{θ_s} deployed on the server performs subsequent compute-intensive inference $\mathbf{Y} = f_{\theta_s}(\mathbf{H})$ and sends predictions $\mathbf{Y}$ back to the client. In this scenario, only representations are shared with service providers, avoiding the leakage of text.

2.2 Privacy Threats

Privacy threats in the inference phase mainly come from service providers, who have access to the client model f_{θ_c}, server model f_{θ_s} and client's word representation $\mathbf{H}$. Recently studies (Song and Raghunathan, 2020) have shown that the word information in the representation is sufficient to reconstruct the original text. For example, the shallow representations are usually similar to their embedding, privacy attackers can compare the

[1]https://www.consilium.europa.eu/en/policies/data-protection/data-protection-regulation/

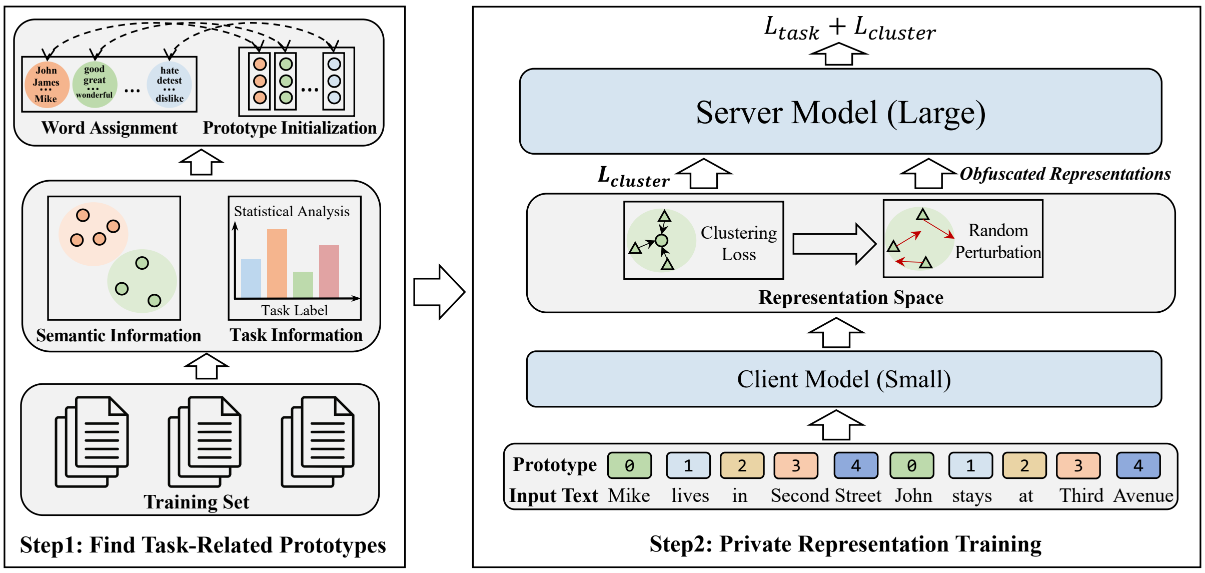

Figure 2: Overview of the proposed method. In the first step, we find the task-related prototype using semantic and task information. Prototypes get initialized, and each word is assigned to a prototype. In the second step, we use word representation from the client model to calculate cluster loss, which encourages functionally similar representations clustered together. Then random perturbations are applied to representations to mislead privacy attackers, and we send these perturbed representations to the server model for task loss. Finally, we optimize the small client and large server models via cluster loss and task loss.

representation and embedding matrices to identify the most similar word. Furthermore, service providers can generate word representations via client model and train a powerful inversion model to directly transform the representations back into the original text $X = f_{\theta_{inv}}(\mathbf{H})$, even if privacy-preserving methods have been applied on $\mathbf{H}$. The challenge in private inference lies in ensuring $f_{\theta_{inv}}$ cannot learn useful word information from $\mathbf{H}$ to reconstruct X, while that the information in $\mathbf{H}$ is sufficient for subsequent inference.

3 Our Method

3.1 Overview

In this section, we present TextObfuscator, a novel framework for privacy-preserving inference. Our method learns private representation from a new perspective, which aims to obfuscate rather than reduce word information. The overall framework of TextObfuscator is shown in Figure 2. We first find prototypes for each word using semantic and task information, these prototypes are used to encourage functionally similar words to cluster together in the training phase. Then random perturbations are applied to each representation, making them indistinguishable from the surrounding clustered word representations. Even if privacy attackers get representation, they cannot establish the correct connection between the obfuscated representations and original words. Furthermore, these representations maintain original word functionality as they remain close to their prototype. Next, we introduce how to find prototypes and learn private representation.

3.2 Find Task-Related Prototypes

In step one, we introduce two crucial components. One is $\mathcal{M}(x_i) = \mathbf{p}_{x_i}$, which assigning word x_i to its prototype $\mathbf{p}_{x_i}$, refered as **word assignment**. The other is obtaining initial prototypes $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^{n_p}$, refered as **prototype initialization**. We enhance word assignment and prototype initialization from the semantic and task perspective, as the function of a word is not solely determined by its semantics, but also by its role in the target task.

3.2.1 Token-Level Task

In the token-level task, each word is assigned a label, which corresponds to the initial definition of the prototype (Snell et al., 2017; Ji et al., 2022). As the result, we take the label of the word as the indicator of prototype for token-level tasks.

Given a token-level dataset $\mathcal{D}_t = \{(X_i, Y_i)\}_{i=1}^{N}$, where $(X_i, Y_i) = \{x_j, y_j\}_{j=1}^{n}$, we first use the client model f_{θ_c} to traverse the dataset $\mathcal{D}_t$ and obtain the representations $\{\mathbf{H}_i\}_{i=1}^{N}$ where $\mathbf{H}_i = f_{\theta_c}(X_i)$. These contextual representations can provide semantic information for prototype initialization. Then we assign words with the identical label to the same prototype and take the average representation of words within a particular class as the initial prototype. Suppose there are k representations belong to the label c, the prototype $\mathbf{p}_c$ of label c can be represented as:

$$\mathbf{p}_c = \frac{1}{k}\sum_{j=1}^{k}\mathbf{h}_j^c, \tag{1}$$

where $\mathbf{h}_j^c$ is the j-th representation of label c and k is the number of representations in label c.

In this way, we leverage the task information from the label to guide the word assignment $\mathcal{M}$, and subsequently utilize the semantic information from the word representations to obtain the prototype initialization $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^{n_p}$, where n_p denotes the number of labels.

3.2.2 Sentence-Level Task

Unlike token-level tasks, there are no natural prototype indicators for words in the sentence-level dataset. To tackle this problem, we perform a clustering algorithm on the representations, using the clustering results to indicate word assignment and prototype initialization.

To perform clustering, we need to prepare a representation for each word. Similar to the token-level task, we first use client model f_{θ_c} to traverse sentence-level dataset $\mathcal{D}_s$ and obtain $\{\mathbf{H}_i\}_{i=1}^{N}$. For word x_i that appears repeatedly in different contexts, we calculate the average of their representations to obtain the final word representation $\hat{\mathbf{x}}_i = \frac{1}{k}\sum_{j=1}^{k}\mathbf{h}_i^x$, where $\mathbf{h}_j^{x_i}$ is the j-th word representation of word x_i and k is the number of words x_i occurs in $\mathcal{D}_s$. Finally, we get $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_i\}_{i=1}^{n_x}$ where n_x is the number of unique words in D_s, and perform K-Means on $\hat{\mathbf{X}}$:

$$\mathcal{M}, \mathcal{P} = Kmeans(\hat{\mathbf{X}}), \tag{2}$$

the clustering algorithm assigns semantically similar words to the same cluster, thus completing the word assignment $\mathcal{M}$. The centroid of the clusters is used as the prototypes initialization $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^{n_p}$ where n_p here is the pre-defined number of clusters.

However, it is not appropriate to assign all semantically similar words to the same cluster. For example, in sentiment analysis, the word representations of "good" and "bad" may be similar in representation space and are often assigned to the same cluster, but if they play the same role in a sentence, it can lead to ambiguity in the sentiment of the sentence. We refer to words that are highly relevant to specific classes as task-related words, and it is important to ensure that task-related words from different classes are assigned to different prototypes.

To identify task-related words for each class, we use the TF-IDF (Salton and Buckley, 1988), a numerical statistic that reflects a word's importance in a document. In this case, we treat all sentences within a class as one document and use TF-IDF to find the keywords for each class. Subsequently, the resulting keywords are used to re-divide $\mathcal{M}$ and update $\mathcal{P}$, the algorithm of re-division is shown in Appendix A.3.

3.3 Private Representation Training

In the training phase, we use prototypes to encourage functionally similar word representations to be clustered in the representation space and apply random perturbation for preserving privacy.

Clustering Loss. Given the input text $X = \{x_i\}_{i=1}^{n}$ and word assignment $\mathcal{M}$, we first use client model f_{θ_c} to get the word representation $\mathbf{H} = \{\mathbf{h}_i\}_{i=1}^{n}$, then use center loss (Wen et al., 2016) to make representation close to its prototype:

$$\mathcal{L}_{close} = \frac{1}{2}\sum_{i=1}^{n}||\mathbf{h}_i - \mathbf{p}_{x_i}||_2^2, \tag{3}$$

where $\mathbf{p}_{x_i} = \mathcal{M}(x_i)$ is the prototype of x_i. Furthermore, we also pull away the distance between different prototypes to prevent these prototypes collapse during training (Li et al., 2021a), thus enhancing task performance. The prototype distance loss is formulated as:

$$\mathcal{L}_{away} = \frac{2}{n_p(n_p-1)}\sum_{i=1}^{n_p}\sum_{j=i+1}^{n_p}||\mathbf{p}_i - \mathbf{p}_j||_2^2, \tag{4}$$

where n_p is the number of prototypes. We refer to $\mathcal{L}_{close}$ and $\mathcal{L}_{away}$ together as $\mathcal{L}_{cluster}$.

Random Perturbation. We apply a random perturbation to each representation $\mathbf{h}_i$, which shifts the representation to another point near its prototype. Following Plant et al. (2021), we take the Laplace Noise as the random perturbation. The perturbed representations are sent to the server model f_{θ_s} for subsequent computation:

$$\hat{\mathbf{Y}} = f_{\theta_s}(\mathbf{H} + Lap(\epsilon)), \tag{5}$$

where $\hat{\mathbf{Y}}$ is the prediction results and ϵ is the hyperparameter to control the scale of noise. The perturbation is applied in both the training and inference phases. Because each perturbation is random, it is difficult for a privacy attacker to establish a link between the perturbed representation and the original word. Perturbed representations deviate from the original words but still serve original functions, thus preserving privacy while maintaining performance.

Overall Loss. The supervised task loss $\mathcal{L}_{task}$ is joint learning in a multi-task learning manner, the overall loss for our model is:

$$\mathcal{L} = \mathcal{L}_{task} + \gamma_1 \mathcal{L}_{close} + \gamma_2 \mathcal{L}_{away}, \qquad (6)$$

where γ_1 and γ_2 are weighting factors. Inspired by Li et al. (2021a), we perform the clustering algorithm at the beginning of each epoch to make clustering results more accurate. During the training phase, the client model and server model are optimized together by service providers. During the inference phase, the client performs lightweight inference using the client model, then shares obfuscated representations with service providers and potential privacy attackers. Aside from perturbations, the inference phase of our method is the same as standard PLMs, thus, we do not introduce additional inference time.

4 Experiment

4.1 Datasets

To verify the effectiveness of our methods, we conduct experiments on both token classification and sentence classification tasks, covering named entity recognition: **CoNLL2003** (Tjong Kim Sang and De Meulder, 2003) and **OntoNotes5.0** (Weischedel et al., 2013), sentiment analysis: **SST-2** (Socher et al., 2013) and topic classification: **AGNEWS**, (Zhang et al., 2015). These tasks are close to real-world applications, which can verify the actual utility of our methods. The statistics of datasets are shown in Appendix A.1.

4.2 Baselines

4.2.1 Attack Methods

We use three recently proposed text reconstruction methods for privacy attacks. **KNN-Attack** (Qu et al., 2021) computes the distance between each representation and public word embedding matrix and takes the nearest word in the embedding matrix as the attack result. The attacker can be anyone who has access to the client's representation. **Inversion-Attack** (Höhmann et al., 2021) requires the attacker to train an inversion model, which directly transforms the client representation to a word in a one-to-one manner. The attacker can be the service provider with access to the client and server model to generate training data for the inversion model. **MLC-Attack** (Song and Raghunathan, 2020) also trains an inversion model like Inversion-Attack, but it runs in a multi-label classification manner and predicts a set of words in the sentence independent of their word ordering.

4.2.2 Defence Methods

We compare our **TextObfuscator** with three representative privacy-preserving methods and standard **Fine-tune** (Devlin et al., 2018). **DPNR** (Lyu et al., 2020b) uses differential privacy and word dropout to provide a privacy guarantee. **CAPE** (Plant et al., 2021) further adopts differential privacy and adversarial training to reduce privacy information in representation. **SanText+** (Yue et al., 2021b) replaces the sensitive words in plain text based on differential privacy and word frequency.

4.3 Privacy Metrics

TopK is a token-level metric that measures the percentage of correct words in the attacker's top k predictions. **RougeL** (Lin, 2004) is a generation metric that measures the overlap between two sentences. We follow Gupta et al. (2022) and take it as a sentence-level metric to measure the coherence of attack results. **Set** is a metric specific to MLC-Attack, which quantifies the proportion of words in original sentence that are present in prediction set. Details of metrics are shown in Appendix A.2.

4.4 Experimental Settings

In our experiments, all methods are implemented based on $roberta_{base}$ (Liu et al., 2019). We divide the model into a smaller client model f_{θ_c} with three transformer layers and a large server model f_{θ_s} with the remaining nine transformer layers. The privacy attack methods are all performed in the output representations of f_{θ_c}, which will be shared with service providers and under the risk of privacy leakage. For the privacy defence methods, DPNR, CAPE, and our TextObfuscator are applied to the output representation of f_{θ_c}, and SanText+ is applied to the input text directly. The

Dataset	Method	Acc/F1 ↑	KNN-Attack ↓			Inversion-Attack ↓			MLC-Attack ↓
			Top1	Top5	Rouge	Top1	Top5	Rouge	Set
CoNLL2003	Fine-tune	91.72	87.33	97.72	90.89	99.99	100	99.90	41.41
	DPNR	79.14	**0.03**	**0.47**	0.99	14.60	28.91	11.54	10.21
	CAPE	84.47	**0.03**	0.51	**0.82**	10.39	22.28	8.94	9.37
	SanText+	76.94	60.59	75.29	50.80	81.54	87.68	69.18	13.36
	Ours	**89.11**	0.24	1.42	1.01	**6.18**	**18.56**	**5.44**	**8.32**
OntoNotes5	Fine-tune	89.68	80.18	98.17	92.65	100	100	100	71.13
	DPNR	72.38	0.07	**0.73**	1.72	18.18	33.94	17.62	15.87
	CAPE	85.89	**0.05**	0.82	**1.31**	14.57	30.02	14.25	**13.62**
	SanText+	71.57	57.35	73.40	51.21	78.99	86.07	68.05	46.90
	Ours	**87.17**	0.68	2.31	2.13	**7.97**	**22.22**	**9.88**	**13.62**
SST-2	Fine-tune	94.38	88.21	98.75	96.04	100	100	100	62.09
	DPNR	87.84	0.02	1.76	0.87	**4.39**	16.39	**5.72**	16.82
	CAPE	89.44	**0.03**	1.86	**0.70**	5.06	**16.15**	6.37	17.12
	SanText+	87.27	70.78	75.95	60.05	81.79	89.01	69.17	52.73
	Ours	**91.51**	0.05	**0.47**	0.87	5.48	17.97	11.35	**15.74**
AGNEWS	Fine-tune	94.71	89.45	98.87	96.37	100	100	100	86.13
	DPNR	93.12	0.02	2.32	1.79	3.97	13.53	6.82	15.86
	CAPE	93.99	**0.02**	3.41	1.58	3.39	12.60	2.22	14.26
	SanText+	91.92	59.31	64.58	51.57	78.20	85.11	70.86	61.36
	Ours	**94.52**	0.04	**0.53**	**1.12**	**3.38**	**12.37**	**2.01**	**13.16**

Table 1: Main results on privacy and task performance evaluation. Task metric for SST-2 and AGNEWS is accuracy and for CoNLL2003 and OntoNotes5 is F1. Bold term means the best result except Fine-tune. **Acc/F1** higher is better, meaning high task performance. **Top1**, **Top5** and **Rouge** lower is better, meaning low privacy leakage.

implementation details and hyperparameters are shown in Appendix A.4.

4.5 Main Results

Table 1 shows the main results of our method and all baselines. We can observe that: **(1) Representations without defence method are vulnerable to privacy attacks.** In the absence of any privacy defence methods, all privacy attacks on Fine-tune are highly successful. Inversion-attack even achieves 100% top-1 attack accuracy, indicating that privacy is fully compromised. **(2) Resisting Inversion-Attacks is key to protecting privacy.** Most defence methods can resist KNN-Attack, it only achieves nearly 0 Top1 and Top5 attack accuracy for all tasks. In the case of MLC-Attack, the attack results are a set of disordered words that may contain redundancy. It is hard to reconstruct the correct sequence from such words. However, for Inversion-Attack, the representation is transformed into the original word one-to-one, and it achieves the highest attack accuracy, which is the most likely to compromise privacy. **(3) Previous defence methods achieve limited task performance.** CAPE and DPNR show good privacy on sentence-level tasks, Inversion-attack on these methods only achieve about 5% top 1 attack accuracy on SST2 and AGNEWS. But for token-level tasks which require richer word information, these methods not only degraded privacy but also suffered significantly from task performance. We speculate that these methods reduce the privacy information, i.e., word information, in the representation, which hinders the understanding of the sentence. There is an inherent contradiction between reducing word information on shallow representations and maintaining task performance, especially on token-level tasks. **(4) With equal or better privacy, our proposed TextObfuscator shows a task performance improvement over baselines.** The advantage of the TextObfuscator is that we do not reduce private information but rather obfuscate clustered representations, which misleads privacy attackers while still preserving functionality for each word representation, thus achieving better task performance while maintaining privacy.

5 Analysis

5.1 Ablation Study

Effect of Different Components. To verify the effectiveness of the different components ($\mathcal{L}_{close}$,

$\mathcal{L}_{away}$ and random perturbation) in our method, we conduct a series of ablation experiments and show the results in Table 2. We can observe that: (1) Without cluster loss ($\mathcal{L}_{close}$ and $\mathcal{L}_{away}$), random perturbation alone is inadequate as a defence against privacy attacks. We speculate that the perturbation applied to unclustered representations can only provide limited obfuscation to the attacker. Most perturbed words still maintain a distance from other words, providing attackers the opportunity to distinguish between them. (2) Cluster loss without random perturbation is completely indefensible against Inversion-Attack. The powerful inversion model can still distinguish different words from clustered representations. **Only the combination of clustered representations and random perturbations can effectively mislead privacy attackers**. (3) Without the $\mathcal{L}_{away}$, some prototypes tend to collapse to one point, resulting in a decline in task performance but a privacy boost.

Dataset	Method	Task↑	KNN ↓	Inversion ↓
SST-2	**TextObfuscator**	91.17	0.05	**6.01**
	w/o $\mathcal{L}_{away}$	90.37	**0.00**	6.47
	w/o $\mathcal{L}_{cluster}$	90.13	4.29	31.44
	w/o *Perturb*	**93.12**	**0.00**	100
CoNLL03	**TextObfuscator**	89.11	0.26	**7.02**
	w/o $\mathcal{L}_{away}$	88.44	**0.23**	7.41
	w/o $\mathcal{L}_{cluster}$	89.06	2.21	31.60
	w/o *Perturb*	**91.42**	0.05	100

Table 2: Ablation Study on our method. KNN and Inversion-attack use Top1 accuracy.

Effect of Clustering Algorithms. Sentence classification tasks require two additional processes, clustering and re-division algorithms. We conduct experiments on SST-2 to verify the performance and privacy impact of the cluster number and TF-IDF-based re-division. From experimental results shown in Figure 3, we can observe that re-division (KMeans-TFIDF) consistently improves task performance and privacy for all cluster numbers. Besides that, we find that a large cluster number can damage privacy (lower Top1 means better privacy), and a small cluster number can lead to a degradation in task performance. Therefore, a moderate cluster number is deemed to be optimal.

5.2 Visualisation

Representation Visualisation. To intuitively show the influence of the cluster loss and random perturbation, we employed T-SNE (Van der Maaten and Hinton, 2008) to visualize representations of TextObfuscator. Specifically, we select six classes from the CoNLL2003 dataset and utilized the full test set to generate these representations. From the visualization results in Figure 4, we can observe that, before perturbing (triangular point), functionally similar representations are clustered together while maintaining a certain distance from other clusters. After perturbing (round point), the representations are shifted and mixed with surrounding representations but remain within their respective functional clusters. Such representations obfuscate word information as they are indistinguishable from each other, but maintain word function as they still perform the same function in the representation space. We take NER as an example, perturbing the word representation of "John" may result in it being similar to another word, such as "Mike". However, a privacy attacker will only be able to establish a false association between the representation of "Mike" and the original word "John", thereby effectively protecting privacy. But for NER task, both words "John" and "Mike" serve the same role as "PER (Person)" and do not negatively impact the model's ability to classify them. These visualization results provide empirical evidence for the principles and effectiveness of TextObfuscator.

Attack Results Visualisation. To intuitively show the effectiveness of our privacy-preserving method, we visualize the results of privacy attacks for one sample from OntoNotes5. As shown in Table 3, we can observe that the attack results on TextObfuscator are largely unreadable, with only some high-frequency words “the” and wrong words

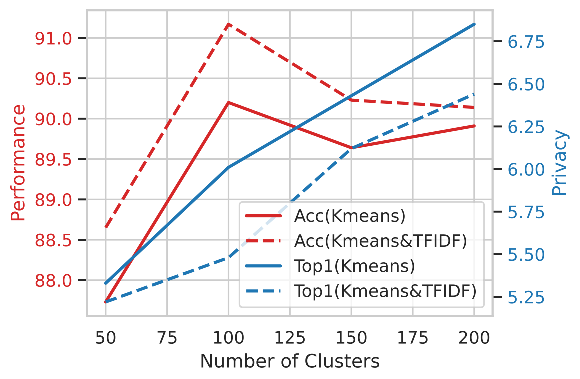

Figure 3: Privacy and task performance under different cluster numbers. Accuracy is used for performance and Top1 under Inversion-Attack is used for privacy.

Input text:	**President Bush** called his attention to the matter during the **Italian leader’s visit** here **last week**.
Fine-tune	**KNN:** **President Bush** his attention to matter during **Italian leader ’s visit** here **last week**.
	Inversion: **President Bush** called his attention to the matter during the **Italian leader’s visit** here **last week**.
	MLC: { **Bush** | **President** | **visit** | **Italian** | **week** | **last** | **’** | **s** | the | called | here | during | to | his | . | this }
Text Obfuscator	**KNN:** ...... anybody ls <= our Israeliibble >ancial clinicians, Wednesday Sag Jin relocation teleport.
	Inversion: the The Putin the the the the the the the the Israeli the the the the the next year,
	MLC: { to | the | . | in, }

Table 3: Results of privacy attack. Text in red represents successfully recovered words. Text in bold means **privacy information**. Attacks on TextObfuscator only recover meaningless words, no useful information is leakaged.

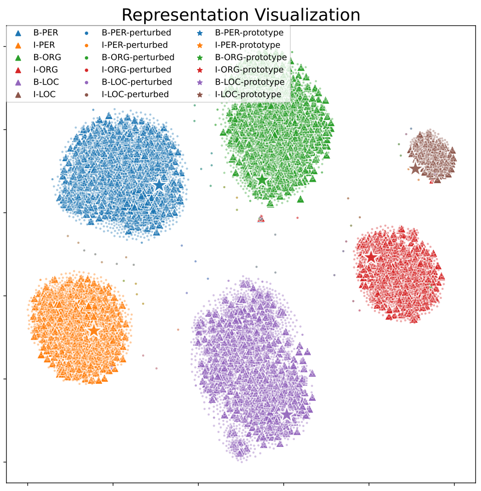

Figure 4: Visualization of representations. Obfuscated representations are indistinguishable from surrounding word representations but still remain in their clusters.

such as “Putin” being recovered. Keywords such as people, places, and time that may contain privacy have not been recovered correctly, indicating that our method is effective in protecting privacy.

6 Related Work

The high performance and computational cost of PLMs have accelerated the development of inference services (Soifer et al., 2019; Pais et al., 2022). These services enable clients to perform compute-intensive PLM inference in the cloud by uploading personal data, which brings convenience but also raises concerns about privacy (Zhang et al., 2021; Liu et al., 2020a).

In order to mitigate privacy leakage, many sought to upload representations that have been privatized by privacy-preserving technologies instead of the original text to the cloud (DALE, 2015). One method is to encrypt the representation, using either homomorphic encryption (Chen et al., 2022b) or a customized encryption protocol (Hao et al., 2022) to enable computations to be performed on the encrypted representation. Encryption-based methods often require high computation time and communication costs (Gilad-Bachrach et al., 2016) and may not be practical for real-world applications. Therefore, we did not compare this method in our experiments. Another method is to use Differential privacy (Xu et al., 2020; Lyu et al., 2020a; Yue et al., 2021a; Hoory et al., 2021) and adversarial training (Coavoux et al., 2018; Plant et al., 2021; Chen et al., 2022a) to learn private representation, which reduces privacy attributes in representation. Applying these works to reduce word information leads to limited performance, as the word information in the shallow layer is important for subsequent inference. Our method proposes to obfuscate word information while maintaining word functionality, thus providing better performance and privacy. Recently, Zhou et al. (2022a) propose TextFusion, which utilizes token fusion to hinder privacy attackers from training a targeted inversion model. We explore a stronger and more realistic setting than TextFusion, where the privacy attacker is the service provider itself. As the service provider is aware of TextFusion’s defense strategies, they can design targeted privacy attack methods to disclose more sensitive information. We did not compare our method with TextFusion due to different settings.

In addition to NLP, there are also many works for protecting inference privacy in computer vision (Xiang et al., 2019; Osia et al., 2020; Liu et al., 2020b), However, most of these methods cannot be used directly in NLP because they only consider one single image, and we need to protect the privacy of a sequence of words. The popularity of transformer structures (Dosovitskiy et al., 2020) in computer vision may alleviate this situation, but the adaptation of these methods still requires further exploration.

7 Conclusion

In this paper, we propose TextObfuscator, a novel representation learning method for privacy-preserving inference. The main idea of our method is to obfuscate word information and maintain word functionality. We achieve this by applying random perturbations to the clustered representations. The perturbed representations are indistinguishable from the surrounding representations but still around their functional clusters. To learn clustered representation, we find prototypes for each word and encourage the word representation to be close to its prototype. Additionally, we propose different methods to find prototypes for token-level and sentence-level tasks, utilizing semantic and task information. Through experiments on token and sentence classification tasks, we evaluate the effectiveness of TextObfuscator and provide further analysis of the principles of our proposed method. Overall, our results suggest that TextObfuscator is a promising method for preserving inference privacy.

8 Limitations

We summarize the limitations of our method as follows: (1) TextObfuscator was designed to protect word privacy in the inference phase, and we did not verify its ability to preserve other privacy attributes and training phase privacy. (2) Although we have done empirical experiments and visualizations to demonstrate the effectiveness of our method, a mathematical proof would enhance its privacy guarantees. (3) Our method requires more training steps than fine-tuning, resulting in an increased computational cost.

Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No.62206057,62076069,61976056), Shanghai Rising-Star Program (23QA1400200), Program of Shanghai Academic Research Leader under grant 22XD1401100, and Natural Science Foundation of Shanghai (23ZR1403500).

References

Sam Altman. 2022. Openai api. URL. `https://openai.com/api/`.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Jikun Chen, Feng Qiang, and Na Ruan. 2022a. Adversarial representation sharing: A quantitative and secure collaborative learning framework. *arXiv preprint arXiv:2203.14299*.

Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, and Jianxin Li. 2022b. The-x: Privacy-preserving transformer inference with homomorphic encryption. *arXiv preprint arXiv:2206.00216*.

Maximin Coavoux, Shashi Narayan, and Shay B Cohen. 2018. Privacy-preserving neural representations of text. *arXiv preprint arXiv:1808.09408*.

ROBERT DALE. 2015. Nlp meets the cloud. *Natural Language Engineering*, 21(4):653–659.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An image is worth 16x16 words: Transformers for image recognition at scale.

Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178.

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

Samyak Gupta, Yangsibo Huang, Zexuan Zhong, Tianyu Gao, Kai Li, and Danqi Chen. 2022. Recovering private text in federated learning of language models. *arXiv preprint arXiv:2205.08514*.

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

Johannes Höhmann, Achim Rettinger, and Kai Kugler. 2021. Invbert: Text reconstruction from contextualized embeddings used for derived text formats of literary works. *arXiv preprint arXiv:2109.10104*.

Shlomo Hoory, Amir Feder, Avichai Tendler, Sofia Erell, Alon Peled-Cohen, Itay Laish, Hootan Nakhost, Uri Stemmer, Ayelet Benjamini, Avinatan Hassidim, et al. 2021. Learning and evaluating a differentially private pre-trained language model. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1178–1189.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Bin Ji, Shasha Li, Shaoduo Gan, Jie Yu, Jun Ma, Huijun Liu, and Jing Yang. 2022. Few-shot named entity recognition with entity-level prototypical network enhanced by dispersedly distributed prototypes. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1842–1854, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Ryan Lehmkuhl, Pratyush Mishra, Akshayaram Srinivasan, and Raluca Ada Popa. 2021. Muse: Secure inference resilient to malicious clients. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2201–2218.

Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. 2021a. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*.

Linyang Li, Demin Song, Ruotian Ma, Xipeng Qiu, and Xuanjing Huang. 2021b. Knn-bert: fine-tuning pre-trained models with knn classifier. *arXiv preprint arXiv:2110.02523*.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. *arXiv preprint arXiv:1805.06093*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Ximeng Liu, Lehui Xie, Yaopeng Wang, Jian Zou, Jinbo Xiong, Zuobin Ying, and Athanasios V Vasilakos. 2020a. Privacy and security issues in deep learning: A survey. *IEEE Access*, 9:4566–4593.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Zhijian Liu, Zhanghao Wu, Chuang Gan, Ligeng Zhu, and Song Han. 2020b. Datamix: Efficient privacy-preserving edge-cloud inference. In *European Conference on Computer Vision*, pages 578–595. Springer.

Lingjuan Lyu, Xuanli He, and Yitong Li. 2020a. Differentially private representation for nlp: Formal guarantee and an empirical study on privacy and fairness. *arXiv preprint arXiv:2010.01285*.

Lingjuan Lyu, Xuanli He, and Yitong Li. 2020b. Differentially private representation for NLP: Formal guarantee and an empirical study on privacy and fairness. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2355–2365, Online. Association for Computational Linguistics.

Ruotian Ma, Xin Zhou, Tao Gui, Yiding Tan, Linyang Li, Qi Zhang, and Xuan-Jing Huang. 2022. Template-free prompt tuning for few-shot ner. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5721–5732.

Seyed Ali Osia, Ali Shahin Shamsabadi, Sina Sajadmanesh, Ali Taheri, Kleomenis Katevas, Hamid R Rabiee, Nicholas D Lane, and Hamed Haddadi. 2020. A hybrid deep learning architecture for privacy-preserving mobile analytics. *IEEE Internet of Things Journal*, 7(5):4505–4518.

Sebastião Pais, João Cordeiro, and M Luqman Jamil. 2022. Nlp-based platform as a service: a brief review. *Journal of Big Data*, 9(1):1–26.

Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy risks of general-purpose language models. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1314–1331. IEEE.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Richard Plant, Dimitra Gkatzia, and Valerio Giuffrida. 2021. CAPE: Context-aware private embeddings for private language learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7970–7978, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

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

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Jonathan Soifer, Jason Li, Mingqin Li, Jeffrey Zhu, Yingnan Li, Yuxiong He, Elton Zheng, Adi Oltean, Maya Mosyak, Chris Barnes, et al. 2019. Deep learning inference service at microsoft. In *2019 USENIX Conference on Operational Machine Learning (OpML 19)*, pages 15–17.

Congzheng Song and Ananth Raghunathan. 2020. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 377–390.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. 2001. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. 2016. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pages 499–515. Springer.

Liyao Xiang, Haotian Ma, Hao Zhang, Yifan Zhang, Jie Ren, and Quanshi Zhang. 2019. Interpretable complex-valued neural networks for privacy protection. *arXiv preprint arXiv:1901.09546*.

Zekun Xu, Abhinav Aggarwal, Oluwaseyi Feyisetan, and Nathanael Teissier. 2020. A differentially private text perturbation method using a regularized mahalanobis metric. *arXiv preprint arXiv:2010.11947*.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. *arXiv preprint arXiv:2005.07150*.

Xiang Yue, Minxin Du, Tianhao Wang, Yaliang Li, Huan Sun, and Sherman S. M. Chow. 2021a. Differential privacy for text analytics via natural text sanitization. In *Findings, ACL-IJCNLP 2021*.

Xiang Yue, Minxin Du, Tianhao Wang, Yaliang Li, Huan Sun, and Sherman S. M. Chow. 2021b. Differential privacy for text analytics via natural text sanitization. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3853–3866, Online. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Xiaoyu Zhang, Chao Chen, Yi Xie, Xiaofeng Chen, Jun Zhang, and Yang Xiang. 2021. Privacy inference attacks and defenses in cloud-based deep neural network: A survey.

Xin Zhou, Jinzhu Lu, Tao Gui, Ruotian Ma, Zichu Fei, Yuran Wang, Yong Ding, Yibo Cheung, Qi Zhang, and Xuanjing Huang. 2022a. TextFusion: Privacy-preserving pre-trained model inference via token fusion. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8360–8371, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xin Zhou, Ruotian Ma, Yicheng Zou, Xuanting Chen, Tao Gui, Qi Zhang, Xuan-Jing Huang, Rui Xie, and Wei Wu. 2022b. Making parameter-efficient tuning more efficient: A unified framework for classification tasks. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 7053–7064.

A Appendix

A.1 Statistics of Dataset

We use four English datasets, including SST-2 (Socher et al., 2013) for sentiment classification (Li et al., 2021b; Zhou et al., 2022b), AG-NEWS (Zhang et al., 2015) for topic classification, CoNLL2003 (Tjong Kim Sang and De Meulder, 2003) and OntoNotes5 (Weischedel et al., 2013) for named entity recognition (Yu et al., 2020; Ma et al., 2022). We follow the official dataset split for AGNEWS, CoNLL2003 and OntoNotes5. The test set for SST-2 is not publicly available, the reported results of SST-2 tasks are tested on the official development set. The statistics of datasets in our experiments are shown in Table 4.

Dataset	Domain	# Train	#Test	#Labels
SST-2	Movie	67349	872	2
AGNEWS	News	120000	7600	4
CoNLL2003	News	14041	3453	9
OntoNotes5	General	59924	8262	37

Table 4: Statistics of the datasets.

A.2 Privacy Metrics

In our experiments, we use three metrics to measure privacy. Next, we will describe these three metrics in a formulaic way.

TopK. Top-K accuracy is defined as the proportion of times that the real words is among the top K predictions made by the attack model, where K is a pre-defined parameter. Mathematically, it can be represented as:

$$TopK = \frac{1}{N}\sum_{i=1}^{N}[y_i \in top_k(p_i)] \tag{7}$$

where N is the total number of representation, y_i is the real word of representation i, p_i is the predicted probability distribution of the attack model, and $top_k(p_i)$ is the set of top k words with highest probability for representation i.

RougeL (Lin, 2004). RougeL is a widely used metric to evaluate the quality of text summarization. So we do not describe the details of RougeL, but just state that we take the top1 word of the attack results to compose the sentences for calculating RougeL.

Set. The attack results of MLC-Attack are unordered sets of words, a one-to-one metric like TopK cannot be used for this attack, so we use Set to measure the attack success rate of MLC-Attack. Given that set A is different words in a sentence, the set B is the prediction results of MLC-Attack, the Set metirc can be represented as:

$$Set = \frac{|A \cap B|}{|A|} \tag{8}$$

This metric measures how many words in the original sentence are in the set of predicted results of the MLC-Attack.

Algorithm 1 Re-division Algorithm

Require: Representations Matrices $\hat{\mathbf{X}}$; Word Assignment $\mathcal{M}$; Prototype Initialization $\mathcal{P}$; Task-Related Words T.

```
1:  for $t_c \in T$ do
2:      // Set of prototypes assigned to other classes
3:      $conflict \leftarrow \{\mathcal{M}(x) \in T | x \notin t_c\}$
4:      for $x \in t_c$ do
5:          // Divide $x$ to other prototype if conflict occurs
6:          if $\mathcal{M}(x) \in conflict$ then
7:              $\hat{\mathbf{x}} \leftarrow$ get representation of $x$ from $\hat{\mathbf{X}}$
8:              $\mathcal{M}(x) \leftarrow \text{argmin}_{j \notin conflict}\, d(\hat{\mathbf{x}}, \mathbf{p}_j)$
9:          end if
10:     end for
11: end for
12: // Update $\mathcal{P}$ based on the new $\mathcal{M}$
13: for $\mathbf{p}_i \in \mathcal{P}$ do
14:     $P_i \leftarrow \{\mathbf{x} \in \hat{\mathbf{X}} | \mathcal{M}(\mathbf{x}) = \mathbf{p}_i\}$
15:     $\mathbf{p}_i \leftarrow \frac{1}{|P_i|}\sum_{\mathbf{x} \in P_i} \mathbf{x}$
16: end for
17: return $\mathcal{M}, \mathcal{P}$
```

A.3 Re-division Algorithm

As mentioned in Section 3.2.2, after we find the category-related words $T = \{t_c\}_{c=1}^{n_c}$ for each category using TF-IDF, we re-divide the word Assignment $\mathcal{M}$ and prototype Initialization $\mathcal{P}$. The algorithm is inspired by constrained K-means clustering (Wagstaff et al., 2001), but we only apply it once after clustering. The process of re-division is shown in Algorithm 1.

A.4 Implementation Details

In this subsection, we describe the implementation details and the replication process of both attack and defence methods. All methods are based on $Roberta_{base}$ with 125 million parameters. Dataset and models are all loaded from huggingface[2]. Our experiments are conducted on NVIDIA GeForce RTX 3090.

Details for Defence Methods. We reference publicly available code, implement DPNR[3], CAPE[4] and SanText+[5] ourselves. We also conduct a grid search on the hyperparameters to reproduce the baselines on our setting. For each defence method, we train 50 epochs on SST2, CoNLL2003 and Ontonotes5, 30 epochs on AGNews to guarantee convergence, the AdamW optimizer and a linear learning rate scheduler are used during training.

[2]https://huggingface.co/
[3]https://github.com/xlhex/dpnlp
[4]https://github.com/NapierNLP/CAPE
[5]https://github.com/xiangyue9607/SanText

The default learning rate is 5e-5, we do not adjust the learning rate unless we encounter the case of non-convergence. For **DPNR**, we search noise scale ϵ on [0.05, 0.1, 0.5, 1, 5] and the word dropout rate on [0, 0.1, 0.3]. For **CAPE**, we search the adversarial training weights λ on [0.01, 0.05, 0.1, 0.5, 1, 5] and noise scale ϵ on [0.05, 0.1, 0.5, 1, 5]. For **Santext+**, we follow the author's setting and use GloVe (Pennington et al., 2014) to guide the word replacement, the probability of non-sensitive words to be sanitized p defaults to 0.3 and the sensitive word percentage w defaults to 0.9. We search the privacy parameter ϵ on [1, 2, 3]. For **TextObfuscator**, we use the K-Means to cluster representation for sentence-level tasks, and the number of clusters defaults to 100. We search the close loss weights γ_1 from [0.1, 0.5, 1] and away loss weights γ_2 from [0.1, 0.3, 0.5]. Although the noise scale can also be adjusted, we found that the most commonly used parameter (ϵ=1) is sufficient, so we kept the noise scale constant in all experiments. We select the best performance and privacy (but prefer privacy) results from the experimental results to report. The best hyperparameters we tuned are shown in Table 5.

Dataset	Method	lr	bsz	λ_{adv}	ϵ_n	ϵ_w	ϵ_p	γ_1	γ_2
	Finetune	2e-5	32	-	-	-	-	-	-
	DPNR	1e-5	64	-	5	0.1	-	-	-
CoNLL2003	CAPE	1e-5	32	0.1	5	-	-	-	-
	Santext+	1e-5	64	-	-	-	3	-	-
	TextObfuscator	5e-5	128	-	1	-	-	0.5	0.3
	Finetune	2e-5	32	-	-	-	-	-	-
	DPNR	1e-5	64	-	5	0.1	-	-	-
OntoNotes	CAPE	1e-5	32	0.05	5	-	-	-	-
	Santext+	1e-5	64	-	-	-	3	-	-
	TextObfuscator	5e-5	128	-	-	-	-	0.5	0.3
	Finetune	2e-5	32	-	-	-	-	-	-
	DPNR	1e-5	64	-	0.5	0.1	-	-	-
SST-2	CAPE	1e-5	32	0.1	0.5	-	-	-	-
	Santext+	1e-5	64	-	-	-	3	-	-
	TextObfuscator	1e-5	256	-	-	-	-	0.5	0.1
	Finetune	2e-5	32	-	-	-	-	-	-
	DPNR	1e-5	64	-	1	0.1	-	-	-
AGNEWS	CAPE	1e-5	32	0.1	0.5	-	-	-	-
	Santext+	1e-5	64	-	-	-	1	-	-
	TextObfuscator	5e-5	168	-	-	-	-	0.5	0.1

Table 5: Hyperparameters of best restults for defence methods. - means the hyperparameter is not used in this method. λ_{adv} is the adversarial training weights for CAPE. ϵ_n is the noise scale used in CAPE, DPNR and TextObfuscator. ϵ_w and ϵ_p are used in Santext+, representing the word dropout rate and privacy parameter, respectively. γ_1 and γ_2 are the close loss weight and away loss weight for TextObfuscator.

Details for Attack Methods. In our implementation of the **KNN-Attack**, we employed the use of the embedding matrix of the $roberta_{base}$ to calculate the Euclidean distance between the client representation. When attack the CAPE and DPNR methods, which employ max-min normalization on the representation, we also applied the same normalization technique on the embedding matrix before calculating distance. For **Inversion-Attack and MLC-Attack**, the training data is generated by the client model to be attacked on the training set of the target task. We use the $Roberta_{base}$ model as the backbone for the inversion model and search the learning rate from [1e-4, 1e-5, 1e-6] and train 10 epochs to guarantee convergence. We take the words with a probability higher than 0.5 as the prediction result of MLC.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
section 8

☑ A2. Did you discuss any potential risks of your work?
section 8

☑ A3. Do the abstract and introduction summarize the paper's main claims?
abstract and section 1

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

B ☑ Did you use or create scientific artifacts?

section 3 and section 4

☑ B1. Did you cite the creators of artifacts you used?
section 4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Appendix A.1

☑ 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)?
Section 4 and section 5

☒ 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?
The datasets we use are publicly available. And we need to perform NER tasks that involve identifying the names of people, which are usually not anonymized.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
section 4 and Appendix A.1

☑ 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.
Appendix A.1

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix A.4

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?
Appendix A.4

☑ 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?
Appendix A.4

☑ 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.)?
Appendix A.4

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.?
No response.

☐ 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)?
No response.

☐ 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?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.